

UNIVERSITY OF NAVARRA
SCHOOL OF ENGINEERING
DONOSTIA-SAN SEBASTIÁN



DEVELOPMENT AND IMPROVEMENT OF OPTICAL
TRACKING METHODS TOWARDS REGISTERING THE
DEFORMATIONS OF 3D NON-RIGID BODIES IN
REAL TIME FOR AUGMENTED REALITY
APPLICATIONS

DISSERTATION
submitted for the
Degree of Doctor of Philosophy
of the University of Navarra by

Ibai Leizea Alonso

under the supervision of

**Diego Borro Yáguez and
Hugo Álvarez Ponga**

February, 2015

Nire aitite-amamentzat

Agradecimientos

Estos últimos años han servido para mucho más de lo que queda reflejado en estas páginas. A lo largo de este tiempo no sólo he conseguido formarme académicamente, sino que he tenido la oportunidad de conocer a un extraordinario grupo de personas y descubrir aún más a los que ya conocía. Es por ello que a través de estas líneas quisiera agradecer a todos los que habéis querido compartir un segundo de vuestro tiempo, mostrándome vuestro apoyo e interés y sobre todo una actitud que destaca sobre el resto, *paciencia*.

En primer lugar quisiera agradecer a todos aquellos que en su día tomaron parte para que pudiera desarrollar esta tesis. Gran parte de esa confianza se la debo a mi director de tesis Diego Borro. Gracias por haber apostado por mí y animarme a conocer aquel mundo de la Realidad Aumentada y la Visión por Computador, tema del que me alegro de haber acertado en la elección. Junto a Josune, quisiera agradecerles también que me hayan dado la oportunidad de participar en la docencia donde tanto he aprendido y donde hemos disfrutado de muy buenas experiencias. Del mismo modo, agradezco a Alejo Avello, Jordi Viñolas y Luis Matey por haberme permitido desarrollar esta tesis en el Área de Simulación del CEIT. Este agradecimiento es extensible a la Universidad de Navarra, en especial a TECNUN por mis estudios de doctorado, así como al personal de administración y servicios, haciendo que el sistema burocrático resulte más fácil.

Como todas las etapas, ésta también tuvo un principio y no me quiero olvidar de Alex García-Alonso, Dimas López y en especial de Alberto Lozano por haberme animado a continuar en el mundo de la investigación. A éste último quisiera agradecerle la ayuda que he recibido desde el principio y la oportunidad de compartir grandes momentos fuera del CEIT.

Pero si hay una persona a la que tengo que agradecer que esta tesis esté en mis manos, sin lugar a dudas ese es Hugo. Gracias no sólo por haberme ayudado técnicamente (aunque parezca una frase hecha, todo lo que sé es gracias a ti), sino por el apoyo, paciencia, motivación y muchos más valores que me has enseñado en el día a día. Y por supuesto las risas que hemos compartido haciendo viajes, en monólogos, en comidas, creando el *ARDepartment*, etc. Espero que a partir de ahora sigamos organizando muchos de estos planes.

A pesar de como muy bien dice Iker A. (gracias por tu ayuda en esta última etapa y por esas conversaciones donde arreglábamos el mundo) *Nadie dijo que esto fuera a ser fácil*, los compañeros que han pasado por el departamento y el despacho han hecho que todo haya sido mucho más llevadero. Gracias a todos vosotros: Goretti (por tus chistes fáciles) y Aitor 'Moar' (tengo un paquete para ti, NO Fake!) por haber encontrado un apoyo tanto dentro como fuera (mila esker zuen laguntzagatik!). Mi compañero de fatigas I.Herrera (co-co-director de tesis técnico) y Fran, por su lado más friki y técnico ¡Sois imprescindibles! El trío A.Valero (neumomororotorax, Romani Di Roma!), Borja (Bortx) y Pedro (Oilategia!) que me alegro de haberlos conocido y con los que he pasado muy buenos momentos. Gracias también a los compañero/as con los que he compartido muchas horas de mesa y mantel, donde esas interesantes conversaciones/discusiones me han servido como vía de escape. En especial a Álvaro 'Botín' Sañudo (por ser tan claro con tus ideas y casi siempre convencerme), Ibon (el último superviviente de patxangas) y Sergio (por enseñarme a andar en bici). Las chicas de organización, en especial a Maider y Josune (zelan zauden entzutea ez da maiz gertatzen, ta asko eskertzen da, mil esker!). Ainitze (gracias por tu apoyo y por estar siempre ahí para escuchar, también por tu lado más [heavy] musical) y Pablo (por tus consejos, ¡todos!, destacando los diseño-fotográficos). Y el resto de compañero/as con los que he pasado muchas horas: Aiert (suerte con ICP/PCL!), Ilaria, Carlos, Manolo, Alfon, Luis, Borja P. (por los BI-A8-SS), Martin (simplemente, gracias), Ainara, Itziar, Xabi, Esther, patxangeros...

También quiero recordar a todos aquellos que me habéis animado y motivado fuera de estas cuatro paredes. Vuestra ayuda ha sido igual de importante o más. Entre ellos está mi kuadrila: Adrian, G.Álava, Álvaro, J.Arana (gracias por tu media tesis), Eder, G.Garai, Gotzon, Imanol, Iñigo y Mikel. Y un eskerrik asko especial a la persona que ha estado desde el minuto cero y ha tenido que batallar conmigo, A. Barron. Que poca gente aparte de mí querían con tantas ganas que acabase esta tesis. También agradecer a la kuadrila donostiarra que he conseguido a lo largo de este tiempo que he vivido en las afueras de

Bilbao: Lucía (también por la portada), Joseba, Haritz, Txell, Iosune, Unai (baita laburpenaren perfekzioa lortzeagatik ere, irtenbide eraginkorra!), Ane, Oiane, Gorriti, Bea, Ainara... Por supuesto mil gracias también al grupo porque sin vosotros no hubiera sido posible. Y en especial a Ernesto por haberme ayudado a trabajar en mi otra tesis durante todos estos años. Y muchos otros con los que me he cruzado durante este periodo y tanto me habéis apoyado: Euitianos, Sangulianos, Landetas, Itziar (and Roger, It's just a job!), Rodenses, Asturianos...

Por otra parte, merecen una mención especial dos personas muy importantes para mí, que nunca han perdido la confianza y me han apoyado desde el principio. Gracias a mi aita Iñaki y a mi ama Itziar por haber creído siempre en mí. No me olvido por supuesto de mi hermana Janire y toda la familia por estar siempre ahí, guardando un recuerdo especial de mi aitite Paco.

I finalment, i no per això menys important, vull donar les gràcies a la meva família catalana pel seu suport, especialment a la Maria, qui ha tingut el coratge d'aguantar-me, de fer-me veure que les coses són més senzilles del que un s'imagina i d'animar-me en els moments difícils. Difícil estar a la teva alçada! Moltes gràcies per la teva paciència, Marieta!

Pidiendo perdón y agradeciendo a todos aquellos que me haya podido olvidar. Por todo esto y mucho más,

Eskerrik asko.

Donostia-San Sebastián, Febrero de 2015

Ibai Leizea Alonso

Abstract

Augmented Reality (AR) is a technology that aims to embed virtual objects in the real world, showing to the user the set of objects (virtual and real) as a single world. For that purpose, it is necessary to offer a perfect alignment between virtual and real objects, which increases the effectiveness of AR. The solution to this problem is known as *tracking*. The object tracking consists in determining at any time the position and orientation of the camera relative to the scene. Optical sensors are most commonly used to overcome the tracking problem due to their low cost implementation. However, it is often difficult to provide robustness, accuracy and low computational cost at the same time.

This thesis tackles the improvement and development of the main optical tracking techniques, primarily focused on detecting the deformations of the bodies. First, it has been achieved the tracking of rigid and non-rigid planar surfaces through a monocular camera, and then, the object deformation estimation with a more complex device as a RGB-D camera has been developed.

Surface tracking systems such as those based on markers have the problem of not being able to handle occlusions. Thus, this thesis raises a new marker design that offers robustness against occlusions. Furthermore, in order to handle the deformations of surfaces, a solution that recovers the camera pose and the non-rigid surface simultaneously is proposed. Continuing with the deformation handling, it has also developed a robust tracking system for reconstructing the 3D shape of deformable objects using two different physical formulations. One offers a correct physical behaviour with a low computational cost, whereas the other achieves higher levels of accuracy at the expense of higher processing time.

In addition, all the presented solutions have the common factor that all are executed in real time, which is a key property for a fluently visual feedback of an AR application.

Resumen

La *Realidad Aumentada (RA)* es una tecnología que busca añadir objetos virtuales en el mundo real, mostrando al usuario el conjunto de objetos (virtuales y reales) como un solo mundo. Para ello, es necesario alinear correctamente los objetos reales y virtuales, lo que incrementa la eficiencia de la RA. La solución a este problema es conocido como *tracking*. El tracking de un objeto consiste en determinar en cualquier instante la posición y orientación de la cámara relativas a la escena. Los sensores ópticos son los más usados para resolver el tracking debido a su bajo coste de implementación. Sin embargo, no siempre es posible ofrecer robustez, precisión y un bajo coste computacional al mismo tiempo.

Esta tesis aborda la mejora y desarrollo de las principales técnicas de seguimiento óptico, principalmente las orientadas a la detección de deformaciones de objetos. Primero, se ha gestionado el tracking de superficies planas rígidas y deformables a través de cámaras monoculares, y después se ha desarrollado la estimación de las deformaciones de objetos mediante un dispositivo más complejo, una cámara RGB-D.

Los sistemas de tracking basados en marcadores tienen el problema de no soportar oclusiones. Así, esta tesis plantea un nuevo diseño de marcador personalizable que ofrece robustez frente a oclusiones. Asimismo, para controlar las deformaciones de las superficies, esta tesis propone una solución que calcula tanto la posición de la cámara como la deformación simultáneamente. Igualmente, se ha desarrollado un tracking robusto para la reconstrucción de la estructura 3D de objetos deformables a través de dos formulaciones físicas distintas. Una ofrece un correcto comportamiento físico y bajo coste computacional, mientras que otra alcanza mayores niveles de precisión a cambio de mayor procesamiento.

Además, todas las soluciones descritas tienen el factor común de ejecutarse en tiempo real, propiedad clave para que la respuesta visual de una aplicación RA sea fluida.

x

Laburpena

Errealitate Areagotua (EA) mundu errealean objektu birtualak gehitzea helburua duen teknologia da, erabiltzaileari objektuak (birtualak eta errealak) erakutsiz mundu bakar bat balitz bezala. Horretarako, ezinbestekoa da objektu errealak eta birtualak zuzen lerrokatzea, EAren eraginkortasuna handiagotzen duena. Arazo honen konponbidea *tracking* izenez ezagutzen da. Objektu baten tracking-a kameraren posizioa eta orientazioa edozein unetan zehaztean datza. Sentsore optikoak dira gehien erabiltzen direnak lerrokatze-arazoa konpontzeko; batez ere, bere inplementazio-kostu baxuagatik. Hala ere, ez da beti posible aldi berean irtenbide eraginkorra, zehatza eta konputazio-kostu baxukoa garatzea.

Tesi honek tracking optikoko teknika nagusien hobekuntza eta garapena jorratzen ditu, objektuen deformazioen hautematera bideratutakoak, nagusiki. Lehenengo eta behin, gainazal lau zurrunen eta deformagarrien tracking-a kudeatu da kamera monokularren bidez, eta gero objektuen deformazioak estimatu dira gailu konplexuago baten bitartez, RGB-D kamera batekin.

Arazo bat dute markagailuetan oinarritutako gainazal zurruneko tracking-etako sistemek: ezin dute oklusiorik jasan. Horregatik, oklusioen kontra duen markagailu pertsonalizatua eskaintzen du tesi honek; halaber, deformazioak kontrolatzeko, hala kameraren posizioa nola gainazalaren deformazioa aldi berean kalkulatzeko duen irtenbidea proposatzen du deformazioak kontrolatzeko. Deformazio-hautematearekin jarraituz, tracking eraginkorra ere garatu da objektu deformagarri baten 3D egitura berreraikitzeke bi formulazio fisiko desberdinen bidez. Batak portaera fisiko zuzena eta konputazio-kostu baxua eskaintzen ditu; bestea, ordea, zehaztasun maila handiagoetara heltzen da prozesatze handiagoren truke.

Gainera, faktore komuna dute deskribatutako irtenbide guztiek: denbora errealean burutzen dira, ER aplikazio bateko ikus-erantzuna arina izateko ezaugarri gakoa delako.

Contents

List of Figures	xxiii
List of Tables	xxvi
List of Nomenclature	xxvii
I Introduction	1
1 Introduction	3
1.1 Augmented Reality	3
1.1.1 Applications	6
1.2 Motivation	13
1.3 Contributions	16
1.4 Dissertation Organization	18
2 Background	19
2.1 Camera Geometry	20
2.1.1 Intrinsic Parameters	21
2.1.1.1 Camera Calibration	25
2.1.2 Extrinsic Parameters	27
2.2 Visual Cues	28
2.2.1 Features	28
2.2.1.1 Detection	29
2.2.1.2 Description	32

2.2.1.3	Motion	39
2.2.1.4	Matching	41
2.2.1.5	Outlier Removal	46
2.3	Camera Tracking	50
2.3.1	Stereo System	51
2.3.2	RGB-D System	53
2.3.3	Monocular System	56
2.3.3.1	Rigid Objects	56
2.3.3.2	Deformable Objects	66
2.4	Discussion	80
II	Proposal	83
3	Rigid Surface Marker Tracking	85
3.1	Introduction	85
3.1.1	ARToolKitPlus	87
3.1.2	Marker Occlusion	90
3.1.2.1	Previous Works	91
3.2	Proposed Method	94
3.2.1	Justification of the New Design	94
3.2.2	Algorithm Overview	96
3.2.3	Offline Phase	98
3.2.3.1	Keyframe Selection	99
3.2.3.2	3D Point Cloud Generation	99
3.2.3.3	Descriptors Generation and Database Creation	100
3.2.4	Online Phase	100
3.2.4.1	Frame-to-Frame Tracking	101
3.2.4.2	Tracking-by-Detection	104
3.2.5	New Interface Possibilities	106
3.2.6	Experiments	108
3.2.6.1	Tracking-by-Detection	108
3.2.6.2	Frame-to-Frame Tracking	113

3.2.6.3	Occlusion Pipeline	115
3.3	Discussion	118
4	Non-Rigid Surface Tracking	119
4.1	Introduction	119
4.2	Previous Works	121
4.3	Overview	122
4.4	Simultaneous Pose and Non-Rigid Surface Recovery	123
4.4.1	Offline Phase	123
4.4.1.1	Appearance Database	123
4.4.1.2	Deformation Database	124
4.4.2	Online Phase	129
4.4.2.1	Initialization	129
4.4.2.2	Frame-to-Frame Tracking	131
4.5	Experiments and Results	138
4.5.0.3	Parametrization	138
4.5.1	Experiments	140
4.5.1.1	Robustness	140
4.5.1.2	Visual Quality	144
4.5.1.3	Performance	146
4.6	Discussion	147
5	Deformable Object Tracking	151
5.1	Introduction	152
5.2	Previous Works	154
5.3	Overview	157
5.4	Proposed Method using a Mass-Spring Model	158
5.4.1	Tracking	160
5.4.1.1	Template Generation	161
5.4.1.2	Online Execution	162
5.4.2	Object Deformation	164
5.4.2.1	Model Preprocessing	164

5.4.2.2	Mesh Registration	169
5.4.2.3	Mesh Physical Simulation	174
5.4.2.4	Model Fitting	175
5.4.3	Experiments	176
5.4.3.1	Performance	177
5.4.3.2	Robustness	181
5.4.3.3	Adaptability	183
5.5	Proposed Method using a Finite Element Method	185
5.5.1	Tracking	185
5.5.2	Object Deformation	186
5.5.2.1	Model Preprocessing	188
5.5.2.2	Mesh Registration	190
5.5.3	Experiments	191
5.5.3.1	Performance	192
5.5.3.2	Adaptability	194
5.6	MSM versus FEM	196
5.6.1	Accuracy Level	196
5.6.2	Computation Time	197
5.7	Surgical Simulation System	199
5.7.1	Previous Works	200
5.7.2	Justification	201
5.7.3	Proposed System	202
5.7.3.1	Robotic Platform	203
5.7.4	Experiments	204
5.7.4.1	Accuracy Level	205
5.7.4.2	Abaqus	205
5.7.4.3	Scanner	206
5.7.4.4	Computation Time	210
5.8	Discussion	211

CONTENTS	xvii
III Conclusions	213
6 Conclusions and future work	215
6.1 Conclusions	215
6.2 Future Research Lines	219
IV Appendices	221
A Homography	223
A.1 Definition	223
A.2 The Direct Linear Transformation (DLT)	225
A.3 Pose Estimation from a 3D Plane	226
B Resolution of Kinect depth data	227
B.1 Depth Resolution	227
B.2 Depth Sensor Comparison	230
C Design of Experiments	231
C.1 Description of the Methodology	231
C.2 Case Study	236
D AR Framework links	245
E Generated Publications	247
References	255

List of Figures

1.1	Augmented Reality	4
1.2	History of AR	5
1.3	Reality-Virtuality Continuum	5
1.4	Assistant tools for industrial maintenance	7
1.5	AR Medicine	9
1.6	AR Games	10
1.7	AR Films	10
1.8	AR Marketing	11
1.9	AR Design	12
1.10	AR Glasses	12
2.1	Pinhole camera	20
2.2	Pinhole camera geometry	21
2.3	Perspective projection model	22
2.4	Skewed pixels distortion	24
2.5	Radial distortion	25
2.6	Calibration patterns	26
2.7	Euclidean transformation	27
2.8	Different types of visual cues	29
2.9	FAST feature detection	31
2.10	Descriptor extraction	33
2.11	SIFT scale space	35
2.12	DoG feature detection	36

2.13	SIFT steps	37
2.14	FREAK descriptor	38
2.15	FREAK pairs	39
2.16	Optical flow	41
2.17	Matching	42
2.18	K -NN	43
2.19	K -d Tree	44
2.20	LSH	45
2.21	RANSAC matching	48
2.22	Optical tracking classification	51
2.23	Depth image	52
2.24	Two camera geometry	52
2.25	Kinect sensor	54
2.26	Kinect depth image	55
2.27	Marker tracking system overview	57
2.28	2D displacements of control points	59
2.29	Tracking using a particle filter	61
2.30	3D object recognition based on appearance	62
2.31	3D scene reconstruction	64
2.32	Physics-based models	67
2.33	Learning-based models	69
2.34	Morphable face models	70
2.35	Database of feasible shapes	71
2.36	Shape recovery	73
2.37	Reconstruction of 3D surface	75
2.38	Time constraint	76
2.39	Distance constraint	76
2.40	NRSfM	79
3.1	Different types of markers	87
3.2	ARToolKitPlus pipeline	88
3.3	Perspective correction of a BCH marker	90

3.4	Segmentation for different marker occlusions	91
3.5	Output response for ARToolkitPlus and Occlusion module	95
3.6	Evolution of the new marker design	96
3.7	Occlusion pipeline overview	97
3.8	Offline phase of Occlusion pipeline	98
3.9	FtF initialization	102
3.10	GREY descriptor	102
3.11	Refinement of the FtF	103
3.12	Overview of the TbD	105
3.13	An example of Occlusion Signal output	107
3.14	Photo viewer sequence	108
3.15	Marker designs	109
3.16	Simulation of the partial occlusion of the marker	110
3.17	Scale study for TbD	110
3.18	Simplified-SIFT parametrization study for TbD	112
3.19	Camera pose using the FtF	114
3.20	Occlusion-patches response	116
3.21	Occlusion-patches output	117
4.1	Ambiguities	120
4.2	Appearance database	124
4.3	Mesh representation	125
4.4	Deformation database	126
4.5	Deformation profile	128
4.6	Online phase	130
4.7	Initialization	131
4.8	Particle filter	135
4.9	PnP	136
4.10	<i>Guernica</i> factors behaviour	141
4.11	<i>Stones</i> factors behaviour	142
4.12	Noise test	143
4.13	Reprojection error	144

4.14	Results of 3D recovering surfaces	145
4.15	Clothes recovery	145
4.16	False positive results	147
4.17	Different points of view	149
5.1	Mesh adquisition	158
5.2	Raw information	159
5.3	Overview MSM	160
5.4	Template learning	161
5.5	LINEMOD database	162
5.6	Tracking online execution	163
5.7	ICP	163
5.8	Preprocessing	165
5.9	Voxelization	167
5.10	Voxel classification	168
5.11	Object deformation	170
5.12	Occlusion query	171
5.13	Detection	172
5.14	Correspondence matching MSM	173
5.15	Voxel vertex displacement	174
5.16	Isoparametric coordinates	176
5.17	MSM execution times	178
5.18	Execution times of the physical integration step	178
5.19	Different points of view	180
5.20	Transformations robustness	181
5.21	Noise	182
5.22	Visual MSM results	184
5.23	Models FEM	186
5.24	Overview FEM	187
5.25	Rheometer	188
5.26	Correspondence matching FEM	191
5.27	Visual accuracy level of FEM	193

5.28	FEM execution times	194
5.29	Visual FEM results	195
5.30	Comparison accuracy level MSM with FEM	198
5.31	Surgical models	202
5.32	Surgical simulation system	203
5.33	Representation of the robotic arm	204
5.34	Abaqus comparison with FEM	207
5.35	Scanner comparison with FEM	208
5.36	Visual feedback of surgical object deformation	209
5.37	Execution times of the surgical simulation system	210
A.1	The mapping of points between two planes	224
B.1	IR geometric model	228
B.2	Kinect data	229
C.1	DoE methodology	232
C.2	Pareto charts <i>Stones</i>	239
C.3	Pareto charts <i>Guernica</i>	240
C.4	Normal plots <i>Stones</i>	241
C.5	Normal plots <i>Guernica</i>	242
C.6	Optimization	243

List of Tables

2.1	Differences between SIFT and FREAK descriptors.	34
2.2	Advantages and disadvantages of different types of models	81
4.1	Textures for DoE	139
4.2	ANOVA	140
4.3	Execution times	146
5.1	Execution times (in <i>ms</i>) for the tracking module.	177
5.2	Error for different number of iterations	179
5.3	Error for different types of models	180
5.4	Mechanical properties of the tested materials	189
5.5	Error estimation	192
5.6	Error comparison between FEM and MSM formulations	196
5.7	Computation time comparison between FEM and MSM formulations	197
5.8	Mechanical properties of the tested materials for the surgical system	205
5.9	Error estimation FEM with Abaqus	206
5.10	Error estimation FEM with Scanner	207
B.1	Different depth sensors	230
C.1	Primary factors	236
C.2	Region of operation	237
C.3	Region of interest	237

C.4	Region of experimentation	238
C.5	ANOVA	238
D.1	AR SDKs.	245

List of Nomenclature

Acronyms and Definitions

AABB	Axis Aligned Bounding Box
ANOVA	ANalysis Of Variance
AR	Augmented Reality
BA	Bundle Adjustment
BCH	Bose, Chaudhuri, Hockquenhem
BF	Brute Force
BSP	Binary Space Partitioning
CCD	Charged Coupled Device
CEIT	Centro de Estudios e Investigaciones Técnicas
CMOS	Complementary Metal-Oxide-Semiconductor
CAD	Computer-Aided Design
CS	Consensus Set
CV	Computer Vision
CoP	Centre of Projection
DLT	Direct Linear Transformation
DoE	Design of Experiments
DoF	Degrees of Freedom
DoG	Difference of Gaussian
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
FEM	Finite Element Method
FoV	Field of View
FREAK	Fast Retina Keypoint
FtF	Frame-to-Frame tracking
GPL	General Public License
GPLVM	Gaussian Process Latent Variable Model

GPU	Graphics Processing Unit
HMD	Head-Mounted Display
HSV	Hue-Saturation-Value
ICP	Iterative Closest Point
IR	Infrared
KF	Kalman Filter
LLAH	Locally Likely Arrangement Hashing
LSH	Locality Sensitive Hashing
MR	Mixed Reality
MSM	Mass-Spring Model
MSS	Minimal Sample Set
NCC	Normalized Cross-Correlation
NRSfM	Non-Rigid Structure from Motion
NURBS	Non-Uniform Rational B-Spline
KPCA	Kernel-PCA
K-NN	K-Nearest Neighbour
OBB	Oriented Bounding Box
OHMD	Optical Head-Mounted Display
PCA	Principal Component Analysis
PCL	Point Cloud Library
PF	Particle Filter
PnP	Perspective-n-Point
PROSAC	PROgressive SAmple Consensus
RANSAC	RANdom Sample and Consensus
RGB-D	Red Green Blue-Depth
SDK	Software Development Kit
SfM	Structure from Motion
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SMC	Sequential Monte Carlo
SOCP	Second Order Cone Programming
SVD	Singular Value Decomposition
TbD	Tracking-by-Detection

List of Symbols

d	Euclidean distance	K	Stiffness matrix
D	Damping matrix	L	Length of spring
Δt	Integration time step	\vec{m}	2D image coordinates
e^2	Mean squared error	\vec{M}	3D coordinates
E	Young's modulus	M	Mass matrix
ϵ	Global energy	N	Interpolation function
ϵ_D	Internal energy	n	Vector size
ϵ_C	External energy	ν	Poisson's ratio
f	Focal length	P	Camera projection matrix
f	Elastic force	P_N	Normalized P
G	Gaussian kernels	R_t	Extrinsic parameters matrix
G	Gaussian weight	ρ	Density
G	Shear modulus	s	Scale factor
H	Homography transformation	T	Template
i	Regions (SIFT)	v	Deformed mesh
l	Intensity	w	Canonical mesh
j	Histogram bins (SIFT)	w	Particle weight
k	Patch size (SIFT)	(X, Y, Z)	3D world coordinate
k	Stiffness value	(x, y)	2D image coordinate
K	Intrinsic parameters matrix		

Part I

Introduction

Chapter 1

Introduction

*Ez dago protokolorik,
baldintza bakarra dago,
disfrutatu eta sentitu.*
GORKA URBIZU

1.1 Augmented Reality

Augmented Reality (AR) is a technology that aims to embed virtual objects in the real world, showing the user the set of objects (virtual and real) as a single world. In contrast to *Virtual Reality (VR)*, where the user is completely immersed in a synthetic world, AR consists in adding virtual objects to the real world (see Figure 1.1). An AR system is composed of a camera that captures the images from the scene, a data processing unit to analyse them and a display device in order to combine real and virtual elements.

The first work of AR is attributed to Ivan Sutherland (Sutherland, 1968). This work, which is also the first VR system, uses an *Optical see-through Head-Mounted Display (OHMD)* where the position is measured by mechanical and ultrasonic sensors (see Figure 1.2(a)). However, it was not until the beginning of the 1990s when in (Caudell and Mizell, 1992) Tom Caudell and David Mizell coin the term of *Augmented Reality*. The authors describe the implementation of a heads-up (see-through) head-mounted display (called *HUDset*) to guide workers to improve the efficiency and quality in their performance of manufacturing or assembly operations (see Figure 1.2(b)). Furthermore, they refer to this technology as *Augmented Reality* since it is



Figure 1.1: Illustration of AR. Virtual skeleton superimposed on the body of a human (*by Ben Heine*).

used to augment the visual field of the user with the necessary information for the performance.

Later in 1994, Paul Milgram and Fumio define the concept of a *Virtuality Continuum* (Milgram and Kishino, 1994) that serves to describe a continuum scale that goes from real to virtual environments. This scale distinguishes between VR and AR, depending on the synthetic information displayed in the scene. VR would be at the right side of the scale; while on the left would be the reality, as shown in Figure 1.3. Thus, a new term appears, called *Mixed Reality (MR)*, which is a particular subset of VR which involves the merging of real and virtual worlds. The AR therefore, refers to all cases where the display of real environments is augmented through virtual objects.

(Azuma et al., 1997) depicts the first survey on AR and underlines the main three requirements for any AR system.

1. Combine real and virtual objects.
2. Interactive real-time executions.
3. Correct 3D registration, overlaying the virtual information with its corresponding real counterpart.

Taking into account the above-mentioned points, the main challenge for an AR application is to give the illusion of seeing both realities; virtual and real,

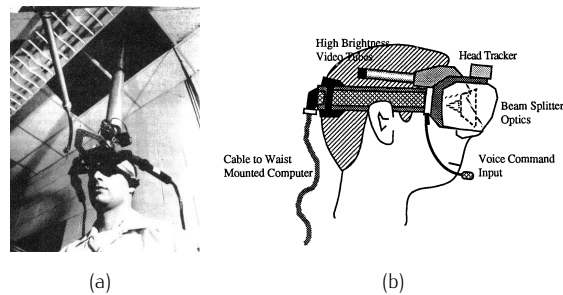


Figure 1.2: Head tracking sensors used by (Sutherland, 1968) (a) and a head-mounted system for AR manufacturing activities in (Caudell and Mizell, 1992) (b).

as one. Therefore, it is necessary to achieve a robust and accurate *registration*, which consists in finding a perfect alignment between real and virtual objects. To overcome the registration problem, the position and orientation of the observer (known as *camera pose*) has to be determined (the calculation of the camera pose is also known as *tracking*). Using this information, a virtual camera can be configured, which indicates exactly where the virtual objects should be drawn in the image. It requires the extraction of the 6 *Degrees of Freedom (DoF)* that represent the user's point of view: 3DoF for orientation and 3DoF for translation.

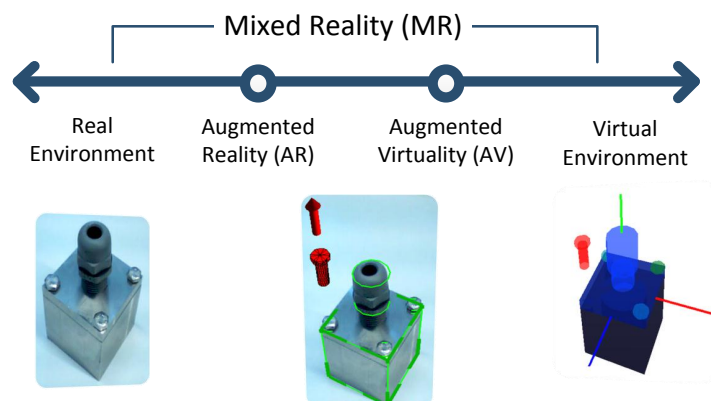


Figure 1.3: Reality-Virtuality Continuum (Milgram and Kishino, 1994).

There are many alternatives that differ from the type of sensor used to address the registration problem. (Rolland et al., 2001) offers the following

classification of tracking technologies that are necessary to record the position and orientation of real objects in physical space:

- **Inertial sensors.** Combine accelerometers and gyroscopes to estimate the translations and rotations respectively.
- **Ultrasound sensors.** Rely on the delay times of ultrasonic pulses to deduce the position and orientation of the camera.
- **GPS receivers.** Use the signals emitted by a set of satellites to triangulate its pose.
- **Magnetic sensors.** Measure the magnetic fields to deduce the viewpoint parameters.
- **Optical sensors.** Process the image of the scene captured by a camera to obtain its corresponding 6DoF.

This thesis has been focused on those solutions based on optical sensors.

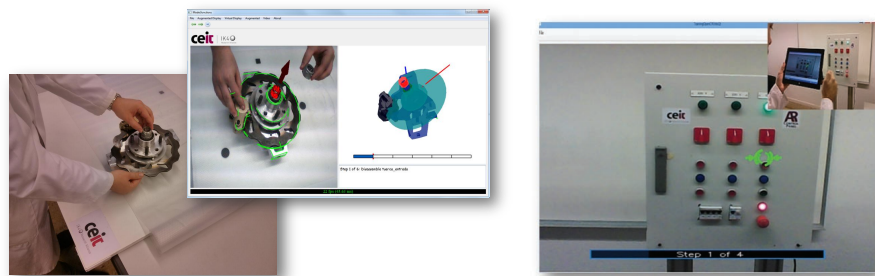
1.1.1 Applications

AR technology can be used, beneficially, in several application areas. (Azuma et al., 1997) classified the environments where this technology is applicable. It showed AR applications in at least six areas: medical visualization, maintenance and repair, annotation, robot path planning, entertainment, and military aircraft navigation and targeting. However, due to advances in new technologies, today AR has evolved and has spread to many fields that previously were limited to certain industrial applications and managed by few research groups. The following presents examples of how visual AR has been used in some of the areas cited by Azuma and many others.

Maintenance

The prototype called *KARMA (Knowledge-based Augmented Reality for Maintenance Assistance)* is considered one of the first important practical applications of an AR system. *KARMA* (Feiner et al., 1993) consists of a prototype system where the user uses a *Head-Mounted Display (HMD)* to explain simple end-user maintenance of a laser printer. This application was

widely quoted within the science community and it was also one of the first approaches to include AR technology in *maintenance* and *assembly tasks*. These types of assistance applications for industrial maintenance provide assistance and traceability tools in assembly/disassembly tasks of complex machinery using virtual annotations. They detect and recognize the desired object to perform maintenance and from there, a set of instructions are given to the user (see Figure 1.4). The system can sometimes even detect whether a step is incorrectly being carried out, and a visual alert is given. Likewise, these techniques can offer an automatic report generation or a pictographic documentation of a workflow (Petersen and Stricker, 2012).



(a) AR assembly and disassembly.

(b) AR assistance tool for a control panel.



(c) AR projector tool for industrial maintenance assistance.

Figure 1.4: Assistance and traceability tools in execution of procedures.

This way, following the strategy of *KARMA*, numerous applications have been developed in the field of maintenance and assembly. Examples include

projects such as (Wohlgemuth and Triebfürst, 2000) or (Schwald et al., 2001), mainly dedicated to assembly tasks. The *ARMAR* project (Henderson and Feiner, 2009; Henderson and Feiner, 2011a; Henderson and Feiner, 2011b) also explores the use of AR in the execution of procedural tasks in the field of maintenance and repair, as well as the recent project *AR-Mentor* (Zhu et al., 2014). Furthermore, maintenance and assembly applications also differ in the tracking method used. *Marker-based tracking methods* (Savioja et al., 2007; Salonen et al., 2007; Sääski et al., 2008; Salonen et al., 2009) provide a high level accuracy and they do not usually require very powerful hardware such as mobile devices (Siltanen et al., 2007; Savioja et al., 2007; Hakkarainen et al., 2008). However, the main disadvantage of this type of tracking is that they require the environment to be prepared in advance with artificial landmarks, requisite which is not always possible in an industrial field. Likewise, there are other approaches which are focused on *model-based tracking*, which rely on the prior knowledge of the 3D models of the environment (Caponio et al., 2011; Álvarez et al., 2011; Neubert et al., 2012; Meden et al., 2014).

Medicine

Medicine is another field where AR plays an increasingly important role. This is supported by the fact that an operation plan can be developed to reduce the workload of surgeons during an intervention or perform a smaller number of incisions (making the recovery phase faster and less painful). One of the most premature contributions in this field (Ohbuchi et al., 1998) generates a 3D representation of the foetus inside the womb in order to allow the doctor see it. Following a similar criteria, this practice could be extrapolated to other kind of surgeries to see inside the patient. Nevertheless, there is one important thing that has to be taken into account: accuracy is the main requirement for these applications, something that does not happen in other fields like entertainment. Approaches such as (Kim et al., 2012) propose a robust solution to track deformable organs but do not achieve the necessary accuracy level. In this regard, there are already methods like (Haouchine et al., 2013b; Haouchine et al., 2013a) that achieve a high level of accuracy. In these cases, they believe in a biomechanical model-based simulation for minimally invasive hepatic surgery. The obtained data from the pre-operative, such as vascular network tumours and cut planes, can be overlaid onto the laparoscopic view for image-guidance (see Figure 1.5). Furthermore, biomechanical-based deformable models have been demonstrated to be relevant since they allow defining elastic properties of the

shape, including anatomical information and deformation constraint (Pratt et al., 2010; Speidel et al., 2011). Similarly, AR technology has enabled research in many medicine sectors, like performing an image-guided tumour identification (Hamarneh et al., 2014), surgical oncology (Nicolau et al., 2011) or coronary surgery (Figl et al., 2010).

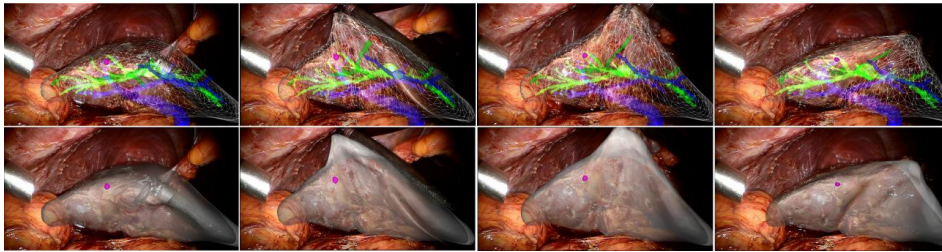


Figure 1.5: Biomechanical model onto the human liver during Minimally Invasive Surgery (by (Haouchine et al., 2014b)).

Guidance

If there is a field where the concept of AR has been exploited, this is *entertainment* (entertainment being defined as everything related to marketing, advertisement, film or game industry). The advance of new technologies, paying particular attention to smartphones, opens new possibilities in this field. This is the case of location-based AR services that are very popular on mobile platforms. Applications like *Layar*, *Wikitude*© or *Metaio*© use the camera and different sensors of the cellular (gyroscopes or GPS). This gives additional information of the environment of the user and together with geolocated online database it is possible to provide the user information related to the place the user is visiting. Hence, (Rao et al., 2014) takes advantage of sensors available in cars, including the GPS, the steering sensor, the wheel odometer and the inertial measurement unit, in order to develop an *AR In-Vehicle Infotainment (AR-IVI)* system that is installed in the *Mercedes-Benz*© *R-Class* car.

Game and Film Industry

The *game industry* has also included the AR technology, letting the user be part of the game. AR enables interaction between the user and the environment. Based on the *ARToolKit* tracking system (created by Hirokazu Kato and

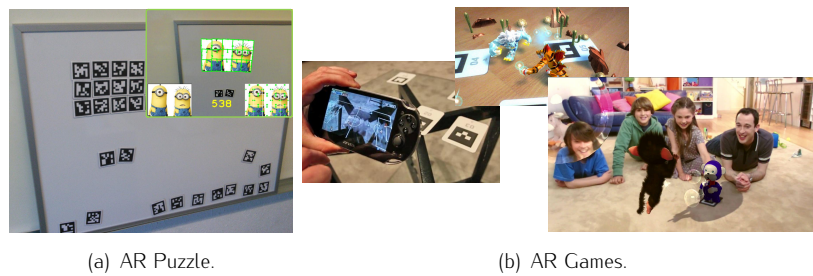


Figure 1.6: Examples of AR games.

Mark Billinghurst (Kato and Billinghurst, 1999) and available as open source) numerous games have appeared in the market. To mention some examples, *Invizimals™*, *PS® Vita AR Cards*, *Eye Pet™* for Nintendo DS™, *ARQuake* or *AR Invaders* (see Figure 1.6) can be highlighted. And eventually, the introduction of AR into *film industry* has seduced some producer companies to try these tools, since integrating real actors in several virtual backgrounds leads cost savings. Some of the most relevant films are *Terminator*, *Robocop*, *Minority Report* or *Iron Man* (see Figure 1.7).



Figure 1.7: The AR has also been incorporated into the film industry.

Marketing

Following the same lines, the *marketing* and *sales* fields have also taken AR on board. AR is a tool that is well-associated for these claims. The most notable cases are in automotive, dressing or fashion accessories companies. *Mini*, for example, presents a very detailed 3D model of its *Cabrio*© on a magazine's page (see Figure 1.8(a)). On the other hand, the watch brands like *Tissot*® or *Dezeen Watch Store*© proposed campaigns letting users wear virtual watches before buying them (see Figure 1.8(b)). There are other fashion accessories companies that have used this kind of digital mirrors called *AR Shops* where the user can try out different face cosmetic products, jewellery or dresses (see Figure 1.8(c)).

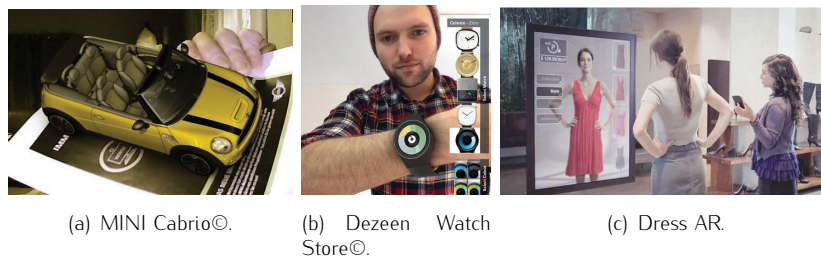


Figure 1.8: Examples of AR applications for marketing and advertisement.

Design

AR applications can also be used for the on-site visualization of both indoors and outdoors, trying to imagine exactly how the furniture will look in a room of a house (Siltanen et al., 2014) or a building in a certain location of the city. In *interior design* for example, it is possible to find the new *IKEA AR Catalogue*© to visualize how furniture could look inside the room (see Figure 1.9(a)). *Sayduck*© also released a similar mobile app that helps users see certain products in their homes (see Figure 1.9(b)). In the case of outdoor design, existing solutions can visualize building projects using AR with a camera (Kähkönen et al., 2007; Schattel et al., 2014).

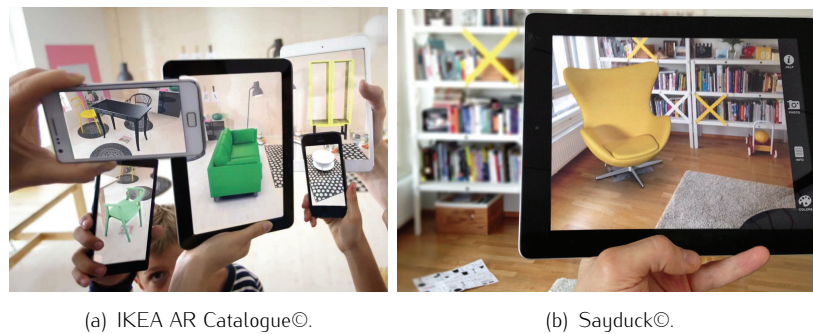


Figure 1.9: AR helps the work of interior designing.

Some of these devices are not able to provide the immersion experience desired by many users due to the inconvenient of their use. However, this scenario could change with the arrival of new commercial solutions: AR glasses (see Figure 1.10). Products like *Google Glass* or *Vuzix Smart Glasses*© allow the user achieve a much more realistic experience by keeping information in sight. Moreover, these eyeglasses display information and allow the user to communicate with a smartphone efficiently, merely through voice, so there is no need to check the screen of the phone. This is a great step forward in fields like industry, since sometimes is difficult to handle a mobile device (smartphones, tablets...) while the operator is working on a specific task where both hands are required.



(a) Google Glass©.



(b) Vuzix Smart Glasses©.

Figure 1.10: Google (a) and Vuzix (b) glasses.

1.2 Motivation

Versatility of AR

Unlike VR, AR provides enhanced reality, since it interacts with real objects. Furthermore, VR is mainly oriented to training, with the aim of improving the skills of the user through multiple simulations, while AR can be used for both training and guiding.

AR as a potential tool in medicine

If there is an area where this type of assistance tools can be applied, that would undoubtedly be medicine. It can develop, for example, a system that combines basic functionalities of medical imaging and real simulations in order to guide/help in every moment. Moreover, it involves numerous advantages:

- Reduce fatigue, resulting in a greater precision by the surgeon. Assistance tools lessen the mental workload of the users, creating greater accuracy and allowing more fluent decisions.
- Reduce the error rate. The alerts provided by the system serve as a guide in order to make corrections for possible mistakes and reduce the error rate.
- Costs reduction. Replacing the work of assistants by robotic, electronic or computer equipment.
- Automatic generation of documentation. Automatic data recording of concerning details in an operation. These data can serve as material for teaching and training as well as for complementing the traditional manuals.

Low cost and non-invasive optical tracking solution

In this respect, tracking based on optical sensors (also called *visual tracking*) is the most popular solution because it is inexpensive and does not require a significant adaptation of the environment. It does not require bulky machines to be added to the scene or force the user to carry heavy devices; it only uses a

camera to capture images of the scene, a computer to process the image, and a screen to overlay the virtual information.

Finding a robust marker tracking system

In order to obtain an accurate and robust tracking system based on *Computer Vision (CV)*, markerless solutions use natural features or a 3D model to compute the camera pose. These alternatives do not require environment adaptation, because they rely on natural features that are on the scene or lie on the surface of the model to be tracked. Nevertheless, in some cases, a rich texture scene is required, whereas in other cases, the 3D model must be known, something not always easy to obtain. Marker tracking systems are a real alternative to those problems, because they add special patterns to the scene. Furthermore, they are faster, more accurate and more robust than markerless tracking solutions. The main drawback is the environment adaptation, although in some cases, this adaptation is feasible without incurring any extra work. Besides the environment adaptation, marker occlusion is another shortcoming, as the system fails even if the marker is only slightly occluded. This produces an undesirable effect on users who lose the sense of realism. This is a clear limitation that must be dealt with.

Shortage solution in the relatively unexplored deformable tracking field

On the other hand, most of the research on AR has focused on tracking systems of rigid objects. Recognizing these objects in the image and register with virtual models are classic problems in AR, numerically delicate but with numerous approaches in the literature. However, in certain applications it is very likely to recognize and track deformable objects, adding an extra complexity to the problem. In the particular case of medical applications for example, it is very likely to track deformable objects (skin, tissues, organs...). Thus, recovering the shape of 3D non-rigid objects from monocular video sequences is an ill-posed problem because many 3D surfaces or objects can have the same 2D projection. Even having the intrinsic camera parameters and a well-texturized surface, it is difficult to model all the possible configurations of a deformable object in order to overcome the depth ambiguities. In (CV), approaches on AR oriented to deformable objects are still very limited in the literature and have been explored by a few research groups. Moreover, the number of approaches decreases when

a real-time constraint is set. This is mainly caused by the complexity of the recognition and non-rigid registrations steps.

Real time key to provide correct visual feedback

Finally, one of the keys of AR is how realistically augmentations can be integrated in the real world. This realism is not only achieved by a proper alignment between real objects and their virtual counterparts, but it is necessary to take other features into account. Among these can be included the occlusion computations or the estimation of real illumination to integrate virtual objects with real ones. Furthermore, to increase the level of realism it is necessary to return a fluent feedback essential for user interaction, which is achieved by applications running in real time.

1.3 Contributions

The use of AR enables the projection of virtual information that sometimes is even impossible for the naked eye, for example in the medicine field (information of a patient). Thus, this thesis falls within this context. The present work aims to study and improve the main AR tracking techniques, especially those oriented to handle deformable objects, very common in medical environments. It consists in a tour through the different ways that exist to perform an object tracking (rigid and non-rigid). Among the different types of tracking, this work deals with the tracking of markers, the tracking of 3D deformable surfaces and finally the tracking of 3D deformable objects. The aim of the first tracking solution, markers, focused primarily on acquiring the knowledge required in optical tracking methods through efficient and simple systems such as marker tracking systems. This enables us to tackle the non-rigid issue in a more efficient way. It has also made an improvement to these techniques, successfully achieving a contribution.

All these tracking methods achieve real-time executions. By real time we mean 25-30 frames per second, enough to get stability and continuity in the image flow. A brief descriptor of each of these three research developments of the thesis are detailed below.

1. *Rigid surface tracking (Marker)*

A robust real-time marker tracking system that supports partial occlusions through a new marker design which lets users create their own markers (marketing opportunities) and implement human-machine interfaces.

This proposal relies on the widely used ARToolKitPlus (Wagner and Schmalstieg, 2007) non-commercial marker tracking library. This library does not take advantage of their frame to codify information. By placing textured patches along the frame of the marker there are more visible features (with known 3D coordinates). Then, during marker occlusion it is possible to update the 6DoF of the camera in real time offering a higher robustness. Moreover, the presented design does not change the functionality of ARToolKitPlus since it is backward compatible with it. Nevertheless, the method can be adapted to any marker that does not codify information on its frame. This method has been developed to be public access available and it can be downloaded.

2. *Non-rigid surface tracking*

A novel solution recovers the camera pose and the non-rigid 3D surface simultaneously along a video sequence in real time.

The main idea of the proposed method is to use an efficient *Particle Filter (PF)* which performs an intelligent search in a database where a range of deformation templates and appearance descriptors are stored in order to achieve a real-time performance. Furthermore, two methods are combined to offer robust tracking. The first is based on appearance, which serves to do both the initialization and the recovery from failure, while the second is based on temporal coherence.

This solution is oriented to applications of cloth simulation or the tracking of sheets of papers.

3. *Deformable object tracking*

A robust tracking method that is able to track in real time the camera pose and to reconstruct the 3D structure of a deformable object by applying a set of physical constraints to obtain a realistic behaviour using a RGB-D camera.

The RGB-D camera information usually has too much noise with large incomplete areas which causes incorrect visual feedback. Accordingly, through a physics-based method, the mesh is adjusted to the raw information obtained from the camera, providing a realistic behaviour in order to calculate the deformations regardless of their geometric shape.

Additionally, detection is not based on features, and therefore it can operate with textured or untextured objects. Moreover, this method offers accurate visual feedback, versatility (the ability to work with different types of objects) and real-time performance over accuracy.

This kind of techniques can result in guiding on surgery simulations, tracking the organs of the patient.

1.4 Dissertation Organization

The dissertation is divided into 6 chapters. Chapter 1 has presented the status of the AR technology and the fields where it has been implemented. In addition, the factors that have motivated this work and the main objectives to be achieved have also been presented. As this thesis is focused on the study of the main AR tracking techniques, Chapter 2 introduces some base information that will be useful for understanding the subsequent chapters. Furthermore, it gives a classification of different optical tracking approaches. Chapter 3 presents the monocular optical tracking based on markers, including a proposed solution to overcome occlusions. Chapter 4 describes the problem presented by non-rigid 3D surface tracking and provides all the steps to be carried out in order to overcome the problem. Afterwards, a *Design of Experiments (DoE)* is presented to determine the optimal parameters values of the solution. Chapter 5 deals with the problem of recovering a non-rigid 3D object and deducing the camera pose that best fits in the projection. Finally, conclusions and future works are enumerated in Chapter 6.

Some appendices also appear at the end of this document in order to explain some technical concepts in more detail.

Chapter 2

Background

*Bakoitzak urraturik berea
denon artean geurea
etengabe gabiltza zabaltzen
gizatasunari bidea.*

MIKEL LABOA

Some ideas introduced in this chapter can be found in:

Álvarez, H., Leizea, I., and Borro, D. "A survey on optical tracking for augmented reality". In Proceedings of the XXII Conferencia Española de Computación Gráfica (CEIG), pp. 31–40. Jaén, Spain. September, 2012.

The main goal of *Augmented Reality (AR)* is to integrate virtual objects into a real scene. This requires a perfect alignment between virtual and real objects in order to create the sensation that virtual objects belong to the real world. This problem known as *camera tracking* is solved by computing the position of the virtual camera. Thus, the major challenge of AR is to obtain a good tracking, which is solved by determining the position and orientation of the camera (*camera pose*).

In this way, *optical tracking* has been extensively studied by several authors, resulting in many different optical tracking methods. However, it is not a solved problem and it still remains as an important research topic because it is often difficult to provide robustness, accuracy and low computational cost at the same time. This chapter reviews the mathematical tools for understanding the camera model as well as the methods that have been designed to overcome the problem of tracking by using images from the camera as the sole source of information.

2.1 Camera Geometry

The light-sensitive sensors of the cameras are responsible for obtaining the image. The amount of light captured through the camera lens is processed by the so-called *Charged Coupled Device (CCD)* and *Complementary Metal-Oxide-Semiconductor (CMOS)* sensors. This light is converted to an electric charge at each pixel and depending on the amount of the charge; the intensity of each pixel is obtained, thus producing the final image as a result. The camera is in turn, a device that captures 3D information from the scene and projects it onto an image plane of two dimensions.

The *pinhole camera* is the most commonly used representation of the *perspective projection model* in *Computer Vision (CV)* to describe the whole mathematical procedure of a real camera. It is a simple camera without a lens and with a single point aperture (a pinhole). As shown in Figure 2.1, the rays of light from the scene pass through this hole to form an inverted 2D image of the 3D object. The image quality of a pinhole camera is directly related to the size of the hole: a small size produces a sharp image but it will be dimmer due to insufficient light, whereas a large hole generates brighter but more blurred images.

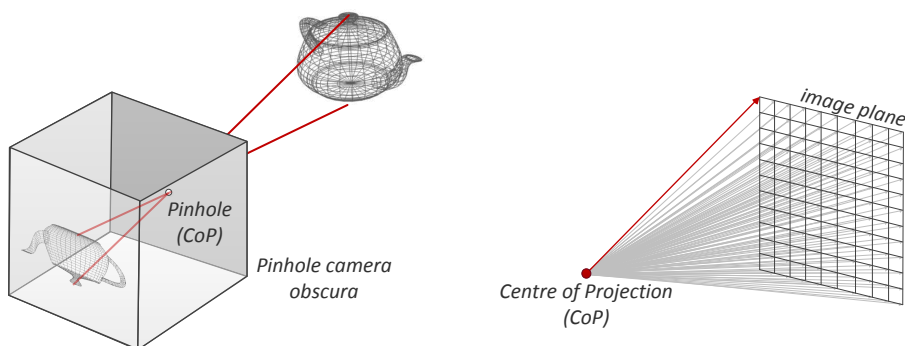


Figure 2.1: The light of the scene passes through a single point (the CoP) projecting an inverted image.

As with real cameras, there are certain technical aspects to be considered. Cameras are represented by two main groups of parameters: *intrinsic* and *extrinsic parameters*. The *intrinsic* parameters, also called *internal camera*

parameters, represent the internal characteristics of the camera, while the *extrinsic* parameters express the position and orientation of the camera.

2.1.1 Intrinsic Parameters

Let's describe the pinhole model as a central projection or *Centre of Projection (CoP)* C in space called *camera centre* and a plane located at a distance f from the CoP. This plane is called the *image plane* or *focal plane*. In order to avoid the image appearing inverted, an equivalent geometric configuration is used, where the optical centre is moved behind the image plane. For this reason, the image plane is located at $(0, 0, f)$, where f is a non-zero distance.

The projection generated by this model is the intersection between the image plane and the line joining the point to be projected to the CoP. Therefore, a 3D point in the world space (X, Y, Z) is mapped to a 2D point (x, y) on the image plane. A demonstration of the model can be shown in Figure 2.2.

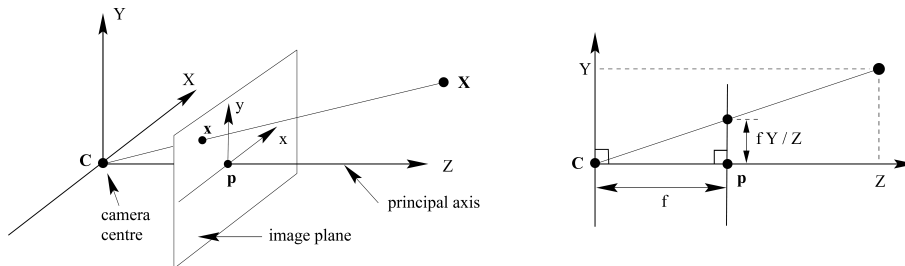


Figure 2.2: Pinhole camera model (Hartley and Zisserman, 2003).

Under the assumption that the camera and world coordinate systems are aligned, the plane XY through the camera centre is parallel to the image plane and is called the *principal plane*. Moreover, the Z axis coincides with the *optical axis* or *principal axis*, which is the axis passing through the CoP. The intersection between the principal axis and the image plane is called the *principal point* P .

Accordingly, due to the similar triangles rule, the projection of the 3D world coordinate (X, Y, Z) is mapped to the 2D image coordinate (x, y) on the image plane:

$$x = f \frac{X}{Z}, \quad y = f \frac{Y}{Z}. \quad (2.1)$$

When the focal length f is equal to 1, the camera is said to be normalized or centred on its canonical position. Therefore, this function can be defined in order to transform points from the Euclidean 3-space \mathbb{R}^3 to Euclidean 2-space \mathbb{R}^2 .

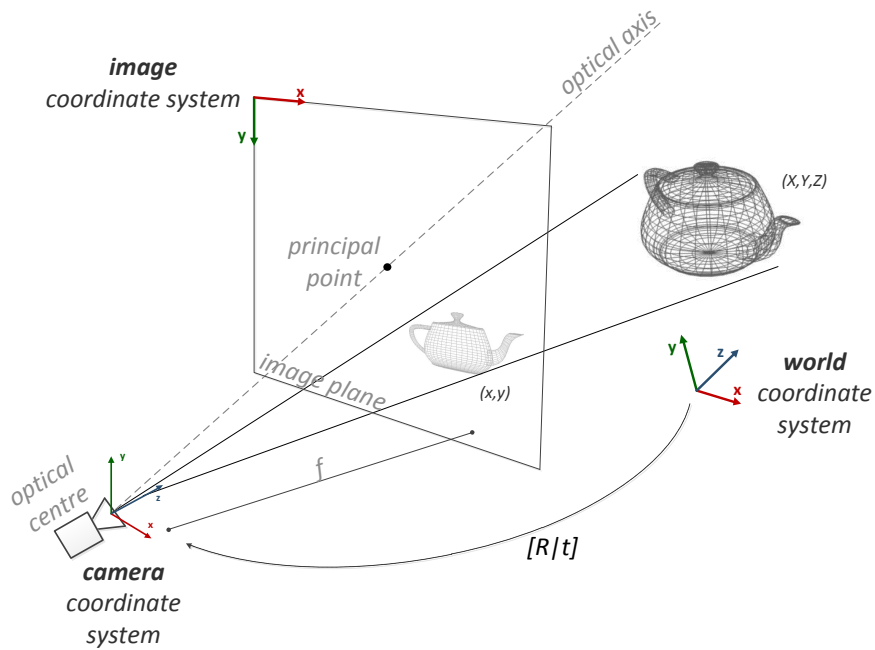


Figure 2.3: Perspective projection model.

a. Homogeneous coordinates

By representing the former mathematical expression under homogeneous coordinates, the process can be expressed as a linear system. Using this notation, Equation 2.1 can be expressed in terms of matrix multiplication, obtaining the following system:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} \sim \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (2.2)$$

Thus, the rays mapped onto the image plane can be uniformly described in a compact way by the classic 3x4 homogeneous *camera projection matrix* P :

$$x = PX = KP_N X, \quad (2.3)$$

where X denotes the world point represented by homogeneous 4-vector $(X, Y, Z, 1)$ and x the image point by homogeneous 3-vector $(x, y, 1)$. P_N is the projection matrix of the normalized camera and K is the calibration matrix or intrinsic parameter matrix, which describes the characteristics of the camera.

b. Principal-point offset

However, the Equation 2.2 models the behaviour of an ideal pinhole camera. So far, it is assumed that the origin of coordinates in the image plane is at the principal point. Nevertheless, in practice, the image coordinate system is centred on the top left of the image (see Figure 2.3), so the pixel coordinates of the principal point are not $(0, 0)$, but (p_x, p_y) :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} P_N \sim \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (2.4)$$

c. Image-sensor

Another issue is the implicit assumption that the pixels of the image sensor are square, i.e., the aspect ratio is 1:1 (see Figure 2.4). Nevertheless, there is also the possibility of having incorrectly positioned the camera sensor with respect to the lens, producing non-square pixels (see Figure 2.5). This has the extra effect of introducing unequal scale factors in each direction (s_x, s_y) , providing two different focal lengths, one for each axis $(f_x = f/s_x, f_y = f/s_y)$.

By introducing these parameters into the expression 2.4, the resultant equation is:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim KP_N \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2.5)$$

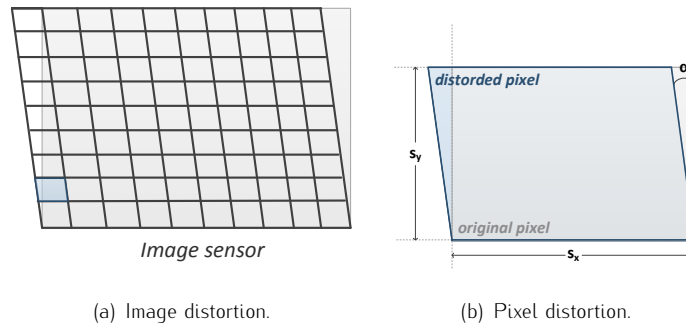


Figure 2.4: Image (a) and pixel (b) distortion. Image sensor with non-square (*skewed*) pixels. s_x and s_y are the dimension of the pixels and α is the degree of tilt.

where s , called *skew parameter*, measures the error in the alignment between the sensor and lens ($s = \tan \alpha * \frac{f}{s_y}$).

d. Distortions

Similarly, as depicted below, the world point, image point and optical centre are not collinear for real (non-pinhole) lenses. Thus, together with the effects cited, there are other important deviations called *distortions* that cannot be modelled linear. These effects can be classified as *tangential* and *radial distortions*. *Tangential distortion* is caused by physical elements in the lens that are not perfectly aligned. They are usually discarded. The *radial distortion* (see Figure 2.5), in turn, is a failure of the lens to be rectilinear (causes straight lines to appear as curves). The presence of the radial distortion is manifested in the form of the *barrel* or *fish-eye* effect. To carry out in the right place, the image can be warped using Brown's distortion model (Brown, 1966):

$$\hat{x} = x_c + L(r)(x - x_c), \quad \hat{y} = y_c + L(r)(y - y_c), \quad (2.6)$$

where (x, y) are the image points (\hat{x}, \hat{y}) are the corrected coordinates, (x_c, y_c) is the centre of radial distortion ($r^2 = (x - x_c)^2 + (y - y_c)^2$) and $L(r)$ is a distortion factor that can be expressed by a Taylor expansion $L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + \dots$. The coefficients for radial correction $\{k_1, k_2, k_3, \dots, x_c, y_c\}$ are considered part of the interior calibration of the camera.

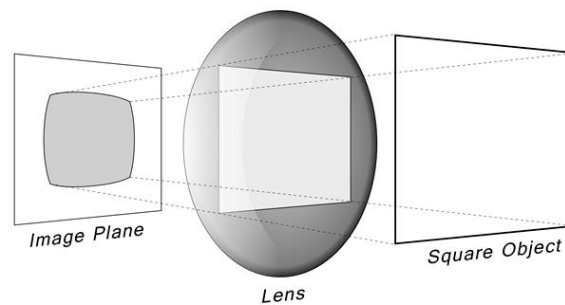


Figure 2.5: Radial distortion caused by the lens effect.

2.1.1.1 Camera Calibration

The task of the camera calibration is to determine the parameters of the transformation between an object in 3D space and the 2D image observed by the camera. This calibration process therefore, consists in determining the K values of Equation 2.5. They are called *intrinsic*, since they only depend on the internal properties of the camera. Given that, these parameters are fixed provided that the focus is not changed (for example in a video sequence), and the image resolution (i.e. zoom) is not altered. Therefore, the calibration process is only executed each time the camera configuration is modified.

The calibration process can be classified into two main techniques: *pattern-based calibration* and *self-calibration*.

Pattern-based calibration is the traditional way to calibrate the camera. The method consists in using 3D reference objects, such as those shown in Figure 2.6. This method compares the known geometric patterns against a set of images captured by the camera. Images must capture the pattern from a considerable number of different points of views in order to make an accurate estimation. Examples of these techniques can be found in (Tsai, 1987; Zhang, 2000).

By extracting a set of detected points in the pattern through a generic corner detector and establishing correspondences with the known 3D points, it is possible to determine internal parameters of the camera very rapidly and with a high level of accuracy. In the pinhole camera model, the camera is assumed to perform a perfect perspective transformation where the image coordinates (x, y) and their corresponding 3D world coordinates are related:

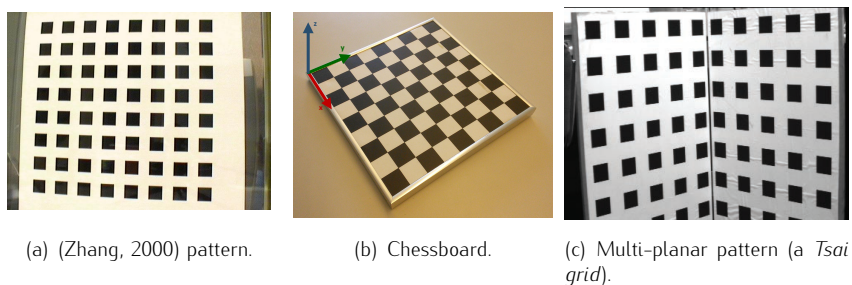


Figure 2.6: Examples of calibration patterns.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2.7)$$

where P is the perspective transformation 3×4 matrix that codifies both intrinsic and extrinsic parameters of the camera. In such a way that the non-singular matrix P can be factored as the product of an upper triangle matrix K accounting for the intrinsic parameters and R_t for camera orientation and position ($P \sim K * R_t$).

However, it is not always possible to apply a pattern-based calibration because markers cannot be added in scenes of pre-recorded videos, for example. Thus there are other alternatives such as *self-calibration* methods, also called *auto-calibration techniques*. These techniques do not require any particular calibration object. By using image information alone, the camera internal parameters can be deduced. They obtain the parameters by applying constraints to the images taken by the same camera from different points of view (multi-view geometry) (Faugeras et al., 1992; Hartley, 1997).

A set of constraints will be necessary for this procedure: parallelism and orthogonality, zero skew ($s = 0$) or constant aspect ratio ($s_x = s_y$) among others. For more details, please refer to the presented survey of self-calibration techniques in (Hemayed, 2003).

2.1.2 Extrinsic Parameters

As discussed above, the world coordinate system is aligned with the projection of the centre of the camera. Nevertheless, 3D points in the scene will be expressed in terms of the world coordinate system and not that of the camera. Therefore, assuming that the system is calibrated, it is necessary to make an additional transformation to convert world coordinates to camera coordinates (see Figure 2.7). This Euclidean transformation is composed of a rotation and translation, and represents the alignment between the camera and world coordinate systems. Taking into account this transformation, the 3D-2D projection pipeline of the Equation 2.5 results in:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K P_N R_t \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \sim K P_N \begin{bmatrix} \mathbf{R} & \vec{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2.8)$$

where R is a 3x3 rotation matrix and \vec{t} is a 3x1 translation vector. Both compose the R_t matrix called the *camera extrinsic parameters* that describes the position and orientation of the camera. K is the matrix which contains the intrinsic parameters and P_N is often convenient for clarity in the notation. It is also noticeable that in a video sequence the camera is not static, and hence it is necessary to recalculate the extrinsic parameters in each frame.

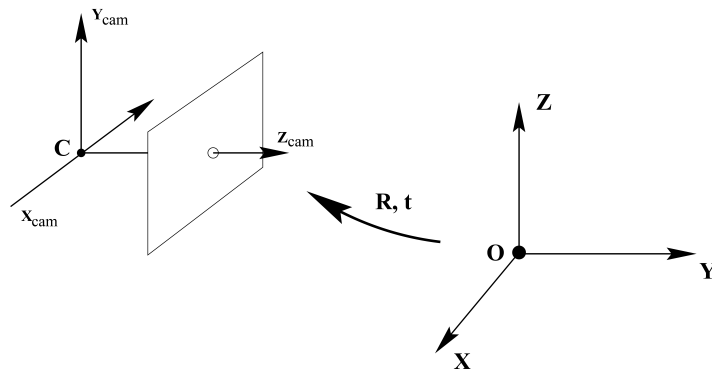


Figure 2.7: The Euclidean transformation between the world and camera coordinate frames (by (Hartley and Zisserman, 2003)).

2.2 Visual Cues

Visual cues are defined as those measures taken from an image (an array of colour pixels), which are used to perform multiple tasks, including optical tracking. Different types of visual cues can be detected in an image (see (Tuytelaars and Mikolajczyk, 2008) for a similar classification) (see Figure 2.8). These include *image points* that are very distinguishable from their neighbourhood (*features*), areas with abrupt intensity changes (*edges*), or regions with homogeneous intensity levels (*blobs*). In addition, there are more complex structures derived from these visual cues, such as *lines* (straight edges), *circles* (blobs with circular shape) or *junctions* (a point where two straight edges meet). Additionally, each visual cue is represented by its image descriptor, which is a vector that contains the image appearance (colour, intensity, etc.) of the visual cue in a local vicinity. Thus, descriptors are used to distinguish each visual cue from others.

In this thesis *features* have been used as reference visual cues. There follows a summary of some basic information concerning this type of cue.

2.2.1 Features

Feature points, also called *keypoints*, *corners* or *interest points*, are defined like a compact but rich representation of a salient point of the image. Features are characterized primarily because they can be distinguished from their local neighbourhood, providing unequivocal measurements used for image analysis.

The *feature extraction* or *feature detection* process determines the presence of a feature in the image. It is a low-level image processing step that is usually performed as the first operation of image processing. A description is then given to each feature by the *feature descriptor* process in order to identify and distinguish it from the rest. In general, descriptors contain information, usually colour, shape, region, texture and motion.

There are many feature detectors as well as descriptor algorithms that can be found in the literature. (Mikolajczyk and Schmid, 2004) describes several feature detectors, while (Mikolajczyk and Schmid, 2005) presents a comparison of different feature descriptors. Likewise, (Gauglitz et al., 2011) offers an evaluation of some feature detectors and descriptors oriented to optical tracking.

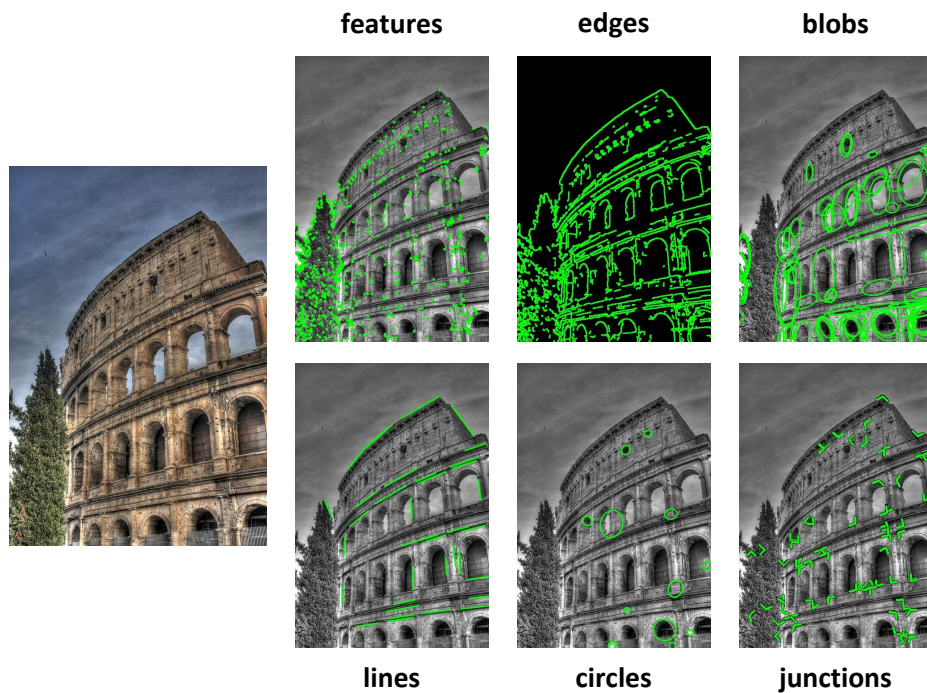


Figure 2.8: Different types of visual cues.

This section provides a brief explanation of the *Features from Accelerated Segment Test (FAST)* interest point detector (Rosten and Drummond, 2006) and the *Fast Retina Keypoint (FREAK)* (Alahi et al., 2012) and *Scale Invariant Feature Transform (SIFT)* (Lowe, 2004) feature descriptors, as they have been used in this thesis.

2.2.1.1 Detection

A *feature detector* is an approach to extracting interest points and inferring the contents of an image. It is a low-level image processing operation that examines every pixel (sometimes with sub-pixel accuracy) to determine whether there is a feature or not. The quality of a feature detector is entirely related to the following characteristics:

- **Repeatability.** Ability to detect the same corner in different images.
- **Invariant**
 - **Illumination changes.** Stable under local and global perturbations in the image such as illumination/brightness changes.
 - **Transformations.** Stability for as many transformations as possible, including camera rotations, translations, or scales.
- **Accuracy.** An important issue when selecting an appropriate feature detector.
- **Performance.** Low computational cost detecting features under different viewing conditions.

FAST

The FAST algorithm was proposed by (Rosten and Drummond, 2006) and later revised in (Rosten et al., 2010). It was developed to use in real-time applications. The segment test criterion operates by considering a discretized circle of 16 pixels (a Bersenham circle of radius 3) around the corner candidate p (see Figure 2.9). The feature detector identifies the pixel as an interest point if there is a set of contiguous n pixels in the circle that are all brighter than the intensity of p (I_p) plus a given threshold (t) I_p+t , or all darker than I_p-t (see Equation 2.9).

$$\begin{aligned}
 S_{bright} &= \{x \mid I_{p \rightarrow x} \geq (I_p + t)\}, \\
 S_{dark} &= \{x \mid I_{p \rightarrow x} \leq (I_p - t)\}, \\
 p \in Features &\Leftrightarrow (|S_{bright}| \geq n) \quad (|S_{dark}| \geq n). \quad (2.9)
 \end{aligned}$$

n was chosen to be 12 (FAST-12) or 9 (FAST-9) in the first version of the algorithm. For each location on the circle $x \in [1..16]$, the pixel at that position relative to p is denoted by $p \rightarrow x$.

A high-speed test is also used to exclude a large number of non-corners. This test examines only the four pixels at 1, 9, 5 and 13 (called the four compass directions). At least three of these must be brighter or darker than I_p and the threshold. The full segment test can then be applied to the remaining candidates by examining all the pixels in the circle.

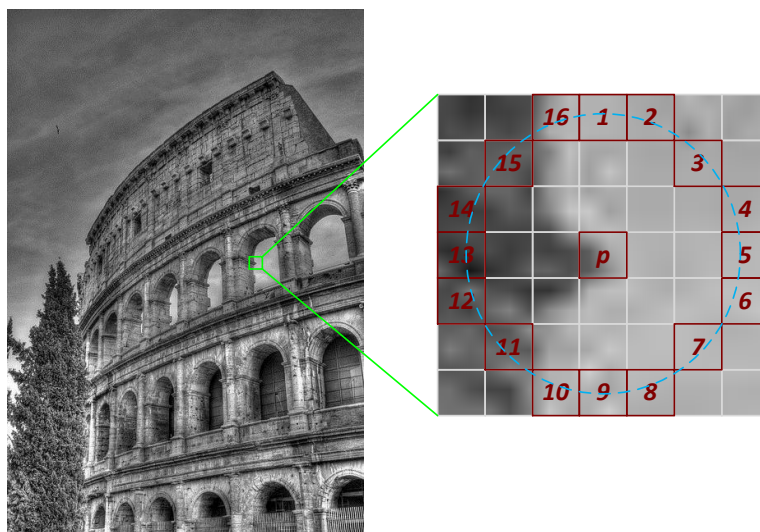


Figure 2.9: FAST feature detection (left) and enlarged image patch of a detected corner (right).

Detection of multiple interest points adjacent one to another was one of the problems of the initial version of the algorithm. To address it, a score function V is computed to apply a non-maximal suppression, which gets stable features by discarding corners with lower *response* value than the adjacent one (see Equation 2.10). The response (quality) of each feature p is given by the intensity contrast between I_p and its surrounding pixels, i.e., the sum of the absolute difference between the pixels in the contiguous arc and the centre pixel:

$$V = \max \left(\sum_{x \in S_{\text{bright}}} (|I_{p \rightarrow x} - I_p| - t), \sum_{x \in S_{\text{dark}}} (|I_p - I_{p \rightarrow x}| - t) \right) \quad (2.10)$$

In short, this feature detector offers a good balance between repeatability and performance (it processes high resolution images in a few milliseconds), which is the reason for its popularity in real-time applications. Additionally, the set of final features can be categorized as bright or dark at no extra cost, which is useful, since bright features do not require to be compared with dark features in post-processing steps such as matching (see Section 2.2.1.4).

2.2.1.2 Description

Once a keypoint is detected, a higher level of detail is given by its *descriptor* (see Figure 2.10). The descriptor is the representation of its corresponding keypoint, and it is extracted from the image region of its vicinity. Thus, the similarity measure of feature descriptors is used to get the correspondences in different images to facilitate the matching step (see Section 2.2.1.4). A good feature descriptor should satisfy the following properties:

- **Highly distinctive:** Two different features should have two different descriptors, i.e., the probability of a mismatch is low.
- **Robustness:** The descriptor of a feature should remain unchanged despite the transformations applied to the image, i.e., the descriptor of a feature is preserved after rotations, translations, scales or illumination changes.
- **Good performance:** The computational cost of building and comparing feature descriptors should be low, being able to run in real-time systems. The Euclidean distance between two descriptors is usually enough to measure their similarity.

To compare two image patches, first the descriptors are computed and then their similarities are measured by the descriptor similarity, which in turn is done by computing their descriptor distance.

In this sense, two types of descriptors can be found: *histogram-based patch descriptors* and *binary descriptors*. The former are based on histogram of gradients, as is the case of the widely used SIFT feature descriptor. The binary descriptors, in turn, encode the most information of a patch as a binary string using only comparison of pixel intensity. This can be very fast, as only intensity comparisons need to be made. Using the Hamming distance as a distance measure between two binary strings, the matching between two patch descriptions can be achieved using a single instruction (as the Hamming distance equals the sum of the XOR operation between the two binary strings). FREAK is one of the multiple binary descriptors that can be found in the literature.

Table 2.1 shows the differences between the SIFT (histogram) and FREAK (binary) descriptors, which are described in the following paragraphs in detail.

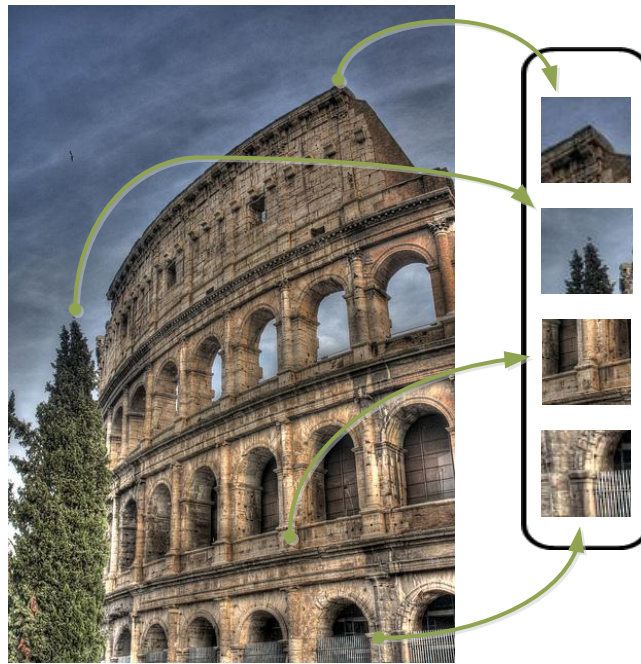


Figure 2.10: Extracting descriptors from detected keypoints.

SIFT

SIFT transforms image data into scale-invariant coordinates relative to local features. This method proposed by (Lowe, 1999; Lowe, 2004) includes both a feature detector and a feature descriptor. The first makes features invariant to image scaling and rotation, while the second is as invariant as possible to remaining variations such as change in illuminations and 3D camera viewpoints.

The *corners detection* is performed by using the *Difference of Gaussian (DoG)* function. It is able to detect stable keypoint locations in a scale space defined as a function L . The input image (I) is incrementally convolved with different Gaussian kernels (G) to produce these multiple scales:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \quad (2.11)$$

where $*$ is the convolution operation.

Table 2.1: Differences between SIFT and FREAK descriptors.

	SIFT	FREAK
Type	Histograms of gradients	Binary descriptor
Includes	Corner detector and descriptor extractor	Descriptor extractor
Invariance	Illumination and rotation	Slightly negative performance in rotation and illumination
Computation	Histograms of gradients orientation	Comparison of intensity images
Distance measure	Euclidean distance	Hamming distance
Matching	K -d Tree	Brute Force
Computational cost	High	Low
Intellectual property	Patented ('non-free')	Non-patented

To efficiently detect stable keypoint locations in scale space, the adjacent Gaussian images separated by a constant factor k are subtracted to produce DoG function D as can be shown in Figure 2.11 (left).

Pixels that are local maxima and minima are considered features points. Each sample point is compared to its 8 neighbours in the current scale and 9 neighbours in the scale above and below (see Figure 2.11 right). The keypoint will be selected only if it is larger or smaller than all of these neighbours. Additionally, the scale where the corner is detected is stored, as it determines the size of the local image region used to build feature descriptors (*scale invariance*).

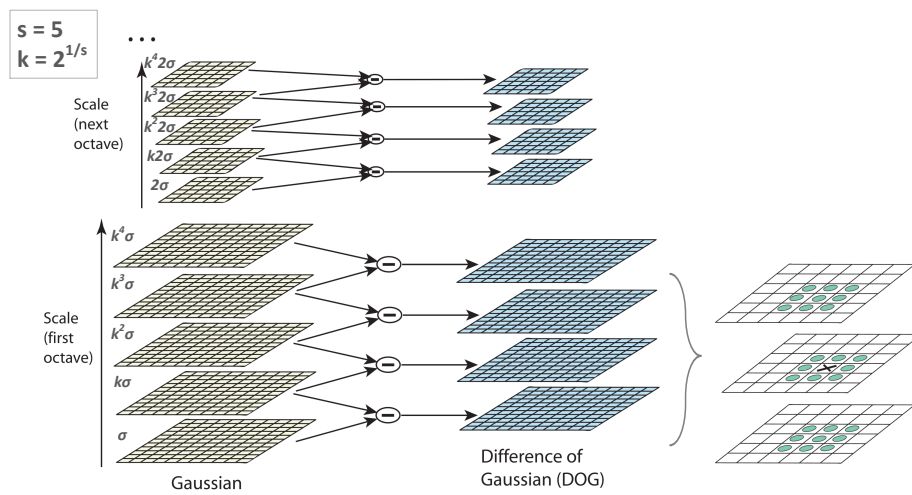


Figure 2.11: DoG process (left) and maxima and minima of the DoG (right). Original source (Lowe, 2004).

Once the localizations of keypoints are defined (see Figure 2.12), each located corner is represented by a *descriptor* (SIFT descriptor). It is computed using the *location* (x, y), *scale* (σ), *gradient magnitude* (m) and *gradient orientation* (θ) information of each keypoint.

First, the *orientation assignment* is defined for each keypoint. An orientation histogram is determined by computing the gradient orientations of sample points within a window around the keypoint. This histogram is discretized in 36 bins covering the 360 degree range of orientations (10 degrees per orientation). The amount added to the bin is weighted by the gradient magnitude of each sample point and a Gaussian-weight function. This function gives less importance to gradients that are far from the centre of the descriptors. The dominant orientation of a keypoint is defined by the highest peak of the histogram. If more than one orientation is assigned to a keypoint, then the corner is duplicated with different dominant orientations. In order to achieve *rotation invariance*, the coordinates and gradient orientations of the pixels that belong to the local image window of the features are rotated according to the corresponding dominant orientation.

The image window for each detected corner is defined by the subdivision of $n \times n$ subregions around the keypoint, and each subregion is characterized by an orientation histogram of b bins, where n and b are application dependent



Figure 2.12: Feature detection using DoG. Each circle represents the scale and orientation of a feature.

parameters defined by the user (see Figure 2.13). Finally, a single histogram deduced from the concatenation of all these histograms represents the descriptor. Furthermore, it is normalized to unit length to reduce the effects of illuminations. Generally, $n=4$ and $b=8$ is a typical parametrization, giving a 128 ($4 * 4 * 8$) dimensional feature vector.

SIFT provides an invariance to illumination changes, as gradients are invariant to light intensity shift, and to rotations, as histograms do not contain any geometric information. SIFT is also a very robust technique for object recognition, by computing the matching correspondences using the distinctive SIFT keypoints between a reference and an input image. However, while the SIFT feature is robust, the computational cost involved is high.

There are other similar alternatives to the SIFT descriptor such as *Gradient Location and Orientation Histogram (GLOH)* (Mikolajczyk and Schmid, 2005) or *Speed-Up Robust Features (SURF)* (Bay et al., 2008). SURF, for example, uses

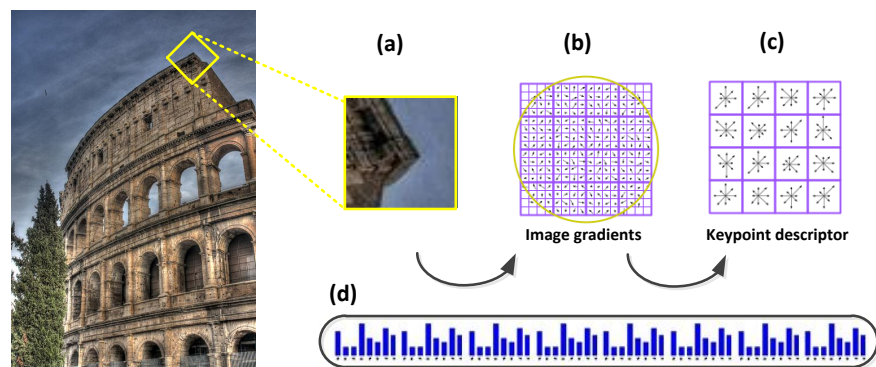


Figure 2.13: SIFT descriptors illustration. By warping the region around the keypoint (a), it is divided into subregions and the orientation is calculated for each one (c). Concatenating histograms from different subregions the final descriptor is performed (d).

efficient tools (as *integral images*) to reduce the computational cost. However, it is not fast enough for some real-time applications. In this sense, a simplified version of SIFT has been used for this dissertation, called *simplified-SIFT* (Álvarez-Ponga, 2012). The main contribution of this implementation is the replacement of the expensive DoG by the FAST detector and the inclusion of some parallel techniques to reduce the computational cost.

FREAK

The binary descriptors are generally composed of three main parts: a *sampling pattern*, *sampling pairs* and *orientation compensation*. By considering a small patch around a keypoint, the descriptor is defined as a binary string. The first step involves taking a sampling pattern around the keypoint. The second consists in selecting a list of pairs and comparing the intensities between the two points of each pair in order to build the final descriptor ('1' if the first value is larger than the second or '0' otherwise). Finally, the orientation is the mechanism for measuring the orientation of the keypoints. By rotating the patch by its dominant orientation and extracting again the binary string it can ensure that the description is invariant to rotation.

FREAK (Alahi et al., 2012), like many other binary descriptors, is also composed of these three steps. FREAK is a descriptor inspired in Human Visual System, and more precisely in the retina. In this way, a *retinal sampling pattern* is suggested, i.e., a circular pattern where points are spaced on circles having higher density of points near the centre as a retinal model (see Figure 2.14 (a)).

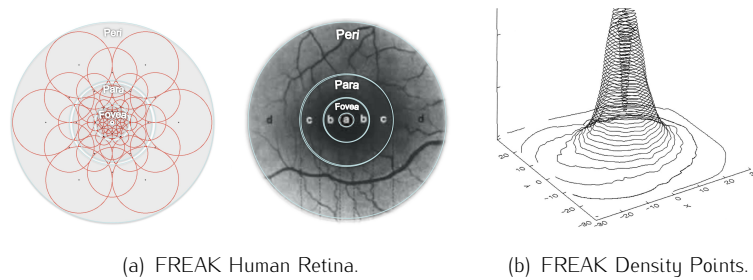


Figure 2.14: The human retina with the distribution of receptive fields over the retina (a) and the density of points (b).

Each sample point requires to be smoothed to be less sensitive to noise. FREAK uses a different kernel size for every sample point, since it has been observed that changing the size of the Gaussian kernels with respect to the log-polar retinal pattern leads to better performance as well as overlaying the receptive fields Figure 2.14 (a). Each circle in Figure 2.14 (b) represents the standard deviation of the Gaussian Kernels applied to the corresponding sampling points.

Once the sampling points have been defined, a set of *sampling pairs* are considered. To increase performance, FREAK tries to learn the pairs by maximizing the variance of the pairs and taking those that are not correlated. It also takes advantage of a coarse-to-fine approach to approximating the resulting pairs to the model of a human retina. First, the pairs that are selected mainly compare sampling points in the outer rings of the pattern while the last set of pairs mainly compare points in the inner rings of the pattern. This procedure is similar to that of the human eye. First, the perifoveal receptive fields are used to estimate the location of an object of interest and then, the validation is performed with the more densely distributed receptive fields in the fovea area.

For computing the *orientation* of the keypoint, FREAK uses local gradients between the sampling pairs. It sums up all the local gradients over selected pairs and the *arctangent* of the component y divided by the component x of

the gradient gives the angle of the keypoint. Consequently, the sampling pairs are rotated by that angle (*rotation invariance*). FREAK also selects a predefined set of symmetric sampling pairs with respect to the centre (see Figure 2.15) in order to compute the global orientation.

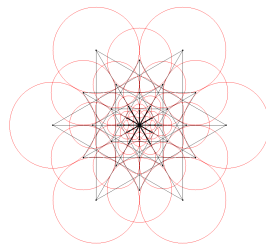


Figure 2.15: FREAK's sampling pairs.

FREAK is considered to be a fast, compact and robust keypoint descriptor. Its technique of selecting pairs to reduce the dimensionality of the descriptor further accelerates the matching. Compared to other binary descriptors such as *Binary Robust Invariant Scalable Keypoints (BRISK)* (Leutenegger et al., 2011), *Binary Robust Independent Elementary Features (BRIEF)* (Calonder et al., 2010) or *Oriented FAST and Rotated BRIEF (ORB)* (Rublee et al., 2011), FREAK slightly outperforms in viewpoint changes, rotation changes or illumination changes. Nonetheless, is slightly worse when there are changes due to blurring.

2.2.1.3 Motion

Knowing the location and description of a corner in a given image, it is possible to determine its position in any other input image. The use of descriptors to perform the matching between two images is a valid alternative (next Section 2.2.1.4). However, the computational cost involved in such solutions is usually high. As an alternative, there are other methods like *optical flow* that estimate the motion between two consecutive image frames. These methods are associated with the idea of not having a reference pattern in order to deduce the transformation between two consecutive frames. They only require the information of the previous frame. Likewise, the optical flow technique is usually associated with the *Frame-to-Frame tracking (FtF)* step in the image processing field.

Optical Flow

Optical flow determines the motion of several keypoints between two consecutive images (taken at times t and $t + \Delta t$) using intensity differences (see Figure 2.16). It is thus assumed that the motion between the two images is small. For a corner at location (x, y) with intensity I at time t that is moved by Δx , Δy and Δt between two images can be given as:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \quad (2.12)$$

This method is called differential since it is based on local Taylor series approximation of signal. Optical flow therefore satisfies the following constraints:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\delta I}{\delta x} \Delta x + \frac{\delta I}{\delta y} \Delta y + \frac{\delta I}{\delta t} \Delta t + H.O.T., \quad (2.13)$$

where $\frac{\delta I}{\delta x}$, $\frac{\delta I}{\delta y}$ and $\frac{\delta I}{\delta t}$ are partial derivatives of I , and $H.O.T.$ are the higher order terms of Taylor series. Assuming a small movement, $H.O.T.$ can be disregarded, resulting in $\frac{\delta I}{\delta x} \Delta x + \frac{\delta I}{\delta y} \Delta y + \frac{\delta I}{\delta t} \Delta t = 0$, which is a single equation with two unknowns (Δx and Δy , corresponding to the feature motion).

Lucas-Kanade is a differential method that solves the *aperture problem* by assuming that the flow remains constant in the local neighbourhood for a sparse feature set. Adding this constraint, it provides the following least squares solution:

$$\epsilon(\Delta x, \Delta y) = \sum_{u, v \in W} [I(u, v) - J(u + \Delta x, v + \Delta y)]^2, \quad (2.14)$$

where I references the first image intensity on the image point u and J in the second, W is a small integration window defining the local neighbourhood, and e is the residual function. Thus, the goal is to find a location $v = u + A$ on the second image where $I(u)$ and $J(v)$ should be similar.

Accuracy and robustness are the main two properties of feature trackers. The first implies small values of the window size since is related with selecting the closest local sub-pixels, while bigger window patches are preferable for robustness in order to be sensitive to illumination changes or large motions. A trade-off between these two requirements is thus required. (Bouquet, 2001)

proposes a solution for this problem through a pyramidal implementation of the Lucas-Kanade algorithm. It consists in computing the optical flow at the deepest resolution image. The result is then propagated to the next image results as an initial guess. This procedure is repeated up to level 0 that is the highest resolution (initial image). Observe that the window of integration is constant size for all resolutions, so large motions computed at low resolutions are refined by the accurate estimation of high ones.

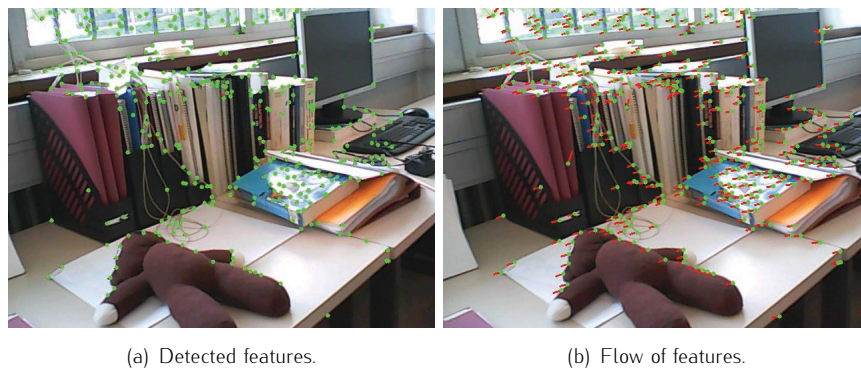


Figure 2.16: Optical flow for two images of a sequence.

Among the drawbacks involved in optical flow techniques, is the liability to drift. This is a very classical issue when dealing with long sequences. The errors in the estimations are dragged over the time. Additionally, the assumption of small motion between images makes losses in sudden movements or even cases of lost features when a point falls outside of the image.

2.2.1.4 Matching

Unlike the motion technique explained in the previous section, the matching process does not require a knowledge of the previous frame, since a comparison against a reference image (its 3D information is known) is performed. This process is usually called *Tracking-by-Detection (TbD)* step in CV area.

Matching based on descriptors relates a feature point in the current image with its homonym in the reference image (see Figure 2.17). At runtime, interest points alongside their descriptors are extracted from the input frame to search out the most similar ones in the database. The extraction must use the same type of descriptor in both offline and online phase.

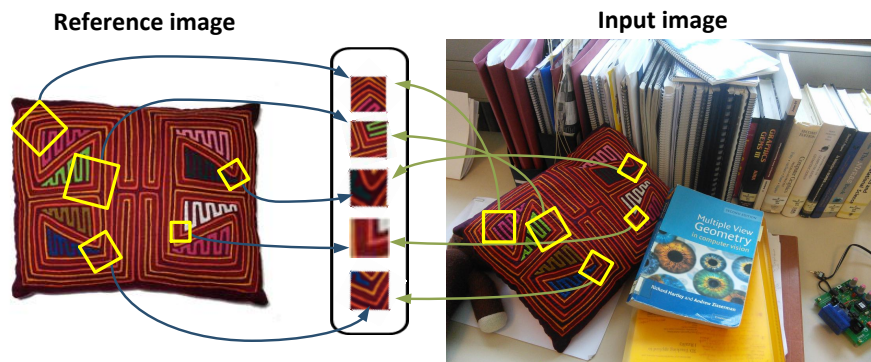


Figure 2.17: Correspondence matching of features between the reference image of an object (left) and the input image (right).

This searching cost depends on the number of features stored in the database, the number of queries to the database and the dimension and type of the descriptor. The distance between descriptors determines whether they are similar or not. In the case of descriptors based on histogram of gradients (e.g. SIFT), the Euclidean distance is the measurement used to determine the distance. If a descriptor vector is described with a binary string (e.g. FREAK), the comparison is performed using the Hamming distance.

To make the search, the *K-Nearest Neighbour (K-NN)* rule (Cover and Hart, 1967; Silverman and Jones, 1989) is the most appropriate. The *K-NN* is a well-known problem in computer graphics, statistics and robotics. Let $R = \{r_1, r_2, \dots, r_m\}$ be a set of m reference descriptors, and $Q = \{q_1, q_2, \dots, q_n\}$ a set of n queries. The *K-NN* search problem consist in searching the k nearest neighbours of each query $q_i \in Q$ in the reference set R using one of the distance listed below (see Figure 2.18).

The brute-force matchers as well as the k -d trees based search are the most frequent methods for computing the *K-NN*. Thus, both are detailed in the present study, as well as the *Locality Sensitive Hashing (LSH)* technique.

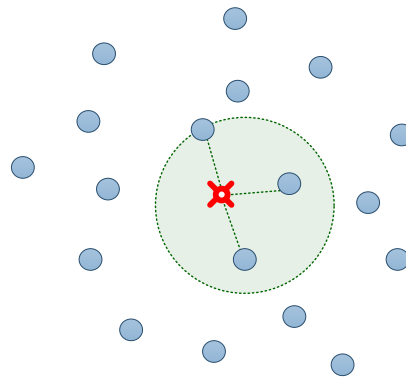


Figure 2.18: Illustration of the K -NN search algorithm for $k = 3$. The blue points correspond to the reference points and the red cross corresponds to the query point. The circle gives the distance between the query point and the third closest reference point.

Brute Force

One way to search the K -NN is the *Brute Force (BF)* algorithm or *exhaustive search*. It is a simple matcher that takes a query descriptor q from the query set and checks it against all other features in the reference set using a distance calculation. The k closest ones are returned. Therefore, each query q_i value follows these steps:

1. Compute all the distances between q_i and r_j , $j \in [1, m]$.
2. Sort the computed distances.
3. Select the k smallest distances (reference corresponding points).

While the BF algorithm is simple to implement, the main issue of this search is its high computational cost since is proportional to the number of values in each set.

K-d Trees

The second method for searching is based on *k-d trees* (Friedman et al., 1977). The *k-d tree* is a binary tree used for sorting and searching data (see Figure 2.19). It is a data structure in a partitioned space that organizes the data in a Euclidean space of k dimensions. k refers to the dimension of the tree. Each node of the tree is a subset of the entire dataset. The root represents all the data and each non-leaf node has two sons or successor nodes. All the leaf nodes, called *buckets*, represent exclusive small subsets of the entire dataset. The *k-d tree* uses only perpendicular axis of the coordinate system. This is the main difference from the *Binary Space Partitioning (BSP)* trees where the planes could be arbitrary. The structure of the *k-d tree* provides an efficient mechanism for examining only those records closest to the query record, thereby greatly reducing the amount of computation required to find the best matches.

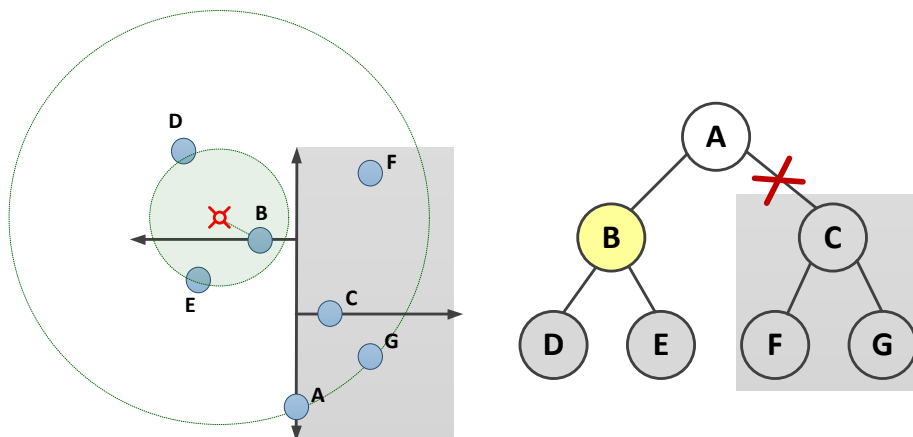
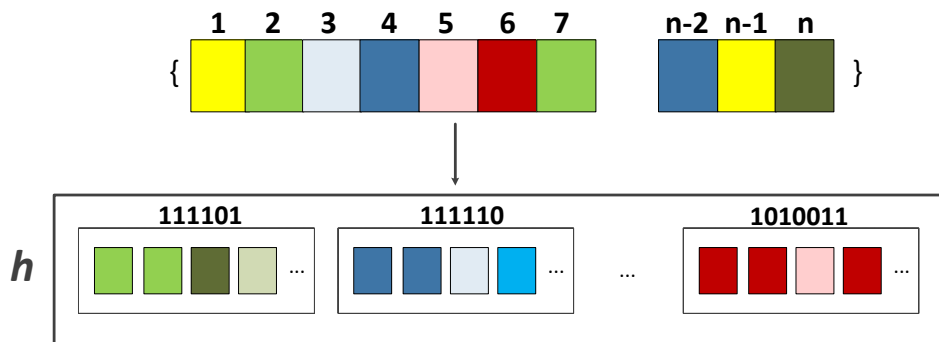


Figure 2.19: Illustration of the *k-d tree* search algorithm. NN searching with a *k-d tree* in two dimensions.

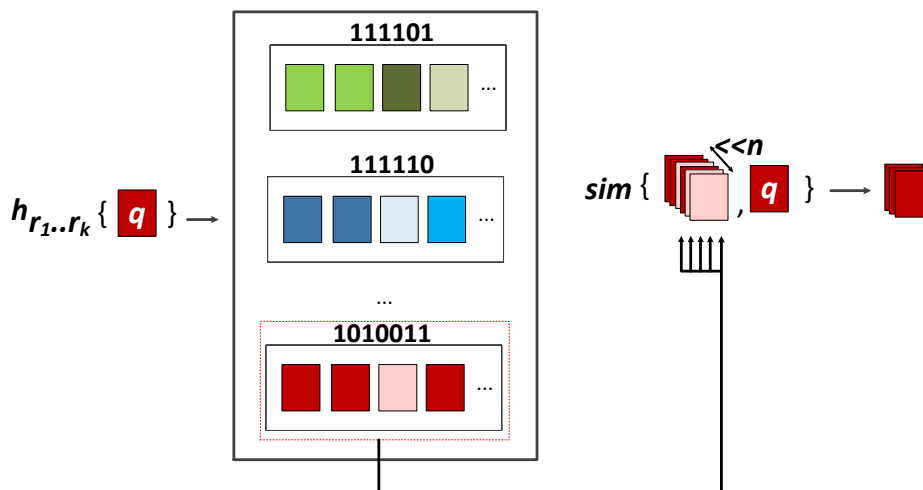
In this thesis the *k-d trees* are space-partitioning data structures that organize descriptors (multi-dimensional space). They offer an efficient nearest neighbour search.

LSH

LSH (Indyk and Motwani, 1998) is an indexing technique for an approximate similarity search where a probabilistic dimension reduction of high dimensional data is performed. The idea is to use hash tables called *buckets* and map similar objects into the same hash buckets through a family of locality-sensitive hash functions (similar objects in the same buckets with high probability) (see Figure 2.20).



(a) LSH Database.



(b) LSH Query.

Figure 2.20: LSH consist in a reduction of high dimensional data into new family of hash functions.

To perform the similarity search, the query object (q) represented by d -dimensional feature vector, requires to find the closest K objects according to a distance function. The distances are within a small factor $(1 + \epsilon)$ by solving the problem as the approximate nearest neighbours problem. However, in order to cover the nearest neighbours a large number of tables is required and therefore, more comparison are made. This is the main drawback of the basic LSH indexing.

Multi-probe LSH indexing (Lv et al., 2007) in turn, uses a more systematic approach to explore hash buckets. It is built like the basic LSH but uses an intelligent probing sequence to look up buckets that have a high probability of containing the nearest neighbours of a query object.

2.2.1.5 Outlier Removal

(Hawkins, 1980) defines an *outlier* as *an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism*. In CV many of the correspondences established in the matching stage could be outliers due to the accuracy of the matching process. As a result, outlier detection and removal steps are necessary in order to achieve a robust set of correspondences. The *RANdom Sample and Consensus (RANSAC)* method is a good mathematical model estimator to discard outliers, and consequently estimate robust camera pose.

RANSAC

The RANSAC algorithm was introduced by (Fischler and Bolles, 1981) as an iterative method to estimate parameters of a mathematical model from a set of data which contains a large number of outliers. The idea behind this algorithm is to select the lowest amount of data points to make an estimation of the model and check how many points fit to the estimated model. Other approaches that estimate the model parameters, such as the method of least squares, give the same weight to each data point of the set. The presence of an outlier may therefore distort the obtained model.

The original RANSAC algorithm is essentially composed of two main steps that are repeated in an iterative way (hypothesis-and-test):

1. Hypothesize. First a *Minimal Sample Set (MSS)* S_1 is randomly selected from the input dataset P and the model M_1 is computed using only the

elements of S_1 . The cardinality of the MSS is as small as possible, enough to determine the model parameters (in the case of estimating a line, for example, the MSS is 2 since two distinct points are required to uniquely define a line).

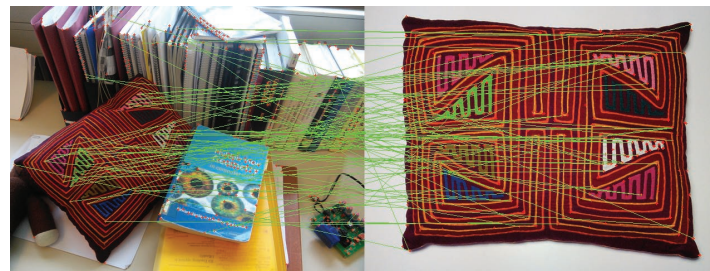
2. Test. In the second step, RANSAC determines a subset S_1^* from P that is consistent with the parameters estimated in the first step. The set of such elements is called a *Consensus Set (CS)*.

RANSAC finishes when the number of votes of the CS exceed a certain predefined value. If the number of inliers S_1^* is higher than a given threshold, then the subset S_1^* is used to compute a new model M_1^* . Otherwise, if the number of inliers is lower than the threshold, then a new subset is selected to make the same process. After N iterations if there is not a subset that achieves this threshold, then the best subset is returned.

As can be seen from the process of the algorithm, there are three parameters that must be estimated: (i) the distance t to consider if a point is consistent or not with the model, (ii) the number of iterations N and (iii) the threshold T that is the number of points that must be overcome to consider the subset good enough.

Additionally, the RANSAC approach is a good method for estimating the global movement between two planes in space. The relation between these two planes is called *planar homography*. Therefore, this approach is primarily used in dealing with planar-surface tracking, since it is just a particular case of the homography estimation. Assuming that $Z = 0$ and knowing the internal parameters of a pinhole camera, RANSAC computes the relation between two images of the same planar surface in space. For more details of the resolution of this mathematical problem, please refer to Appendix A.

However, the estimation of the homography from a set of correspondences between two images obtained from a feature detector can involve erroneous matches. These outlier matches usually have undesired effects in the homography calculation. RANSAC is a reasonable alternative for solving this problem (see Figure 2.21). In the particular case of estimating the homography with RANSAC, four random points from the total of the correspondences are enough to be selected to compute an approximated H_0 . Based on this approximation, all the correspondences are checked to see if they are compatible or not using a certain threshold. If the number of inliers is good enough, the approximated H_0 is refined



(a) No RANSAC matching.



(b) RANSAC matching.

Figure 2.21: Matching without (a) and with (b) RANSAC estimation.

to achieve a robust result. Thus, computing the planar homography in a new tracking step is determined as the Algorithm 2.1.

The random sampling is a good technique in order to remove inaccurate or mismatched corner locations from the homography computation. Thus, the homography is more robust to partially occluded features.

Another variant to RANSAC is called *PROgressive Sample Consensus (PROSAC)*. The difference lies in the hypothesize-and-verify (sample-and-test) method. The most probable matches have a higher probability of being selected. Based on the assumption that correspondences with high similarity are more likely to be inliers, the probability of finding better hypotheses increases using a reduced set of matches with high similarity scores. (Chum and Matas, 2005) exploits the idea of ordering the set of correspondences than random ordering. The comparison of PROSAC and RANSAC on randomly ordered matches showed that their performance was identical. Therefore, the samples in PROSAC are drawn in a semi-random way. Nevertheless, the performance of PROSAC and RANSAC for the worst-case situation (where the correspondences are ordered randomly) has not been proved analytically. The main challenge of the PROSAC

Algorithm 2.1 RANSAC Homography estimation

```
1: function RANSAC (NumIterations, AllCorrespondences)
2:   InitializeVariables()
3:   for  $i = 0 \rightarrow \text{NumIterations}$  do
4:     LocCorr  $\leftarrow$  selectCorrespondences(AllCorrespondences)
5:      $H' \leftarrow$  computeHomography(LocCorr)
6:     if bestHomography( $H'$ ) then
7:        $H \leftarrow H'$ 
8:     end if
9:   end for
10:   $H \leftarrow$  refine(InlierCorrespondences)
    return  $H$ 
11: end function
```

method is to define the prior knowledge of the probabilities of each of the correspondences to be selected.

2.3 Camera Tracking

As mentioned previously, in order to overlay virtual objects into a real scene it is necessary to set a point of view (*camera tracking*). The camera tracking is known as the problem of finding the parameters that define the camera, which is the process that extracts the position and orientation of the camera (jointly called *pose*) relative to a global coordinate system (usually cited as *world coordinate system*). More precisely, optical tracking finds the camera's extrinsic parameters (R_t) that best align the camera and real-world coordinate systems. To that end, the image(s) of the scene captured by the camera(s) are processed using CV techniques. CV is a field of study and research that focuses on interpreting the world that is seen in one or more images. It is used in AR to calculate the camera pose, recognizing some visual cues in the image(s) captured by the camera(s).

Given an input image, the image positions of some visual cues are detected and matched with their corresponding 3D locations to extract the camera pose. This problem can be expressed mathematically assuming the pinhole camera model, solving $\vec{m}_i = P\vec{M}_i$ for a set of $\vec{m}_i \leftrightarrow \vec{M}_i$ correspondences, where \vec{m}_i is the 2D image position of the visual cue i , \vec{M}_i are their corresponding 3D coordinates, $P = KR_t$, and K represents the intrinsic parameters that describe the characteristics of the camera (focal length, skew, etc.), which can be extracted through a camera calibration process. The *Direct Linear Transformation (DLT)* algorithm (Hartley and Zisserman, 2003) solves that linear equation in the case where the camera is not calibrated (K unknown). Apart from this technique, the *Perspective-n-Point (PnP)* methods (Lepetit et al., 2009) are also used when K is known. Nevertheless, all these linear methods lack precision when the \vec{m}_i measurements are inexact (generally termed *noise*), so it is preferable to use a non-linear minimization of the reprojection error, i.e., the squared distance between \vec{m}_i and the projection of \vec{M}_i :

$$\operatorname{argmin}_{R_t} \sum_i \left\| \vec{m}_i - KR_t\vec{M}_i \right\|^2 \quad (2.15)$$

The non-linear least-squares *Levenberg-Marquadt (LM)* algorithm (Madsen et al., 1999)¹ is extensively used to solve Equation 2.15. It is an iterative process that converges onto the local minima by combining the Gauss Newton method

¹A widely used open source implementation of this method can be found in <http://www.ics.forth.gr/~lourakis/levmar/>.

with the Gradient Descent approach. Moreover, it requires a starting point, so the estimation computed by a linear method (DLT-PnP) is used to initialize the final solution.

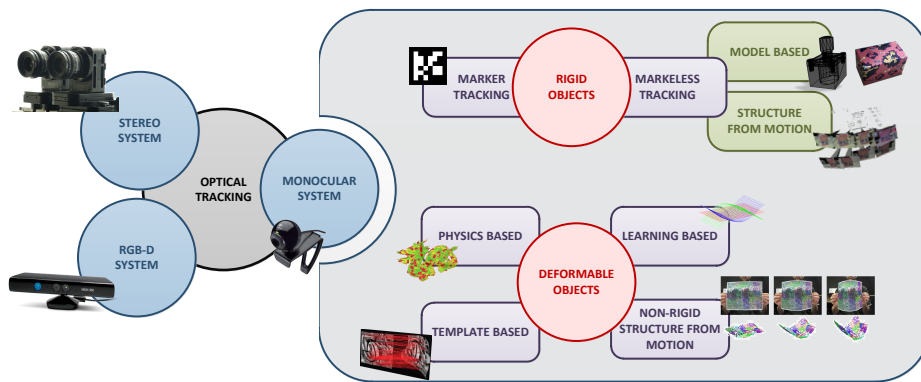


Figure 2.22: Classification of the optical tracking methods.

There are many ways to get $\vec{m}_i \leftrightarrow \vec{M}_i$ correspondences, which result in different optical tracking methods. A possible classification of the existing optical tracking methods is shown in Figure 2.22, which is detailed in the next sections. This thesis has been primarily focused on *Monocular System*, but most of the monocular methods are also applicable to the *Stereo System* and *RGB-D System* branches.

2.3.1 Stereo System

A stereo system processes several images of a scene at the same time. The images are taken from different points of view by a set of strategically placed cameras. In its simplest form, this is similar to the biological stereo vision of the human visual system, where two images of the same scene are captured from two different and known locations (left and right eyes). Similarly, the 3D information about an object that appears in both images is extracted by triangulating its 2D image positions. This is how humans perceive the depth of objects in a scene. This also explains why it is hard to estimate the distance of an object when the vision of one eye is lost. After triangulating all the point correspondences, a depth image map can be obtained, like that presented in Figure 2.23.

In order to perform the triangulation, those points that are the projections of the same point in the 3D space must be identified in two or more views

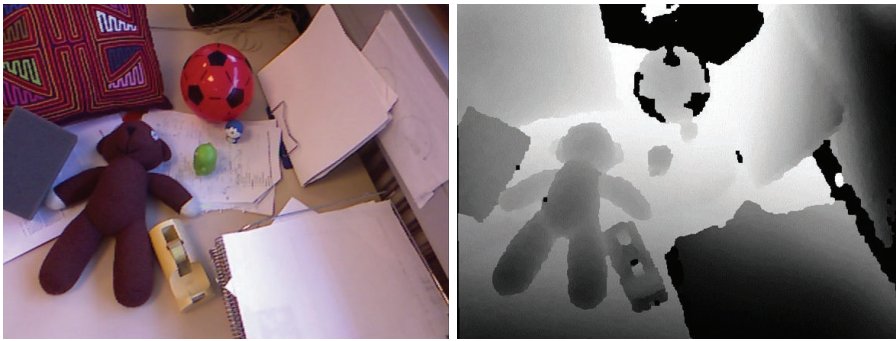


Figure 2.23: Depth image (right) for a given scene (left).

(*correspondence problem*). Generally speaking, the image's appearance in the local vicinity of each point (*point-descriptor*) is used to match points throughout a sequence of images. These correspondences are constrained by *epipolar geometry* (Hartley and Zisserman, 2003) (see Figure 2.24) in order to simplify the problem of matching. Given a point in the left image (\vec{x}_0), epipolar geometry states that its correspondence in the right image (\vec{x}_1) belongs to a straight line (\vec{l}_1). The line that belongs to the right image (\vec{l}_1) is the projection of the ray formed by the left optical centre (\vec{C}_0) and the image point \vec{x}_0 , so it is called an *epipolar line* associated to \vec{x}_0 . This is analogous for points in the right image and their left correspondences.

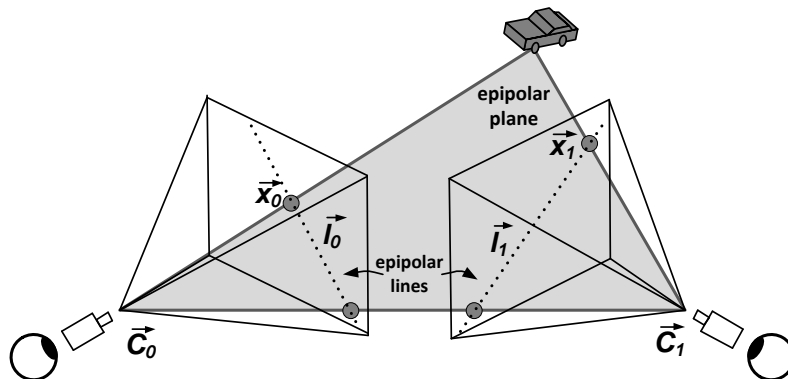


Figure 2.24: Two camera geometry.

The extraction of the 3D information, together with the known camera calibration parameters, facilitates the process of relating visual cues with their 3D values, i.e., it helps to solve Equation 2.15. The disadvantage of a stereo system is that it requires bulky and expensive hardware. For further reading on stereo systems, please refer to (Brown et al., 2003).

2.3.2 RGB-D System

Nowadays low-cost sensors are presented as an interesting alternative to the traditional laser scanners or 3D-cameras such as time-of-flight-based cameras. Accordingly, structured-light 3D scanners such as the Microsoft Kinect Sensor² have emerged as an alternative to pure optical stereo systems. This is because devices like the Kinect provide depth estimation by using a low-cost and a small hybrid system.

The Kinect Sensor was primarily introduced as an input device for computer game environments (PrimeSense³). However, the CV community quickly discovered that the characteristics offered by this sensor (depth estimation) could be used for other purposes.

Currently, there are three software frameworks for the Kinect Sensor: Microsoft SDK⁴, OpenNI⁵ and OpenKinect⁶. The first is only available for Windows platforms whereas the other two frameworks are multiplatform and open source. Another major difference is that Microsoft SDK has a depth range of $\sim 0.8m$ and $\sim 4m$, while the other two are limited to the depth range of $\sim 0.5m$ and $\sim 9m$ (Andersen et al., 2012).

Kinect Sensor

The Microsoft Kinect camera is composed by an *Infrared (IR)* projector, a 320x240 16 bit IR camera and a 640x480 32bit RGB camera (see Figure 2.25). Both cameras run at 30 fps and the *Field of View (FoV)* is 57° horizontally and 43° vertically. It also has a multi-array microphone for detecting voice commands.

²Microsoft Kinect Xbox 360 <http://www.xbox.com/en-us/kinect/>.

³PrimeSense <http://www.primesense.com/>.

⁴Microsoft SDK <http://www.microsoft.com/en-us/kinectforwindows/>.

⁵OpenNI <http://www.openni.org>.

⁶OpenKinect <http://www.openkinect.org>.

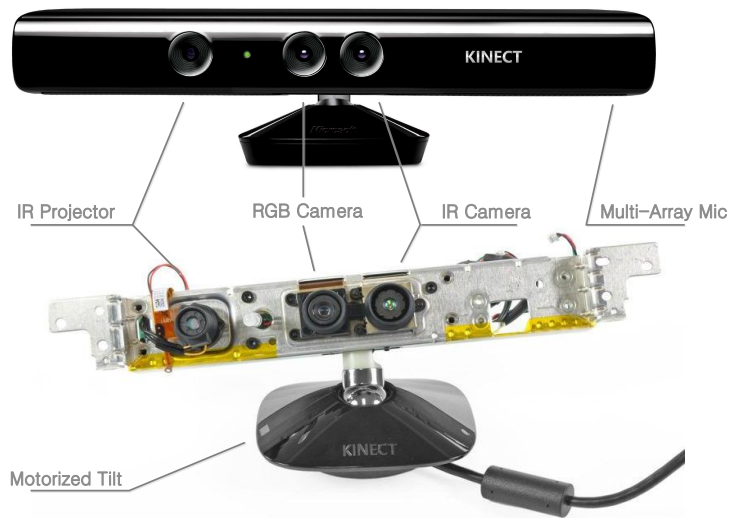


Figure 2.25: The Kinect sensor consists of an IR laser emitter, an IR camera and an RGB camera.

The depth sensor in turn, consists of an IR structured light projector (IR projector) combined with a CMOS sensor (IR camera). The IR projector emits a fixed pattern of light and dark dots projected onto the scene and the reflection is captured by the CMOS IR camera (see Figure 2.26).

One of the main characteristics of the Kinect sensor is that it allows the third dimension of the environment (computing depth images) to be determined, making the task much easier. It involves triangulation from the IR image and the projector. By projecting a pattern of speckles onto the scene, this pattern is captured by the IR camera resulting in the depth image. For more detailed overview of the depth information, please refer to Appendix B.

The calibration of the Kinect, involves the following parameters in order to satisfy the previous triangulation process: focal length (f), principal point offsets (x_0, y_0), lens distortion coefficients (d_x, d_y), base length (b) and distance of the reference pattern (Z_0). Moreover, it happens that the IR and RGB cameras are not aligned. Therefore, it is necessary to calculate three rotations between the camera coordinate system of the RGB camera and that of the IR camera, as well as the 3D position of the RGB camera with respect to the coordinate system of the IR camera. In addition, the

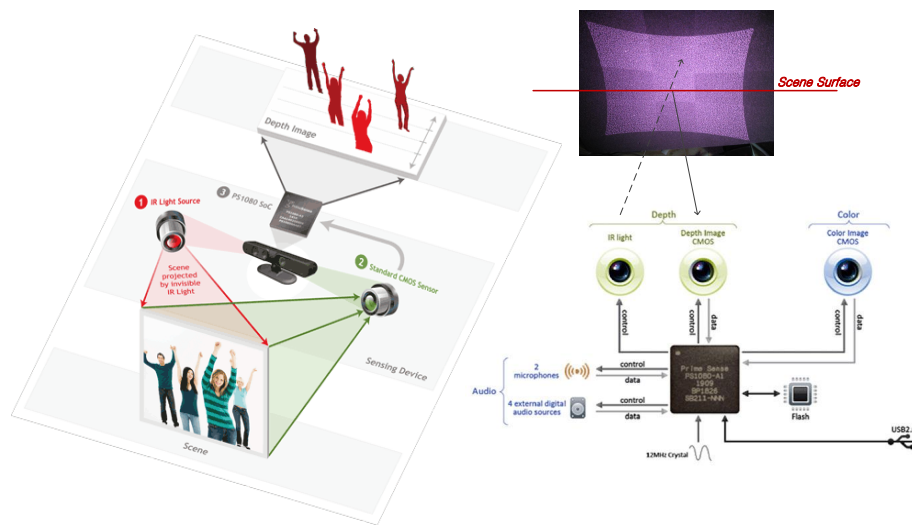


Figure 2.26: The depth image is constructed by triangulation from the IR image and the projector. The laser source projects a pattern of speckles onto the scene and this pattern is captured by the IR camera.

intrinsic parameters of the RGB camera must be taken into account, such as the focal length, principal point offsets and the lens distortion.

It should be noted that even though the linearity of the sensor can achieve a measuring range in approximately $0.5m$ to $9.7m$ using the OpenNI framework, it has been established that the further the points are located, the lower is the accuracy of the sensor. Even if the sensor setup is stationary it has been observed that the value of a given depth pixel fluctuates between a small number of bit levels over time. Thus, the error depth measurements as well as the depth resolution increases quadratically in maximum range of 5 meters. Therefore, better results are acquired within $1-3m$ distance since in large distances the quality of the data is degraded by the noise and low resolution of the depth measurements. A more detailed analysis of accuracy in terms of error estimation can be found in (Khoshelham, 2011; Khoshelham and Elberink, 2012; Andersen et al., 2012; Smisek et al., 2013).

Finally, it is worth mentioning that Microsoft has announced the new generation of Kinect for Windows, Kinect 2.0. Among its main contributions in respect to the old device should be underlined: an accuracy definition,

further depth capture and improvement in voice recognition. Appendix B offers a comparison between some depth sensors available in the market.

2.3.3 Monocular System

Unlike a stereo system, a monocular system is composed of a single camera that captures one image of the scene each time. In addition, monocular systems can be classified as *rigid* or *deformable* in response to the characteristics of the object to be tracked.

2.3.3.1 Rigid Objects

Rigidity implies that the size and shape of the target object do not change over time, even when external forces are applied. This simplifies calculations, since the shape of the object is not treated as an unknown factor (see Section 2.3.3.2). This also allows taking advantage of the prior knowledge of the object in order to cope with the loss of information that is involved when working with a single camera. There are many ways to represent this knowledge (Lepetit and Fua, 2005), which can be classified as a *marker* and *markerless* tracking.

Marker Tracking

A marker tracking system adds known and easily identifiable patterns (called *markers*, *fiducials* or *landmarks*) to the scene. Although there are different patterns, such as circular, planar, or colour-coded-based; black squares printed on a white sheet (Kato and Billingham, 1999; Schmalstieg and Wagner, 2007; Cawood and Fiala, 2008) are a widely used marker due to its good performance and low cost of manufacture.

All markers share the same property of being easy to detect in the image. The information they provide is extracted and used to calculate the camera pose, making it a robust, precise and real-time tracking system. In the case where multiple markers are detected, each one is distinguished by the unique identifier it codifies. Assuming that a single square marker has been added to the scene, its 2D image position (\vec{m}_i) is determined by finding square shapes in the camera image that have a similar inner pattern. Moreover, the world coordinate system is centred on the marker (indeed, the world coordinate system is usually centred on the target object), so its 3D coordinates (\vec{M}_i) are known. Considering

that the marker lies on a plane and the camera is calibrated (K is known), the camera pose is recovered from four $\vec{m}_i \leftrightarrow \vec{M}_i$ correspondences that do not form triplets of collinear points (DLT (Hartley and Zisserman, 2003)). More precisely, the correspondences of the four corners of the marker are used to compute the camera's extrinsic parameters (see Figure 2.27). This is a very fast and accurate technique for building AR applications, which can even be executed on mobile platforms (Schmalstieg and Wagner, 2007).

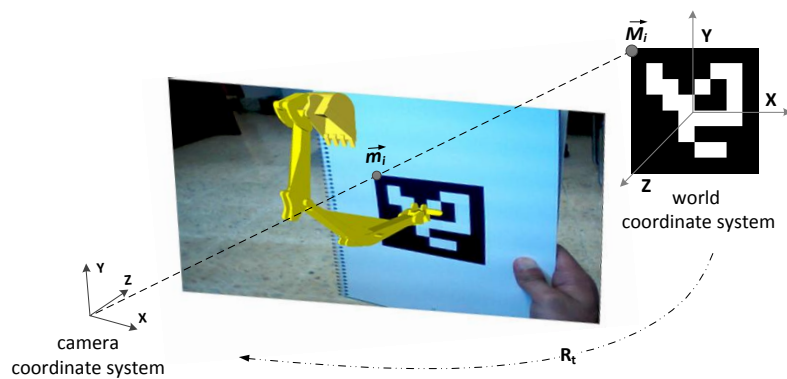


Figure 2.27: Marker tracking system overview. A square marker of the ARToolKitPlus library (Schmalstieg and Wagner, 2007) is shown. A virtual yellow shovel is superimposed in the image.

A drawback of marker tracking systems is that the environment requires adaptation, which is not always possible. Marker occlusion is another shortcoming, as the system fails even if the marker is only slightly occluded, producing an undesirable effect on users, who lose the sense of realism. Because of this, several solutions have been proposed to overcome the marker occlusion problem such as placing multiple markers in the scene (Kato and Billinghurst, 1999), new marker designs that support a certain degree of occlusion (Wagner et al., 2008), tracking the bounding box of the marker (Álvarez and Borro, 2009) or an incremental tracking of the feature points that lie on the marker (Malik et al., 2002).

Markerless Tracking

A markerless tracking system does not add artificial markers to the scene, rather it takes advantage of the visual cues that occur naturally in the scene. Depending on whether or not the scene geometry is known, markerless tracking is divided into two groups (Teichrieb et al., 2007): *Model-Based* and *Structure from Motion*.

1. Model-Based

Model based techniques store prior knowledge of the scene in a 3D model, which is available before starting the camera tracking. The 3D model could be represented by its simple 3D geometry (Drummond and Cipolla, 2002), or by a more detailed description that includes the geometry and the texture of its surface (Vacchetti et al., 2004). In both cases some visual cues belonging to the 3D model are tracked along a sequence of images to estimate the camera's extrinsic parameters. Based on the knowledge of the position of these visual cues in the previous frames, two different techniques are distinguished: *Frame-to-Frame tracking* and *Tracking-by-Detection*.

Frame-to-Frame tracking (FtF), also known as *recursive tracking* or *incremental tracking*, uses the previous pose and temporal coherence to estimate the current one. More precisely, the current 2D image locations of the target visual cues are estimated (\vec{m}'_t), so that the 3D motion of the model between two consecutive frames is recovered from the 2D displacements of these points. It can be distinguished from *FtF based on predictor* and *FtF based on optical flow*.

FtF based on predictor techniques (Drummond and Cipolla, 2002; Barandiarán et al., 2007) combine the previous camera with one predictor (LaViola, 2003; Salih and Malik, 2011) to get an estimation of the current camera pose. Notice that, a predictor stores the camera pose estimated in the previous frames to feed a transition model and provide an estimation of the current camera pose according to the trajectory followed by the camera. In this way, the predicted camera pose is used to project the 3D visual cues and obtain \vec{m}'_t . An edge tracker is the most common example of this alternative. It considers a set of 3D points that lie on the 3D model edges, called control points, and performs a simple loop. Visible 3D edges are detected and projected onto the current camera image using an occlusion query and the previous camera pose (R'_t), respectively. Then,

for each projected 3D edge a set of 2D edge samples (\vec{m}'_i) are selected (control points, whose 3D coordinates \vec{M}'_i are known, $\vec{m}'_i = KR'_i\vec{M}'_i$), and a local search along the edge normal (\vec{n}_i) is carried out to establish the presence of these samples in the current camera image (\vec{m}_i) (see Figure 2.28).

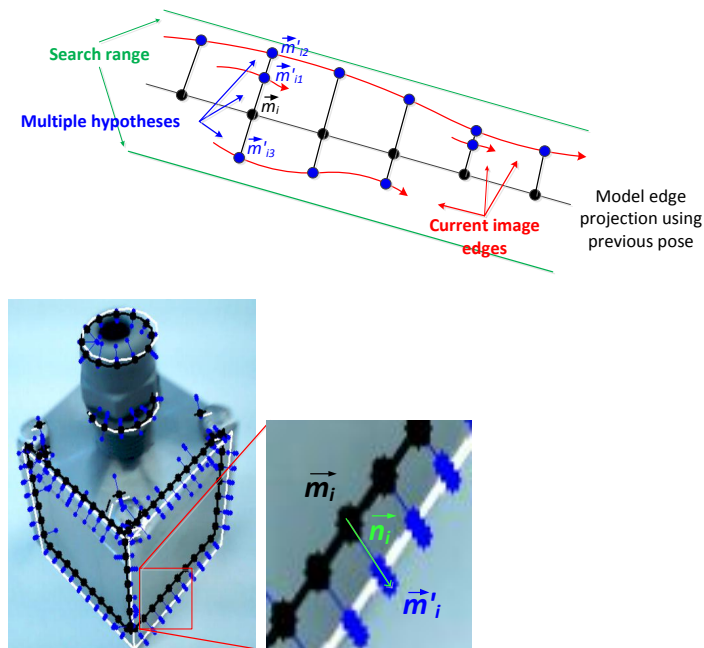


Figure 2.28: 2D displacements of control points. They are used to compute the motion of the model between the previous (black wireframe) and current frames (white wireframe). Multiple hypotheses for each control point are also shown.

There are many alternatives to guide the search for the control points, such as correlating control points that lie on the same edge in order to treat them as a single primitive (stronger movement constraints) (Armstrong and Zisserman, 1995), or considering multiple displacement hypotheses for each control point (see Figure 2.28) (Vacchetti et al., 2004), which is a more robust method against cluttered scenes. If an image edge pixel is detected in the vicinity of a control point, then a positive correspondence

is found. This procedure can thus provide multiple match hypotheses for each control point (see Figure 2.28).

Finally, the camera pose of the current frame (R_t) results from the minimization of the distances between selected image edge pixels and projected control points (minimization of the reprojection error):

$$R_t = \operatorname{argmin}_{R_t} \sum_i P_{Tuk}(\min_j \Delta(\vec{m}_{ij}, \vec{m}'_i)), \quad (2.16)$$

where $\vec{m}_{ij} = KR_t\vec{M}'_i$ represents the hypothesis j for the control point i , $\Delta(\vec{m}_{ij}, \vec{m}'_i) = \|(\vec{m}_{ij} - \vec{m}'_i) * \vec{n}_i\|$, and P_{Tuk} is the Tukey function that reduces the influence of outliers using the M-estimator approach:

$$P_{Tuk}(x) = \begin{cases} \frac{c^2}{6} \left[1 - \left(1 - \left(\frac{x}{c} \right)^2 \right)^3 \right] & \text{if } |x| \leq c \\ \frac{c^2}{6} & \text{otherwise} \end{cases}, \quad (2.17)$$

where c is a bandwidth parameter that is usually set proportionally to the standard deviation of the estimation error. This way, when $|x|$ is larger than c , the Tukey estimator becomes flat and large residual errors have no influence at all.

FtF based on optical flow approaches (Bleser et al., 2005; Platonov et al., 2006), in contrast, use the 2D image positions of the visual cues in the previous image and the intensity differences between two consecutive frames to provide \vec{m}'_i .

Usually a local search is also performed in the vicinity of each \vec{m}'_i to find the correct position of the visual cue in the current image (\vec{m}_i). This search is based on the similarity (shape, texture, etc.) between the reference visual cues and those candidates detected in the current image. Once \vec{m}_i is estimated for a sufficient number of visual cues, the camera pose is computed solving Equation 2.15.

Due to their recursive nature, FtF methods suffer from drift (error accumulation) and they are sensitive to fast camera movements. Nonetheless, *Particle Filters (PF)* are an extension of FtF methods that support rapid camera movements, since they belong to the family of *Sequential Monte Carlo (SMC)* methods and are robust against non-static scenes in which multi-modality is likely (Arulampalam et al., 2002a). The

key idea of PF is to represent the required posterior distribution of the camera motion by a set of random samples with associated weights and to compute estimates based on these samples and weights (see Figure 2.29). They are divided into two stages that are executed iteratively: *particle generation* and *particle evaluation*. In the particle generation step, the previous camera pose is perturbed to generate multiple camera pose candidates of the current frame (particles). The particle evaluation step, on the other hand, is responsible for assigning a weight to each particle and selecting the correct one. Both robustness and computational cost are proportional to the number of particles, so this parameter is critical in order to get good results within reasonable computation times. An in depth discussion of PF can be found in (Arulampalam et al., 2002a; Salih and Malik, 2011).

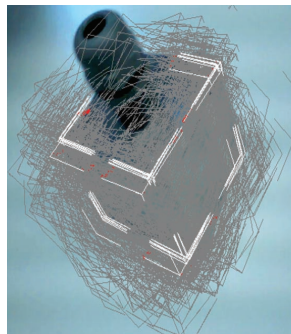


Figure 2.29: Tracking using a PF. Samples are drawn in grey. White indicates particles with higher weight.

Additionally, FtF methods require an initial pose to start the recursive process, which is obtained manually or using a TbD method.

Tracking-by-Detection (TbD), sometimes called *3D object recognition*, faces the challenge of computing the camera pose without requiring a tracking history, so it is used for automatic initialization and recovery from failures. It tries to match some reference visual cues with those detected in the entire image, without limiting the search to a local area imposed by the previous state. Multiple 2D views of the 3D model (*keyframes*) are taken from different positions and orientations during an offline training phase to build a database of 3D visual cues (see Figure 2.30). Each 3D visual cue is characterized by a set of 2D views (\vec{m}'_i), which try to simulate the

online conditions of the 3D visual cue and improve the matching quality between reference (\vec{m}'_i) and detected (\vec{m}_i) visual cues. Considering that each \vec{m}'_i is a 2D view of a 3D visual cue \vec{M}_i , Equation 2.15 is solved as long as sufficient $\vec{m}'_i \leftrightarrow \vec{m}_i$ correspondences are computed. The following paragraphs describe different ways of obtaining these correspondences, which use feature points, contours or geometric properties.

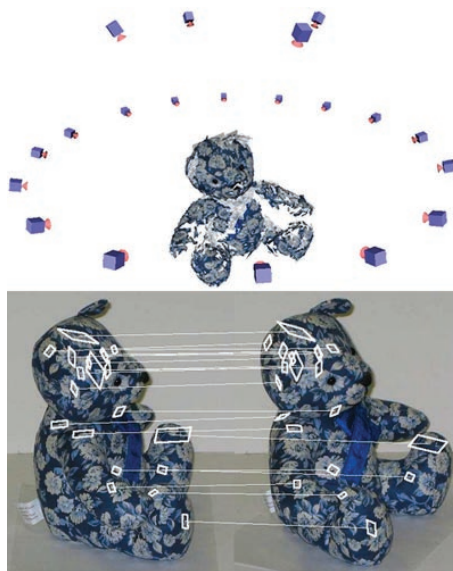


Figure 2.30: 3D object recognition based on appearance (Rothganger et al., 2006). Some keyframes are generated during an offline phase (top) to match reference features (bottom-left) with those detected in the current image (bottom-right).

(Lowe, 2004; Rothganger et al., 2006; Bay et al., 2008) use feature descriptors to determine a set of 2D-3D matches that define the 3D pose of the model. Some 2D views of the model are selected as representative during the training step. These views are processed, and features that lie on the surface of the model are extracted. For each feature its 3D position and descriptor is calculated and used to train a classifier that will be responsible for matching during the online phase. These descriptor-based techniques combined with geometric constraints improve precision and efficiency (Hinterstoisser et al., 2007). All of them achieve good results and high frame rates, but robustness decreases for untextured objects. These types of solutions are oriented to rich textured objects, since descriptors

that can be extracted from their surface are more discriminative, thereby improving the matching results.

Other approaches use the contours of the model to get a positive match (Ulrich et al., 2003; Holzer et al., 2009; Hinterstoisser et al., 2010). They are based on the shape of the model, and therefore, they are suitable for untextured models. In the training step, 2D views of the model are processed to extract the corresponding edges, transforming the online problem to 2D-2D edge pattern matching. Although some of the works cited only address the problem of 2D-2D edge matching (Ulrich et al., 2003), they can be extrapolated to 3D recognition by using a training step similar to that mentioned above. All of them require the acquisition of real 2D views, so they are configured for specific light conditions and edge responses extracted from the real training images. This is not the case with (Ulrich et al., 2009; Stark et al., 2010), which use virtual training to build 2D artificial views of the object, resulting in greater generality.

The third group of alternatives is based on geometric features (Lamdan and Wolfson, 1988; Costa and Shapiro, 2000; Sehgal, 2003). Distinctive 3D geometric features of the model are extracted in the offline phase, and their geometric relationships are indexed in a database. During the online phase, image input features are detected and compared to the database. Consistent matches receive a vote, and the solutions with the highest number of votes are selected for further processing. Generally, these features provide poor distinctiveness, so an input feature generates multiple matches, increasing the computational effort. Due to their computational requirements, they are more popular for non-time-critical applications, such as 3D pose recovery in 3D scenes (Drost et al., 2010). They have also been optimized using the correspondences specified by the user (Franken et al., 2005), but this has the disadvantage of requiring user intervention.

There are also more sophisticated alternatives (Álvarez et al., 2011) that combine both FtF and TbD in order to exploit the best properties of each method. Under this premise, the fast and accurate FtF is executed whenever possible, while the robust TbD acts as a recovery mode.

2. Structure from Motion

In *Structure from Motion (SfM)* approaches the camera movement is estimated and the 3D reconstruction of the scene is also performed (Longuet-Higgins, 1981), i.e., they estimate both R_t and \vec{M}_t . Some visual cues (usually features) are tracked during a video sequence so that the 2D position of each feature is stored for each frame. Thus, different views of the same visual cue are *triangulated* in order to obtain its 3D coordinates (see Figure 2.31). In fact, the camera pose and the structure of the scene is recovered given a minimum of two images related by their visual correspondences. For that purpose, the multiple view geometry theory is used (Hartley and Zisserman, 2003), similar to that presented for stereo systems (see Section 2.3.1). It is noteworthy that these solutions extract all the information from previous frames, without requiring markers or knowledge about the target model. There are two refinements of the SfM algorithm: *batch optimizations* and *recursive estimations*. The first option offers more accuracy, while the second is less time consuming.



Figure 2.31: 3D scene reconstruction from multiple image views.

Batch optimizations minimize a cost function that refers to the difference between the projections of points in an unknown 3D scene and their known image measurements over the video sequence:

$$\operatorname{argmin}_{R_t^j, \vec{M}_i} \sum_j \sum_i \left\| \vec{m}_i^j - K R_t^j \vec{M}_i \right\|, \quad (2.18)$$

where R_t^j is the camera pose of the frame j , \vec{M}_i represents the 3D coordinates of the visual cue i and \vec{m}_i^j corresponds to the 2D image coordinates of visual cue i in the image j . Note that \vec{M}_i values remain unchanged over the entire video sequence, since the reconstruction of a rigid body is assumed.

The *Bundle Adjustment* (BA) technique (Triggs et al., 2000) is used to jointly optimize the 3D structure and the motion parameters (see Equation 2.18). In their first implementations, the entire video sequence was used for the optimization (Hartley, 1994), making them impractical for real time. Recently, (Klein and Murray, 2007) uses a local BA over the last five selected keyframes and parallel techniques (one thread computes the mapping and the other thread only performs the tracking) to make them suitable for real-time applications.

Recursive estimations are probabilistic methods that have been extensively used in the robotics community in order to address the *Simultaneous Localization and Mapping* (SLAM) problem. They compute an online reconstruction of the scene using recursive Bayesian estimators, formulating the problem as a *state-space model*. The state-space model is described by a state model (see Equation 2.19), which is associated to the transition over the time of the 3D structure and motion parameters, and an observation model (see Equation 2.20), which is related to the measurements that condition the transition.

$$\vec{x}_{j+1} = f(\vec{x}_j, \vec{u}_j), \quad (2.19)$$

$$\vec{y}_j = h(\vec{x}_j, \vec{v}_j), \quad (2.20)$$

where \vec{x}_{j+1} is the state vector containing the camera pose and the 3D reconstruction, \vec{u}_j represents the uncertainty of the transition model f , \vec{y}_j is the vector containing image measurements and \vec{v}_j represents the measurement noise of the observation model h .

Provided that a set of observations $Y_j = \{\vec{y}_1, \dots, \vec{y}_j\}$ is available at any instant j and that the Equation 2.19 describes a Markov chain, the problem

can be formulated as the conditional probability of the current state given the entire set of observations: $P(\vec{x}_j|Y_{j-1}) = \int P(\vec{x}_j|\vec{x}_{j-1})P(\vec{x}_{j-1}|Y_{j-1})d\vec{x}_{j-1}$. Using this formulation the parameters of the state space-model are expressed by their probability distributions.

Assuming that f and h are linear functions, the optimal estimation (in a least squares sense) is given by the *Kalman Filter (KF)* (Bishop and Welch, 2001). Otherwise, in the case that they are non-linear, the *Extended Kalman Filter (EKF)* provides a similar framework by *linearizing* the system using the Taylor expansion of f and h (Davison, 2003). However, all variants of the KF assume that the model follows a Gaussian distribution, so more generic approaches such as PF (see Section 2.3.3.1 Model-Based) have been adapted. Following this idea (Sánchez et al., 2010b) presents an efficient implementation based on the GPU, where all the calculations, the 3D reconstruction (Sánchez et al., 2010a) as well as the camera tracking, are performed by the GPU pipeline, making it feasible for real time.

2.3.3.2 Deformable Objects

In contrast to rigid objects, the size and shape of a deformable object change when external forces are applied. This makes the general problem described in Equation 2.15 more difficult, since \vec{M}_i values change in each frame and are considered as an additional unknown (note that unlike SfM approaches, here \vec{M}_i values are not the same for all frames). Moreover, in the optical tracking context, external forces are given by the attraction force imposed by the visual correspondences, which can be computed using similar methods to those explained for rigid bodies (see Section 2.3.3.1). Deformable methods focus on how the deformation of the object is represented and constrained. They can be classified into *physics-based*, *learning-based*, *template-based* and *non-rigid structure from motion*.

Physics-based

The physics-based deformation models are the earliest approaches to modelling the behaviour of an object according to the physical laws that govern it. The key idea is to capture, in a generic way, a priori knowledge about the physical properties of the object to achieve an approximation of physical reality.

(Kass et al., 1988) was an approach that promoted physics-based models oriented to 2D shape recovery. Subsequently, the physics-based models have been also used for 2D surface registration (Bartoli and Zisserman, 2004; Pilet et al., 2005; Zhu and Lyu, 2007; Uchiyama and Marchand, 2011). In addition to 2D surfaces, these models have also been used with 3D surfaces. At first, 3D reconstructions were performed relatively simply (Terzopoulos et al., 1987), using the generalized cylinder model as abstraction of elongated shapes exhibiting axial symmetry. The complexity was then increased through the use of deformable superquadrics (Metaxas and Terzopoulos, 1991; Terzopoulos and Metaxas, 1991), triangulated meshes (Fua and Leclerc, 1995), *Thin-Plate Splines (TPS)* (McInerney and Terzopoulos, 1993; McInerney and Terzopoulos, 1995) and balloons forces (Cohen and Cohen, 1991; Cohen and Cohen, 1992; Sharf et al., 2006) (see Figure 2.32).

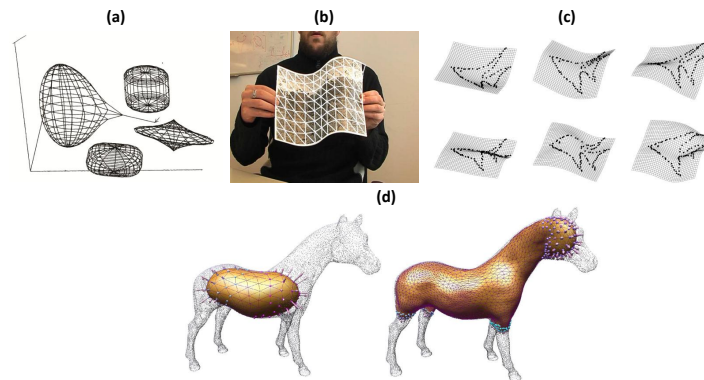


Figure 2.32: (a) Deformable superquadrics (Terzopoulos and Metaxas, 1991), (b) triangulated mesh (Salzmann et al., 2005b), (c) simple example of a fish shape transformation using TPS (Donato and Belongie, 2002) and (d) balloon models (Sharf et al., 2006).

The 3D reconstruction of a surface is a difficult problem that requires too many *Degrees of Freedom (DoF)*. There are two main approaches to solving the complexity of this problem: the inclusion of regularization terms and the *Finite Element Method (FEM)* formulation.

The goal of the first approach is to minimize a global energy $\epsilon(S)$ that is composed of an internal $\epsilon_D(S)$ and an external $\epsilon_C(S)$ energy. The former represents the physical properties of the surface. Essentially, it focuses on controlling the uniformity of the mesh, i.e., a term that regulates the smoothness

of the mesh. The second, in turn, focuses on the image data. It is an energy that exerts a force so that the mesh tends to deform in line with the detected and matched visual cues of the image. In (Fua and Leclerc, 1995), for example, the surface of the model is represented as a hexagonal mesh which is deformed by minimizing the energy that is decomposed into a lineal combination of two components:

$$\varepsilon(S) = \lambda_D \varepsilon_D(S) + \varepsilon_C(S), \quad (2.21)$$

where λ_D is a scalar to balance the weight of each energy and S is the state vector that contains the coordinates of the mesh and their associated visual correspondences.

The FEM formulation offers an alternative approach to this problem. Here, the model is represented by a discrete set of elements such as segments, triangles or tetrahedrons (linked by their nodes) governed by mass, damping and stiffness constraints. The following discrete equation represents the motion of the deformation:

$$M\ddot{u} + D\dot{u} + Ku = f, \quad (2.22)$$

where u is the unknown vertex displacement, \dot{u} and \ddot{u} its corresponding first and second derivatives respectively, f is the nodal data forces, K the stiffness matrix, D the damping matrix and M the mass matrix. It can be noticed that all of these matrices are sparse and symmetrical.

In this context, the non-linear FEM was used for accurate surgery simulations (Picinbono et al., 2000) and for large deformations in the animation area (Irving et al., 2004b). Similarly, in (Tsap et al., 1998a) the incomplete or missing information about the geometry or material properties of the object is solved by using non-linear FEM. Unfortunately, the drawback of this technique is that it requires thorough knowledge. This is why most 3D reconstruction research focuses primarily on simple deformations (Tsap et al., 1998b). Compared with the previous approach, the FEM formulation achieves more accuracy, but is more difficult to implement and requires more computational resources.

Subsequently, in conjunction with the previous, modal analysis (Delingette et al., 1991) was introduced to reduce the high dimensionality appearing in the FEM formulation and to simplify the problem. It consists in estimating the surface deformation as linear combinations of deformation modes (Pentland and Sclaroff, 1991; Nastar and Ayache, 1996) (also referred as *basis shapes*). Computationally

it is a good approach, but it is restricted to objects with very smooth deformations. To overcome this limitation, more accurate but also more complex non-linear models have been proposed (Tsap et al., 1998a; Bhat et al., 2003).

Finally, although the physics-based methods have achieved great success and high levels of accuracy (Bickel et al., 2009), the drawbacks extracted from the results should be highlighted. Firstly, in referring to the properties of the surface material, they are usually unknown. Indeed, even with knowledge of the physical parameters and surface materials, the physics-based methods only achieve relatively accurate approximations for small deformations. Secondly, to actually implement accurate large deformations, physics-based models require a complex design subject to a difficult task such as the use of more sophisticated finite-element methods that imply higher computational cost. At the same time, this methodology is prone to instabilities of convergence, which are often difficult to optimize.

Learning-based

As discussed above, physics-based models arise from the idea of applying physical behaviour to objects. However, the parameters of the physical properties of an object are not always known. Thus, learning-based models (also referenced as *statistical-based models*, *machine learning models* or *example-based models*) are a great alternative to the latter, since they infer the behaviour-model from a set of available examples (hand labelled or generated through 3D laser scans) (see Figure 2.33).

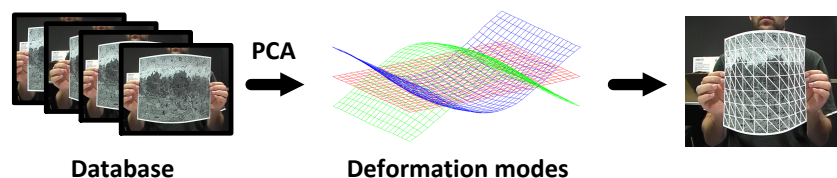


Figure 2.33: Learning-based 3D shape recovery. Original source (Salzmann et al., 2007c).

More precisely, this methodology applies a statistical dimension reduction technique to the training data in order to create a database of representative shapes, i.e., to learn a low-dimensional model. As in the modal analysis technique of the physics-based methods, the surface deformations are expressed as linear

combinations of deformation modes such as those shown in Figure 2.35(c). Nonetheless, these modes are obtained through the training process instead of using stiffness matrices. It is therefore possible to capture a larger number of movements, provided that they are available in the training data set. Depending on the reduction technique that is carried out, learning-based models can be classified into *linear* and *non-linear* methods:

Linear methods have been the most frequently used. The concept is to assume a linear mapping from a low-dimensional manifold to the high-dimensional data space. First, a database from 3D meshes is generated, and then the shape model is learned to dimensionality reduction by *Principal Component Analysis (PCA)*⁷ (Jolliffe, 1986), retaining the most significant components. For example, let's say that the non-rigid surface is represented as triangulated mesh with n_v vertices expressed as x vector. The shape deformations are modelled as a weighted sum of n_m deformation modes, represented by a S vector:

$$x = x_0 + \sum_{i=1}^{n_m} c_i s_i + E = x_0 + S c + E, \quad (2.23)$$

where x_0 is the mean shape and c represents unknown weights that define the current shape. The S matrix is obtained by the PCA and its columns are taken to be the eigenvectors of the covariance matrix. Due to error of fit to the data, the error is accumulated in the matrix E .



Figure 2.34: Morphable face models. Matching a morphable model automatically to a single sample image, the model is rendered changing its facial attributes, in this case, being forced to smile (by (Blanz et al., 2003)).

⁷The data is projected onto its eigenvectors, and these projections, sorted with respect to the eigenvalues, span a new orthogonal space. In the first dimensions, maximal variance of the initial data is encoded, while dimensions with small eigenvalues are most likely to represent noise.

Active Appearance models are the first works making use of these techniques, to use both 2D (Cootes et al., 1998; Zhu et al., 2006) and 3D (Matthews and Baker, 2004) models. Following these, Morphable Models (Blaiz and Vetter, 1999; Blaiz et al., 2003) shown in Figure 2.34 apply the same idea to reconstruct deformable faces. Recently, new approaches that promote the same philosophy of reducing the dimensionality have been proposed (Salzmann et al., 2005a; Salzmann et al., 2005b). These solutions represent the deformation of a mesh by varying angles between facets; leading to a lower dimensional model in comparison to those alternatives based on vertex coordinates (see Figure 2.35). The main drawback of this new formulation is that it imposes more restrictions on mesh deformations, so that the representation may not be as close to the reality as one would like.

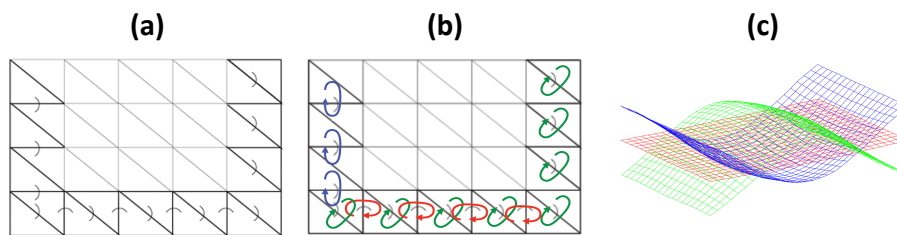


Figure 2.35: Database of feasible shapes. A 3D deformation representation of a mesh by varying angles between facets (a). The red mesh in (b) is the average mesh, x_0 and the other two are obtained by taking a single c_i to be non-zero. A positive value of c_i yields the green mesh and a negative one the blue mesh. Original source (Salzmann et al., 2007c).

On the other hand, *non-linear methods* are used in those cases where linear methods do not fit the training data well and generate poor quality models. *Kernel-PCA (KPCA)* (Platt, 1998) belongs to this type of non-linear dimensionality reduction technique, which uses a kernel function to map the data-space to another dimensional space (latent-space) where the data can be linearly modelled. However, for KPCA-like techniques is not obvious how to project back from the latent-space to the data-space (pre-image problem). Therefore, the *Gaussian Process Latent Variable Model (GPLVM)* (Titsias and Lawrence, 2004) has emerged as a valid alternative. It is a non-linear probabilistic PCA that directly defines a mapping from the low-dimensional representation to the high-dimensional one:

$$x = \sum_{i=1} w_i f_i(c) + \epsilon, \quad (2.24)$$

where f_i are the non-linear functions of the low-dimensional representation of the latent variable c . A prior knowledge is used to generate a probability distribution (more specifically a multivariate Gaussian distribution) from which w_i values are extracted.

This technique also achieves a good performance in human motion (Urtasun et al., 2005), but involves a complex objective function that may be difficult to optimize. Other approaches like (Salzmann et al., 2008b) use the non-linear statistical technique as well, but based on a hierarchical strategy, combining local deformation surfaces to generate a global version representing the shape.

Generally, linear methods impose smoothness constraints and are not always accurate. Non-linear methods overcome this problem but require large training data. In fact, this is the biggest issue of the learning methods, since the generation of the training database is a difficult and time-consuming process. Additionally, training data has to be created specifically for each model, which implies that each object must be processed individually, even if the two objects are composed of the same material. These difficulties have all contributed to the unpopularity of this type of model.

Template-based

Template-based models are well known methods that use a *reference* image (template) to overcome the 3D shape reconstruction of non-rigid surfaces (Zhu and Lyu, 2007; Zhu et al., 2008; Salzmann et al., 2008a; Pilet et al., 2008; Moreno-Noguer et al., 2009; Salzmann and Fua, 2009; Salzmann and Fua, 2011; Perriollat et al., 2011; Uchiyama and Marchand, 2011). More specifically, the 3D shape of a surface in an *input* image is recovered from a set of visual correspondences between the *input* image and the *reference* image in which the 3D shape of the surface is known (see Figure 2.36).

These methods require the execution of two steps: *image registration* and *shape inference*. The first step is responsible for detecting 2D-3D correspondences between the input and reference images, while the second phase tries to adjust the deformation model to those correspondences according to a set of constraints.

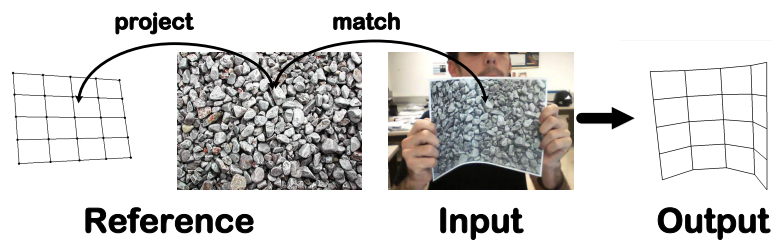


Figure 2.36: Shape recovery from 3D-to-2D correspondences. Detect and match feature points, relate the features to 3D points given in barycentric coordinates between reference configuration and input image. Original source (Salzmann and Fua, 2009).

1. Image registration

The main goal of this step is to obtain the visual correspondences (usually features) that relate the input and reference images. First, feature points are detected in the reference image. Furthermore, since in this image the 3D shape of the surface is known, the 3D coordinates of the features can be extracted by doing the corresponding *back-projection* (detecting for each feature the facet of the surface it belongs to and defining its location in terms of barycentric coordinates). Subsequently, feature points are detected in the input image and matched against those extracted in the reference image. Thus, several 2D-3D correspondences are obtained, whose reprojection error is minimized in order to recover the 3D shape. The idea is to find the new positions of the mesh vertex that minimize the distances between the input 2D features and the projection of the reference 3D features in the image. Based on how these correspondences are obtained, the image registration can be solved by *pixel-based methods* (Gay-Bellile et al., 2007), *feature-based methods* (Torr and Zisserman, 2000) or *hybrid methods* (Zhu et al., 2009):

- Pixel-based methods, also known as *direct* methods, use the difference in intensities between two images to calculate the 2D-3D correspondences (Cootes et al., 1998; Sclaroff and Isidoro, 2003; Matthews and Baker, 2004; Baker, 2004). These methods are highly correlated with FtF tracking techniques, which are based on temporal consistency, in the sense that the previous frame bounds and simplifies the processing of the next frame. Optical flow (Decarlo and Metaxas, 2000; Bartoli and Zisserman, 2004) is one of the most

popular techniques implementing these concepts. Due to mentioned above, pixel-based methods are viewed as an extension of the Lucas-Kanade algorithm (Baker and Matthews, 2004).

This technique has been used with different deformable surfaces such as human faces (DeCarlo and Metaxas, 1998), sheets of paper (Gay-Bellile et al., 2007) or clothing (Hilsmann and Eisert, 2009). It depends largely on a good initialization to avoid getting stuck in local optima. Additionally, these estimations cannot be used indefinitely without any correction, as the errors in the estimations are integrated over time, i.e., this method is prone to drift.

- Feature-based methods attempt to detect the presence of the same feature in two images in order to get a potential set of 2D-3D correspondences (Bookstein, 1989; Salzmann et al., 2007c). These methods are highly correlated with TbD techniques, which are based on appearance. Generally, they try to minimize the difference between the feature descriptors of the input and reference images. The SIFT method (Lowe, 2004; Mikolajczyk and Schmid, 2005; Lepetit and Fua, 2006; Belongie et al., 2002) is one of the most popular feature descriptors used for this purpose.

Contrary to pixel-based methods, feature-based methods do not require initialization. Nonetheless, they have to overcome several challenges such as working with few feature points (textureless surfaces), dealing with outliers, or handling self-occlusions. Some works like (Chui and Rangarajan, 2003) propose a solution for outliers, but at the cost of working with slower algorithms.

- Hybrid methods combine both pixel and feature methods to emphasize their advantages (Fleet et al., 2000; Johnson and Christensen, 2002; Georgel et al., 2008). The work presented in (Pizarro and Bartoli, 2012), for example, provides more robustness to its feature-based implementation by using a point-based warp estimation method that prevents the warp folding in the presence of self-occlusions.

2. Shape inference

This step is responsible for reconstructing the deformation model using the 2D-3D correspondences found in the previous step. Nonetheless these correspondences show certain ambiguities and do not achieve an optimal reconstruction. They only serve as a weak constraint to move the vertices

of the mesh along the projection ray. Indeed, a change in the exact location along the projection ray only results in minor reprojection errors. Accordingly, it is no longer seen as a viable alternative if additional knowledge constraints are not introduced. Therefore, this second step solves the deformation model based on 2D-3D correspondences and some additional constraints (Gumerov et al., 2004; Salzmann and Fua, 2011) (see Figure 2.37). This idea can be expressed mathematically by the following conceptual equation:

$$\underset{Rt, M_i}{\operatorname{argmin}} \sum_i \left\| \vec{m}_i - KRt\vec{M}_i \right\| \quad (2.25)$$

subject to *constraints*,

where \vec{m}_i is the 2D image position of the visual cue i , \vec{M}_i are their corresponding 3D coordinates and K represents the intrinsic parameters that describe the characteristics of the camera (focal length, skew, etc.), and which can be extracted through a camera calibration process.

Regarding constraints, an inextensible surface reconstruction can be carried out using deformation modes in conjunction with local rigidity constrains (Salzmann et al., 2008a; Ecker et al., 2008; Salzmann and Fua, 2009; Perriollat et al., 2011), while in conjunction with shading constraints stretchable surfaces can be reconstructed (Moreno-Noguer et al., 2009). None of these approaches retrieve the camera pose, since they assume that the deformation modes are aligned to the camera or directly recover the shape in an unknown pose. However, more recent approaches such as (Sánchez-Riera et al., 2010; Moreno-Noguer and Porta, 2011) deal with this limitation by simultaneously obtaining the deformation and the camera pose.

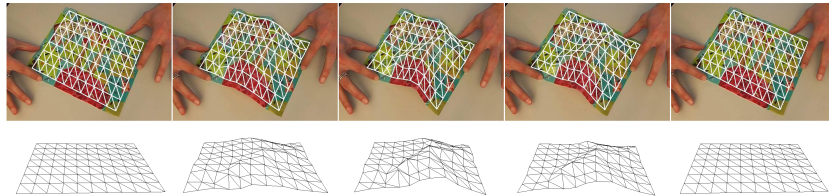


Figure 2.37: Reconstruction results of a well-textured sheet of paper (by (Salzmann et al., 2007b)).

A brief summary of the most frequently used constraints is presented below:

- Time constraints. Through the time constraint, the movement between two frames does not vary too much. For example, (Salzmann and Fua, 2009) restricts the distances of the edges to be the same at t and $t + 1$ instants (see Figure 2.38).

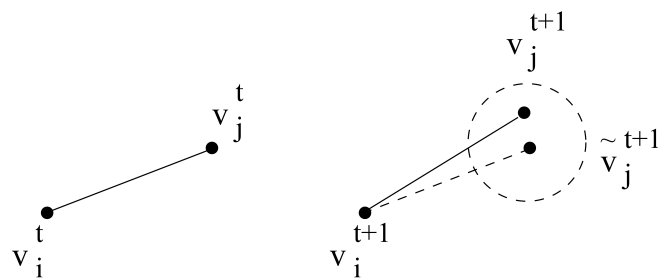


Figure 2.38: Time constraint. The distance between the vertex v_i and v_j is predicted to be the same at time t and $t + 1$ (by (Salzmann et al., 2007a)).

Despite the good results achieved by the use of time constraints, they require high computational resources when several constraints are imposed. Additionally, they suffer from error correspondences and require a good initial estimation.

- Distance constraints The distance constraint consists in assuming that Geodesic distances do not change (Brunet et al., 2011). However, as illustrated by Figure 2.39 Geodesic distances are expensive to use, so there are approaches that use Euclidean distances as an approximation (Shen et al., 2010).

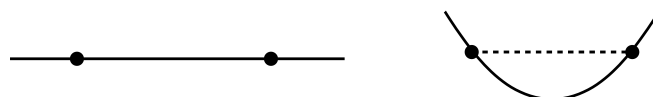


Figure 2.39: Distance differences. In the presence of sharp folds, Geodesic distances (right) remain constant. There are other approaches that use Euclidean distances (left) rather than Geodesic ones (by (Salzmann and Fua, 2009)).

There are alternative works that rely on inextensibility constraints, where the distance between feature points is used instead of the distance between the vertices of the mesh. Following this idea (Ecker et al., 2008) proves a solution under orthographic projection, while (Perriollat et al., 2011) does so under full perspective projection.

This type of constraint is oriented to objects whose material cannot stretch practically, such as sheets of paper (Gumerov et al., 2004; Liang et al., 2005). Moreover, the use of inextensibility assumptions implies small deformations.

The integration of these constraints precludes the use of least-squares techniques. Therefore, other mathematical tools must be used such as the *Second Order Cone Programming (SOCP)* (Boyd and Vandenberghe, 2004), convex optimization (Kahl, 2005; Salzmann and Fua, 2009; Shaji et al., 2010), quasi-convex optimization (Ke and Kanade, 2007), or closed form solutions (Salzmann et al., 2008a; Moreno-Noguer et al., 2009).

The problem with template-based methods is that they suffer from bad correspondences which normally cause erroneous reconstructions. Some studies, such as (Salzmann et al., 2008a), do not return a unique solution but a set of representative solutions and choose the best one by using, for example, shading or movement (related with temporal constraints) information (Moreno-Noguer et al., 2010; Moreno-Noguer and Fua, 2012).

Non-Rigid Structure from Motion

In contrast to the template-based models, which require a reference image with a known 3D shape of the surface, which is not always easily available, *Non-Rigid Structure from Motion (NRSfM)* methods (Torresani et al., 2001; Xiao et al., 2004; Brand, 2005; Bartoli and Olsen, 2007; Agudo et al., 2012a) are recent techniques that do not require prior knowledge to simultaneously track and recover the 3D shape of non-rigid 3D surfaces. Hence, NRSfM enhances effectiveness at the expense of increased complexity with respect to the template-based models.

These algorithms receive multiple images of the target object, which have been taken over time, generally in the form of a video sequence. They extract FtF 2D correspondences by tracking points over the sequence, and try to recover the 3D locations of the individual feature points in each input image (this is the main difference to rigid SfM methods, which recover the same 3D locations of individual feature points for all images). They also obtain the motion of the

camera by modelling it as an additional unknown of the problem. This can be expressed mathematically by the following conceptual equation:

$$\begin{aligned} \operatorname{argmin}_{Rt_j, M_i} \quad & \sum_{j=1}^{n_f} \sum_{i=1}^{n_v} \left\| \vec{m}_{i,j} - K R t_j \vec{M}_{i,j} \right\| \\ \text{subject to} \quad & \text{constraints,} \end{aligned} \quad (2.26)$$

where $\vec{m}_{i,j}$ is the 2D image position of the visual cue i and frame j , $\vec{M}_{i,j}$ are their corresponding 3D coordinates at that instant. n_f is the number of frames and n_v the number of vertices. $R t_j$ represents the transformation in the frame j and K is the calibration matrix.

Because of this complexity, there are various solutions that address similar issues:

- Similar to learning-based methods, modelling a deformable surface as a mesh of vertices typically yields many DoF. However, many of these DoF are coupled, which can be enforced by representing the deformations as a linear combination of basis shapes. Likewise, similar to template-based methods, solving NRSfM requires additional constraints in order to obtain a robust solution. These additional constraints can be summarized as: *orthonormality* (ensuring that rotation matrices are orthonormal), *temporal consistency* (Salzmann et al., 2007b; Olsen and Bartoli, 2008; Rabaud and Belongie, 2009) (ensuring that the variation of the shape and the camera motion are small between two consecutive frames), *geometric constraints* (ensuring a uniform level of smoothness across the whole surface -*global smoothness*- or forcing the surface to be locally or piecewise smooth -*local smoothness*-) or other restrictions such as light (White and Forsyth, 2006).
- Along with these techniques, the idea of subdividing the problem through the motion of several rigid deformations (Xiao and Kanade, 2005) has recently arisen in parallel. It is based on dividing the global reconstruction into local ones as shown in Figure 2.40 and then combining them with a consistent interpretation. In this case, the rigid deformations move independently and the movement of the whole scene is considered a deformation. These local deformations can be modelled following *isometric* (Taylor et al., 2010), *planar* (Varol et al., 2009; Collins and Bartoli, 2010) or *quadratic* (Fayad et al., 2010) representations, which implies that

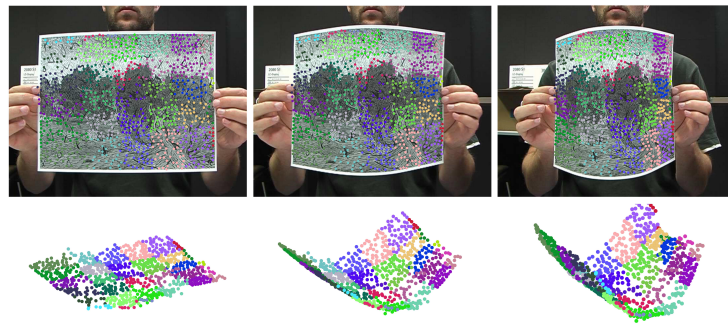


Figure 2.40: Results obtained on real images of a sheet of paper undergoing large deformations (by (Fayad et al., 2010)).

deformations cannot be applied with materials such as clothing due to the large variability of complex deformations.

Finally, it should be noted that although the NRSfM methods are effective and attractive at the same time, they have certain weaknesses that are worth citing. First, these algorithms rely on the tracking of feature points throughout a video sequence, which makes a well textured surface necessary. On the other hand, these methods have only demonstrated applicability when tracking smooth deformations of 3D surfaces (Salzmann et al., 2007c). More complex deformations require a greater number of modes hence making the system ambiguous.

2.4 Discussion

Multiple methods have been presented to recover the position and orientation of the camera from an image. Some of them use multiple cameras to simplify the problem, but they require bulky and more expensive hardware. Other methods rely on a single camera (monocular systems) and more sophisticated CV techniques. Thus, certain solutions add markers to the scene, obtaining a fast and accurate camera pose at the expense of environment adaptation. Other alternatives store (before the tracking occurs) knowledge about the scene in a 3D model, which is matched to the visual cues detected in the image to estimate the camera extrinsic parameters. Although they have to generate the 3D model, they can handle textureless scenes and offer a robust response in a reasonable amount of time. Other approaches solve both the camera motion and the structure of the scene by tracking certain visual cues throughout a video sequence. They only require some previous frames as input, but this increases the computational cost.

There are also some works that deal with deformable objects. Besides solving the problem of visual correspondences, they have to calculate the new shape of the object in each frame. This chapter has presented a study of the different methods (see Table 2.2) found in the literature that solve the problem of the ambiguities that arise when reconstructing this type of surfaces.

The physics-based models achieved really accurate results; however, as pointed out throughout this section, many other parameters and considerations require to be taken into account, such as model representations and the range of physical parameters. As an alternative, the learning-based models have the advantage of the automatic creation of the model without any prior knowledge of the physical parameters. Nevertheless, the model creation requires a specific database that is not always available. It also implies that each model should have its own database. The template-based models, in turn, recover the surface from monocular video sequences or from a single image. Their drawback, therefore, comes from the need to have well-textured surfaces in order to obtain the maximum number of correspondences and reduce the influence of outliers. Other approaches such as the NRSfM, solve both the camera motion and the structure of the scene by tracking some visual cues along a video sequence. They only require some previous frames as an input, but increase the computational cost. They also require well-textured surfaces in order to detect a correct set of correspondences.

In sum, the aims pursued (robustness, real time, accuracy, etc.) and the restrictions set by the specific problem (rigid, deformable, with or without prior knowledge, etc.) determine the selection of the appropriate tracking method.

Table 2.2: Advantages and disadvantages of different types of models.

Type	Advantages	Disadvantages	Oriented to
Physics	Accurate	<ul style="list-style-type: none"> • Difficulty in defining the physical model • Computational cost 	Graphics
Learning	The model is automatically built	The training data generation	CV
Template	Only need a single image to recover the surface	They suffer from bad correspondences	CV
NRSIM	Prior knowledge is not needed	<ul style="list-style-type: none"> • Well-textured surfaces • They need a video sequence 	CV

Part II
Proposal

Rigid Surface Marker Tracking

*Askok irudipena du...
Askok itxura hutsetik,
ez barruz ezagutzetik...
...ai, bistaratuko banu!*

MAIALEN LUJANBIO

A synthesis of this chapter has been published in:

Álvarez, H., Leizea, I., and Borro, D. "A new marker design for a robust marker tracking system against occlusions". Computer Animation and Virtual Worlds, Vol. 23, N. 5, pp. 503–518. 2012.

3.1 Introduction

There are various techniques for implementing the tracking step in *Computer Vision (CV)*. Some solutions use the characteristics of different environments (such as keypoints or edges) as reference points in order to deduce the camera pose. Systems like PTAM (Klein and Murray, 2007), DTAM (Newcombe et al., 2011), VSLAM (Karlsson et al., 2005), or other markerless tracking systems (Barandiarán et al., 2007; Álvarez et al., 2011) compute the pose without adding fiducial markers in the environment. However, this is not always straightforward in unknown environments. It may be that the required information is not adequate to calculate the pose, or even that the calculated pose estimation easily drifts over time. Furthermore, most of these approaches return a relative pose to the starting point (first keyframe) instead of an absolute pose, which involves a difficulty aligning virtual objects (owing primarily to the scale factor).

In order to overcome these challenges, some approaches called *marker-based tracking systems* are often used. They include easily detectable predefined artificial signs in the environment (also called *marker* or *fiducial*). Compared to *natural feature-based* approaches, marker-based tracking systems offer a fast and robust detection since the provided information from markers is used to calculate the pose. They often contain an identification pattern that serves as a reference point in order to deduce the 2D-3D correspondences, since the containing points of the pattern are already known. Hence, the extrinsic camera parameters can be estimated between the image plane and the real world.

As Figure 3.1 shows, markers can have different shapes. *Template markers* are black and white markers that have a simple image inside a black border (ARToolKit (Kato and Billingham, 1999)). This inside pattern serves in turn as identification for each marker. They are usually square markers with a thin black border. Their four corners are easily detectable to compute the pose. *Circular markers* such as (Naimark and Foxlin, 2002) estimate the pose using the shape of a projected circle. They have the advantage that the centroid (the centre of the circle) is invariant to the view direction and the angle. Another fiducial possibility is the use of *dots as markers* (Bergamasco et al., 2013). They are made up of several dots which store the identification based on their relation position. They use projective invariants (which do not change under a projective transformation) to perform both detection and recognition in the image plane. Moreover, the area taken is relatively small in comparison to square and circular markers. Similarly, *Uniform Marker Fields* (Szentandras et al., 2012) are defined as planar structures which mutually overlap partial markers. The large size of the marker allows easy detection within the scene from various points of view and rapid localization can be achieved by recognizing the sub-areas within the field. They combine the features for detection, localization and individual identification since the field is uniformly distributed across the whole area of the marker field. Likewise, other examples such as (Bergamasco et al., 2011) use *hybrid systems* which combine dot-based markers and circular markers. Other markers that can be found are *barcode markers* which consist of black and white cells like the popular *QR codes*. And finally, *image markers* that use colour images or even *infrared markers* that recognize the infrared rays to extract the location.

The popularity of marker-based systems is also due to the ease of implementation and the wide range of toolkits that assist the process (ARToolKit (Kato and Billingham, 1999), ARToolKitPlus (Schmalstieg and Wagner, 2007), ALVAR (ALVAR, 2011), ARTag (Fiala, 2005a)). In addition to computing the correct camera pose, these toolkits provide an encode information (or

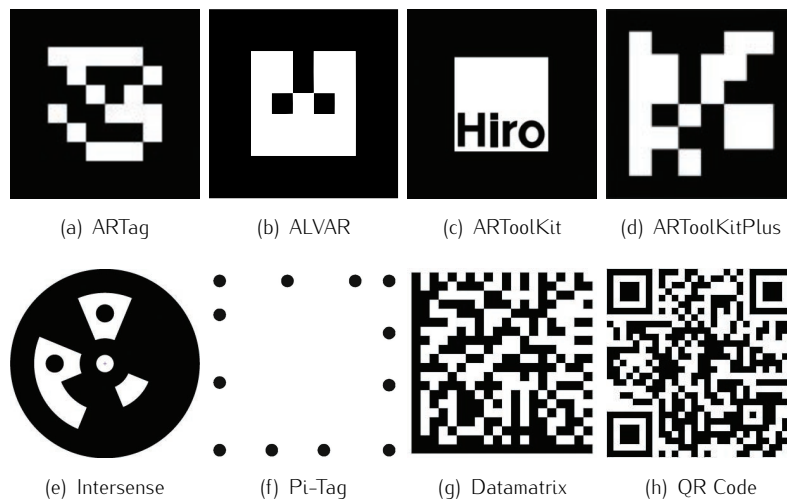


Figure 3.1: Different types of markers.

an identification) that enables attaching certain objects or interactions to the markers. These types of toolkits, and in general the large majority of marker-based tracking systems, are executed according to the following three main stages: *detect the marker*, *identify the marker* and *calculate the pose*.

However, despite marker-based tracking systems being robust and accurate, with a real-time performance, there is a major drawback which is that the environment adaptation is not always possible and other shortcomings such as marker occlusions. Typically, these systems fail when the marker is not completely visible, thereby generating undesirable effects.

The following chapter describes a method that is based on one of the cited toolkits, more concretely ARToolKitPlus, improving its performance when an occlusion is applied to the marker.

3.1.1 ARToolKitPlus

ARToolKitPlus is a widely used non-commercial marker tracking system that uses black squared markers to compute the camera pose. It is an extended version of the popular open source library ARToolKit (Kato and Billinghurst, 1999), but this new version is primarily oriented towards to mobile media devices which technically have a lower hardware. It was developed by (Wagner and

Schmalstieg, 2007) and due to high demand, the code was released (freely available under the GPL open source license).

Apart from the ability to run on mobile devices, the most remarkable characteristics compared to its previous version (ARToolKit) can be summarized as follows: the inclusion of the codification of *BCH* (Bose, Chaudhuri, Hocquenhem) markers increasing the number of markers to 4096, the identification of the markers based on a digital codification or an automatic thresholding through which it is able to analyse the pattern within the markers. Nevertheless, even though ARToolKitPlus is optimized for mobile devices, it shares the same main features as the primary one, such as obtaining the transformation matrix.

The resolution of the camera registration problem through a marker tracking system like ARToolKitPlus involves several steps (see Figure 3.2). The process begins with an input image taken by the camera being introduced into the ARToolKitPlus pipeline. From here, it starts looking for a pattern in the image. If a pattern is found, it tries to identify by matching it to a list of patterns in a precompiled database. If the matching is satisfactory, the associated information about the specific pattern allows determining the relative position of 6 *Degrees of Freedom* (DoF) in real time. The following is a more detailed explanation of the stages that compose the whole process:

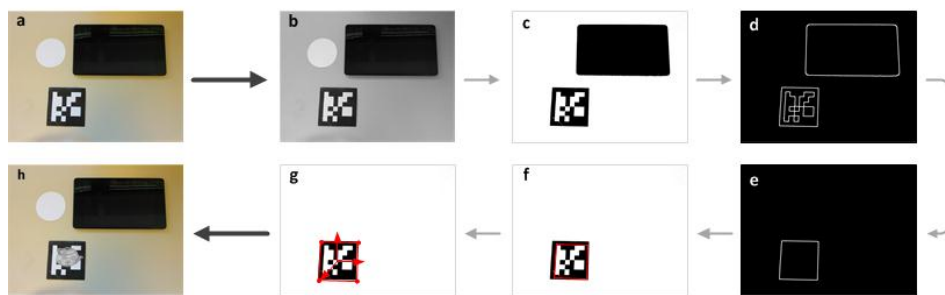


Figure 3.2: ARToolKitPlus pipeline.

1. **Image acquisition.** An input image (the current frame) can be captured through different types of video devices. This image serves as the starting point for the algorithm to find the markers (see Figure 3.2(a)).
2. **Preprocessing.** The image is converted to a greyscale in order to facilitate the detection of squared marks (see Figure 3.2(b)). Then, ARToolKit

performs a low level-image processing (such as edge detection) by thresholding the complete image (see Figure 3.2(c,d)). It includes an automatic thresholding that is dynamically adapted. It uses the mean of all extracted marker pixels if previously detected, or otherwise randomizes the threshold.

3. **Acceptance/rejection candidates.** A fast rejection of obvious non-markers and an acceptance test for potential markers is performed in this step (see Figure 3.2(e)). The resulting areas of the previous step are immediately discarded if they are too large or too small. Small areas means that the marker is very far away from the camera and consequently the pose would be very uncertain.

Another criterion that this algorithm uses is the colour of the marker. The simple template markers have a small number of holes (white pixels) inside the (black) marker area. At this point, it is also important to take into account the fact that completely black areas are not markers.

Finally, the areas that are big enough to include a marker and are based on a square shape are selected. Using the obtained vertices in the previous step (*Preprocessing*), the areas that have 4 corners are considered squares (see Figure 3.2(f)).

4. **Identification.** With the defined region, a normalized phase is performed in order to extract the content of the marker (the central area that differs from other markers) (see Figure 3.3). ARToolKitPlus uses a digital codification (decoding the data markers) in such way, that each pixel in the interior of the marker is represented as a bit. Thus each marker contains a bit value chain (being also the identifier of the marker), and the process consists in searching for this identifier in the database.
5. **Pose calculation.** By knowing the 2D positions of the edges and vertices that define the 2D marker, it is possible to estimate the 3D relative position and rotation of the camera in respect to the marker (see Figure 3.2(g)). This pose is computed through an homography estimation using the *Direct Linear Transformation (DLT)* algorithm (see Appendix A). Since a coordinate system centred on the marker is being used, and taking into account that the 3D coordinates of the markers are known and fixed, the required four 2D-3D matches are then extracted in order to estimate the homography. Note that these matches must be non-collinear and the

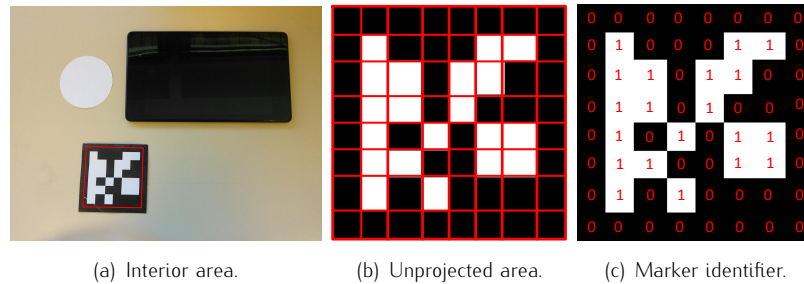


Figure 3.3: Perspective correction of a BCH marker

marker must be in a plane, i.e., all the vertices belong to the same plane ($Z = 0$).

3.1.2 Marker Occlusion

As shown in the previous section, it is necessary to have correct visibility of the marker in order to compute the camera pose. Otherwise, when a marker is occluded (see Figure 3.4), the detected candidates in the image will not be considered as a marker. ARToolKitPlus for example, fails even when the marker is only occluded by approximately 2%.

Thus, some of the conditions for the camera registration using ARToolKitPlus noted in the previous section (see Section 3.1.1) are not met. When a partial occlusion occurs, is impossible to calculate the new pose of the camera due to the following problems:

- The contour ceases to be rectangular. When a marker occlusion occurs, the square shape of the marker will generally be lost. The areas that do not satisfy the condition that the four vertices compose a rectangle, will therefore be discarded (see Figure 3.4 left).
- Inability to retrieve the interior bits of the marker, and therefore to know its unique identifier. An occlusion in the interior pixels of the marker (see Figure 3.4 middle and right), makes the marker identification changes and consequently, the template cannot be matched with any of those from the database.

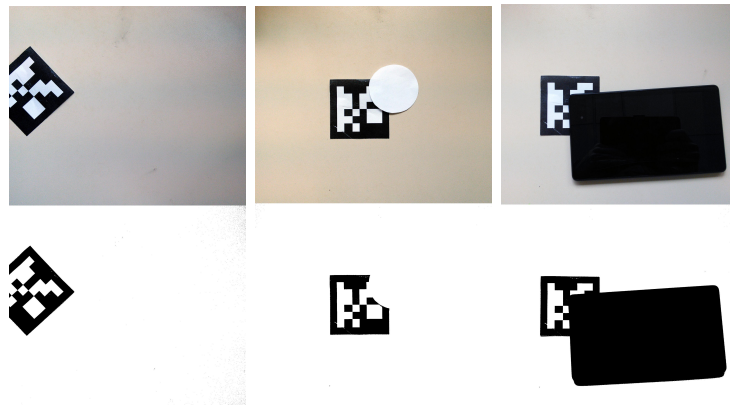


Figure 3.4: Segmentation (bottom) for different occlusions (top).

- Some of the corners of the marker are not visible (see Figure 3.4 left) and therefore there are not 4 required matches to implement the DLT algorithm. The four corners of the marker (with known 3D coordinates) are not identifiable to compute the homography that transforms the marker from its origin to the corresponding location in the current frame.

It is worth stressing that the homography computation does not specially require the four corners, but it requires four non-linear matches.

3.1.2.1 Previous Works

To date, there have been many research works dealing with the problem of handling occlusions in marker tracking systems.

Some authors place multiple markers in the scene to overcome the marker occlusion problem (Kato and Billinghurst, 1999; Garrido-Jurado et al., 2014). In this way, they increase the probability of finding one visible marker at the cost of more environment adaptation. Continuing with this idea, (Tateno, 2007) uses multiple-layer markers. The main disadvantage of this approach is the accuracy waste after switching between marker layers because of scale change. In addition, their markers require larger sizes to get similar recognition rates to that of a common single marker.

ARTag is a marker tracking system (Fiala, 2005a) that uses edge segmentation for the identification of target patterns. It is more robust than ARToolKitPlus under bad illumination conditions, and it is capable of closing

broken contours (Fiala, 2005b). Nonetheless, it only supports small occlusions, such as putting the finger on the marker.

In (Álvarez and Borro, 2009) a method that tracks the bounding box of the marker is detailed. This method supports strong occlusions and does not add extra information to the marker or the environment. It is ideal for mobile devices because of its low computational cost. However, it only updates 4DoF (translation in X, Y, Z axes and rotations in Z axis), so it is not a valid solution for all *Augmented Reality (AR)* applications.

The approach in (Wagner et al., 2008) presents two new marker designs. Although they are not designed for occlusions, they support some restrictive partial occlusions, as they use the frame of the marker to codify their digital identification. An incremental feature tracking to handle marker occlusions is also described. It assumes that the marker lies on a textured plane, from where features are detected. This is not always possible, and therefore this requirement becomes scene dependent. Furthermore, according to the authors, this technique is limited only to a few seconds due to drift.

In (Malik et al., 2002; Maida et al., 2010) an incremental feature tracking is also applied. In these cases, features inside the marker are detected and tracked. A correct spatial configuration of features is required to guarantee a minimum number of features and to obtain a precise homography between two consecutive frames. Therefore, they are not a valid method for all markers, since they depend on the patterns that are inside the marker. They also suffer from the problem of drift.

The method described in (Marimon et al., 2007) uses a *Particle Filter (PF)* to obtain the 6DoF of the camera when the ARToolKit library fails. It minimizes the reprojection error between the marker corners and the features detected in the current image. The number of particles should be large enough to adapt to sudden movements. The drawback here is that the computational cost of this operation is too high. Therefore, the robustness is jeopardized by the number of particles, that is, by the real-time requirement.

Recently, (Suzuki et al., 2013) have presented a new marker design oriented to cylindrical surfaces. It is a robust approach that estimates the transformation matrix even if the marker is partially occluded. The marker is composed by few white dots in a black frame that serve to detect the marker and estimate normals of the surface in order to deduce the curvature of the cylinder. Even so, the design

is an *ad hoc* proposal with low adaptability which needs to improve the dots detection.

The method in (Uchiyama and Marchand, 2011) describes a new marker design that is based on randomly scattered dots (random dot markers). Each dot is described by the spatial location of its neighbouring points (*Locally Likely Arrangement Hashing (LLAH)* geometric descriptor (Nakai et al., 2006)), and a minimum set of dots must be identified to recognize the marker. Occlusions that are supported depend on the spatial distribution of dots. A uniform distribution would be ideal but penalizes distinctiveness of the geometric descriptors. The computational cost of this approach, 50–60 milliseconds when using the publicly available implementation of the author¹ and the same hardware configuration as that presented in the experiments, exceeds real-time requirement (~ 33 milliseconds). In addition, this solution cannot be adapted to popular and existing marker tracking systems such as ARToolKitPlus.

Using a similar system, (Chen et al., 2013) proposes a scalable map of random dot markers for large areas. The scalability is achieved by a smart arrangement of repeated random dot markers that are placed only on the ground surface. Although the system is more robust to occlusion than classic squared-shaped fiducial markers, it has the same problem as the previously cited approach (Uchiyama and Marchand, 2011), since the computational cost increases when the detector finds a lot of points.

¹<http://hvrl.ics.keio.ac.jp/uchiyama/me/code/UCHIYAMARKERS/index.html>

3.2 Proposed Method

Considering the limitation of marker tracking systems that do not support occlusions in recognizing and registering, a new marker design has been developed to overcome the problem of marker occlusions. Instead of using multiple markers so that there is always one marker visible (Kato and Billinghurst, 1999), this proposal offers more robustness and does not require more environment modifications.

The new marker design takes advantage of an untapped frame to place some textures that will be tracked during marker occlusions. These textures are customizable and generate, by default, a uniform distribution of features (with known 3D coordinates). It is thus not scene dependent. It is a highly adaptable solution because it can be used by any marker tracking system that does not take advantage of their frame. This idea has been introduced in the popular marker tracking system ARToolKitPlus because it uses markers that only use its central area to codify the digital identification. In addition, it should be pointed out that this customization allows users to make their own designs.

Unlike systems such as (Marimon et al., 2007; Uchiyama and Marchand, 2011), this new design deals with occlusions updating the 6DoF of the camera in real time (see Figure 3.5), without losing robustness. Moreover, in contrast to the methods described in (Malik et al., 2002; Wagner et al., 2008; Maldi et al., 2010), this proposal offers an incremental tracking combining two methods to offer a robust tracking. The first is a fast technique based on temporal coherence, whereas the second is a robust technique based on appearance, which is used as a recovery mode.

3.2.1 Justification of the New Design

Partial occlusions of the marker cause tracking failure, because none of the detected candidates in the image are considered as a marker due to the change of the shape, i.e., the four corners of the marker are not identifiable to compute the homography (see Section 3.1.2). To overcome this constraint, it is necessary to have as many features as possible (with known 3D coordinates) in order to increase the probability of finding enough 2D-3D matches to update the camera pose.

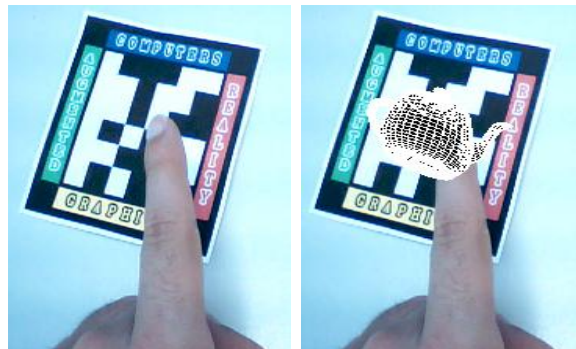


Figure 3.5: Left: An example of failure when the marker is occluded (no virtual teapot is rendered). Right: the output that is obtained using the proposed Occlusion module.

For this reason, customized textured borders have been added to the marker (see Figure 3.6, middle), which constitute an area not used by the ARToolKitPlus library. Furthermore, the pipeline of ARToolKitPlus remains unchanged, since the presented customized borders do not distort the marker identification step and offer more information in case of occlusions.

Each customizable border is painted with a different colour (see Figure 3.6, right) to accelerate the matching step. The colour codification helps to distinguish the border that each point belongs to, while different texture patterns guarantee that not all the points look the same (see Section 3.2.4.2). It is also noteworthy that the borders of the marker have been slightly expanded (the new marker size is 12% larger) to obtain reasonable texture sizes and to maintain a thin black outer border. This border benefits the marker detection of the ARToolKitPlus pipeline (see Section 3.1.1), because the square shape is more easily detected after the thresholding step despite the presence of the textures. Moreover, this black outer border has not increased the global size of the marker too much to be a valid solution for most applications, especially those that have space limitations.

In addition, this design can also be used with non-square markers, such as those that have a circular shape. This change only affects the training phase (see Section 3.2.3), taking into account the area where the texture patches are placed. Hence, only the placement of the textures should be adapted to make the design feasible to other marker shapes.

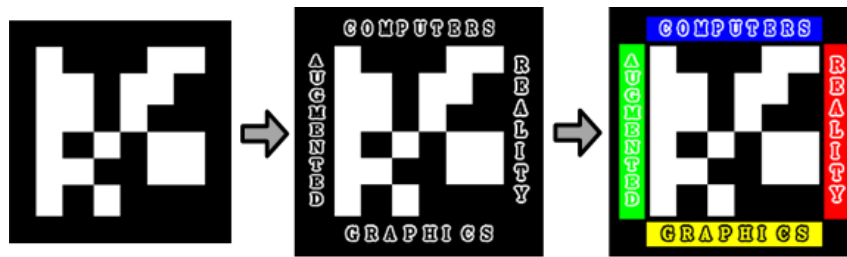


Figure 3.6: Initial version of the ARToolKitPlus BCH marker (left), an evolution that uses its frame to add some textures (middle), and the final version using colour codification to accelerate the feature matching step (right). Notice that these textures are only an example of all possible configurations, as they are customizable by the user.

3.2.2 Algorithm Overview

The proposed marker tracking system is composed of two main modules: the *ARToolKitPlus pipeline* and the *Occlusion pipeline*. The first is the common ARToolKitPlus pipeline detailed in (Wagner and Schmalstieg, 2007), whereas the second is the solution presented here to update the camera pose against the partial occlusions of the marker. The novelty lies in how these two different tracking methods are combined to exploit the advantages of each approach, obtaining a robust real-time tracking. The interaction between these two modules is shown in Figure 3.7. When the ARToolKitPlus pipeline is executed successfully, the datum associated with the current state is stored (*Update Occlusion Data*). This datum is used to initialize the Occlusion pipeline when the ARToolKitPlus pipeline fails, thus guaranteeing that the occlusion state is updated every time the marker is visible.

The Occlusion Pipeline in turn, combines two different tracking methods: *Frame-to-Frame tracking (FtF)* and *Tracking-By-Detection (TbD)*. The first is based on temporal coherence, which provides a robust tracking against non-strong camera movements and has a low computational cost. Furthermore, FtF incorporates a refinement, which minimizes the problem of drift (see Section 3.2.6). The second tracking method, meanwhile, computes the pose despite the movement applied to the camera, i.e., when the FtF fails. It has a higher computational cost because it recognizes the presence of the textures in the image by using their appearance, without temporal coherence assumptions. Additionally, TbD exploits the properties of the new marker design to perform

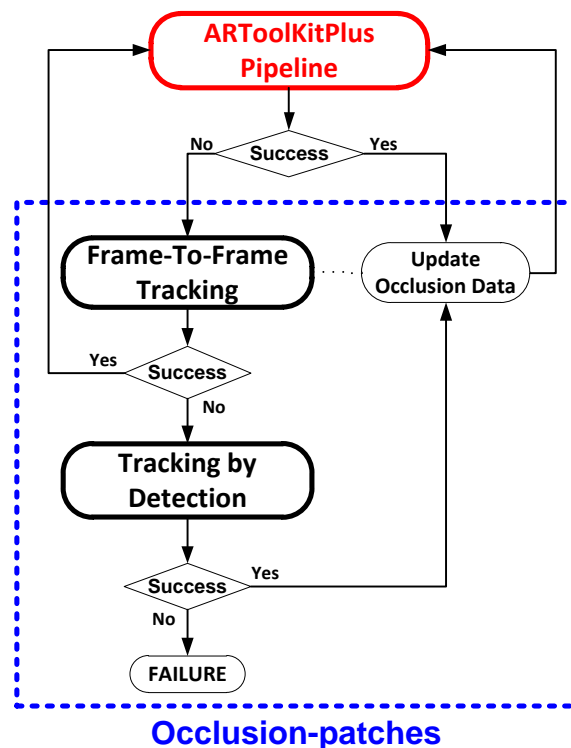


Figure 3.7: Overview of the Occlusion pipeline showing how the ARToolKitPlus and Occlusion pipelines interact, as well as, how the two different tracking methods (FtF and TbD) are mixed.

an intelligent search of the homography that best represents the camera pose. In other words, each point that is used to generate a homography hypothesis is selected from a different texture to avoid the selection of collinear points that produces a singularity.

Therefore, although these tracking methods are based on existing solutions, the main idea of combining both methods consists in using the FtF for as long as possible because of its robustness against smooth camera movements and low computational cost, and using the TbD only when the FtF fails, as a recovery mode, because it involves more computational cost. Also, the TbD allows a less degree of occlusion, because the matching quality between the image appearance and the reference appearance should be reasonably high not to generate false positives.

The TbD method is based on appearance so it requires a database of feature descriptors that belong to the marker. This is computed during an offline phase that is described in the following section. In addition, the online phase of Occlusion pipeline will also be detailed for the two tracking methods cited above.

3.2.3 Offline Phase

The offline phase is a training process (see Figure 3.8) that is executed only once for each marker design. This is essential in order to carry out the TbD step in the online phase. For this purpose, the marker is trained along a set of keyframes. These keyframes are processed, and the 2D features that belong to the surface of the texture borders are back-projected, obtaining their corresponding 3D values. In addition, these 3D points and their corresponding descriptors are indexed in a database, which is used to find a set of matches during the online phase. The steps involved in this phase are detailed below.

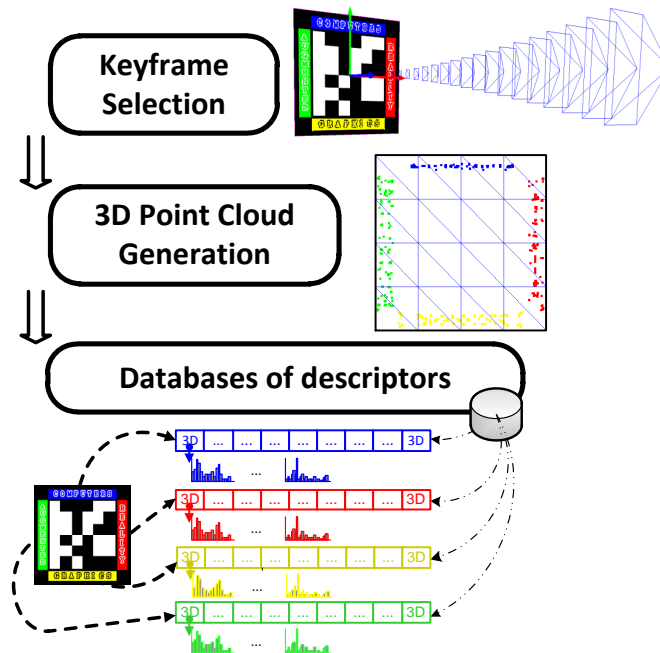


Figure 3.8: Offline phase of the Occlusion pipeline. It is shown step by step.

3.2.3.1 Keyframe Selection

During the execution process, the marker must be detected regardless of the point of view. For this reason, it is necessary to perform preprocess information where a set of keyframes captures the marker and extract the largest number of distinguishable points from different points of view. This preprocessing is performed with a synthetic marker with the aim of minimizing the user intervention in the training phase.

The number of keyframes depends on the set of movements that the camera covers at runtime, without exceeding the storage of the database. Thus, only the keyframes that put the marker in front of the camera are considered, without applying rotations. This is a valid assumption for an object that lies on a plane, and it has been successfully applied by other authors (Xu et al., 2008). According to this method and due to real-time requirements, *Feature from Accelerated Segment Test (FAST)* operator (Rosten and Drummond, 2006) has been applied to extract features during the online stage, which does not provide scale information. Nonetheless, as it is carried out in (Wagner et al., 2010), the marker is trained along multiple scales to overcome the scale ambiguity and discarding any very small scales because of the poor quality of feature detection.

3.2.3.2 3D Point Cloud Generation

Once the keyframes are selected, 2D features are detected using FAST with regard to each one of them. Among the set of detected features, those that do not belong to one of the four markers are discarded. Additionally, the remaining 2D features are divided into four different clusters, according to the border they belong to. Since the positions of the textures are fixed and known, this filter is performed by using a mask for each texture area.

In the next step, since a set of 2D-3D matches are required at runtime to calculate the camera pose, it is necessary to back-project each of the stored 2D features extracted in the previous step in order to obtain their 3D values. The marker is modelled as a triangle mesh, and each triangle is rendered with a unique colour, using the camera pose of the corresponding keyframe, which is known because it is virtually generated. The location of each 2D feature points to a unique colour, which is used to index the corresponding facet. Once each 2D feature is associated with its corresponding 3D triangle, its 3D values are calculated using barycentric coordinates (Gall et al., 2006).

3.2.3.3 Descriptors Generation and Database Creation

This step receives a 3D point cloud for each one of the four textured borders of the marker to which they belong and computes the 3D descriptor of each one. To that end, the *Scale Invariant Feature Transform (SIFT)* descriptor (Lowe, 2004) has been chosen, because it has demonstrated high performance compared with other local descriptors (Mikolajczyk and Schmid, 2005). However, the original SIFT has been replaced by a simplified version called *simplified-SIFT* (Álvarez-Ponga, 2012). It consists in replacing the original *Difference of Gaussians (DoG)* feature detector by the FAST detector and also, the use of parallel techniques (OpenMP) to reduce the computational cost.

Simplified-SIFT implies that no scale information is obtained during the feature extraction step. Therefore, a scale-space like (Wagner et al., 2010) is implemented to avoid scale ambiguity and obtain better matching results at runtime. Each 3D point is associated to multiple descriptors, one for each image that results from applying different Gaussian filters to the corresponding keyframes. More precisely, by using eight Gaussian filters, with a standard deviation factor of $1/\sqrt{2}$ the previous one (recursively), it has obtained a good compromise between storage cost and computational cost. This procedure tries to cover the expected scale ranges at runtime.

On the other hand, the reduction of computational cost involves a fixed patch size for the original SIFT descriptors and the retaining of only the most dominant orientation for each feature. Furthermore, some calculations are precomputed, such as all possible kernel orientations. As reported in (Sánchez et al., 2010b), these simplifications allow hundreds of SIFT descriptors to be obtained in a few milliseconds.

Once all the descriptors of all 3D points have been computed, they are indexed in a database using k -d trees. In addition, a different database is created for each marker border, resulting in four different databases, which offer faster and more robust matches during the online phase, because an input feature is only matched against its corresponding border database.

3.2.4 Online Phase

This phase is executed every time the ARToolKitPlus pipeline is not able to detect the marker. It uses the data provided by the last successful execution of the ARToolKitPlus pipeline to initialize the FtF (see Figure 3.7). This is a

fast method that works fine while no fast camera movements are applied. If a rapid camera movement is performed and the tracking is lost, the TbD method is applied, which uses the databases of descriptors created in the offline phase (see previous Section 3.2.3). This second method, in turn, is based on appearance and can support fast camera movements, so it is used as a recovery mode. This way, if the TbD is unable to compute the camera pose, then the tracking method returns to the initialization step of the FtF, starting the whole process all over again. Likewise, if after a few attempts the marker is not detected (*Failure* status of Figure 3.7), the Occlusion pipeline does not update the camera pose, and the system waits until the ARToolKitPlus pipeline is successfully executed. It is noteworthy that the ARToolKitPlus pipeline is executed every time to check the marker visibility and thus, the Occlusion pipeline is disabled as soon as the ARToolKitPlus detects the marker.

3.2.4.1 Frame-to-Frame Tracking

The first step of the FtF is the initialization of the data that will be used for the tracking. To that end, the last successful camera pose and frame that were registered are used to back-project the features that lie on the surface of the marker and compute their representative descriptors (see Figure 3.9). The FAST detector is used to extract these features, while the camera pose is provided by the ARToolKitPlus pipeline or by the TbD method after recovering from a failure. The back-projection process is similar to that explained in previous sections, using barycentric coordinates to compute the 3D values. Nevertheless, these back-projected features and descriptors are not the same as those generated in the offline phase. These features belong to the entire surface of the marker, that is, they are not limited to the frame of the marker. Furthermore, they do not use the robust SIFT descriptor, but a simple descriptor based on the grey values of their surrounding patches (referred as *GREY descriptor*). It is similar to the *Normalized Cross-Correlation (NCC)*, but using a Gaussian weight.

$$GREY(p) = \frac{1}{N} * \sum_{s \in S} I(p + s) * G(s), \quad (3.1)$$

where $N = \left\| \sum_{s \in S} I(p + s) * G(s) \right\|_2$, S is the set of samples that are considered around the point p , I represents the intensity of the image and G is a Gaussian weight. More precisely, a sparse 11x11 sample grid centred on each feature is used to build a 121 bin descriptor (see Figure 3.10). This descriptor is not

as robust as SIFT, but it has demonstrated considerable robustness and low computational cost (see Section 3.2.6).

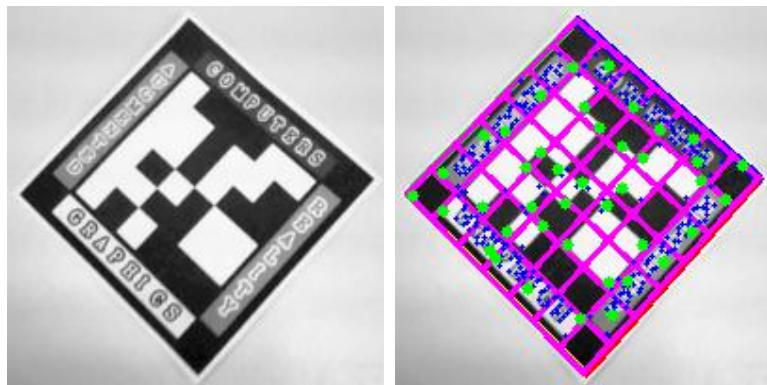


Figure 3.9: FtF tracking initialization. The image of the last successful execution of ARToolKitPlus (left). Feature detection (blue points) and back-projection of the strongest features (green points) that are uniformly distributed (right).

Because of the reduction of the computational cost, the maximum number of back-projected features has also been limited to 49. The marker has been divided into 49 equal squares (7x7 grid), and only the strongest feature is retained for each square. This is a fast method to limit the maximum number of features and maintain them in a uniform spatial distribution, which favours the response against occlusions, since it does not matter which side of the marker is occluded. This initialization step is executed every time the marker changes from visible to occluded, as it guarantees that all features and descriptors are updated with the last visible frame. It is thus adapted to possible environmental changes, such as different illumination conditions.

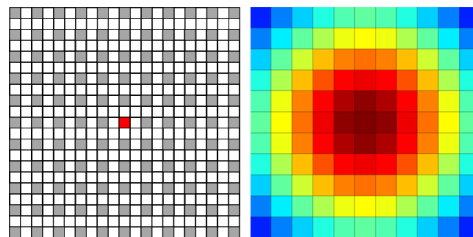


Figure 3.10: Sparse 11x11 sampling grid (left) and Gaussian weight (right) used by GREY. Red indicates more weight.

Once the initialization step has been executed, all back-projected features that are visible in the image are tracked in the successive frames using an incremental tracking similar to those presented in (Malik et al., 2002; Wagner et al., 2008). First, the new feature locations are predicted using a pyramidal optical flow (Lucas and Kanade, 1981). This technique provides a good starting point for the second step, which searches in the vicinity of predicted locations the presence of points with a similar GREY descriptors (see Figure 3.11). This way, a feature is considered successfully tracked if a point is found in a 20×20 window that is centred on the predicted location, and whose descriptor is highly correlated to the original descriptor (correlation value higher than 0.9). This step removes outlier matches and minimizes the drift problem that this type of techniques typically suffers.

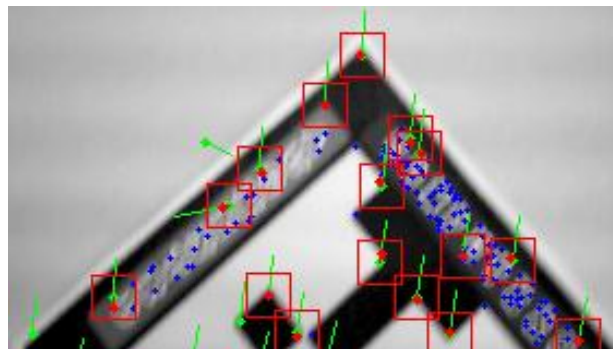


Figure 3.11: Snapshot of the FtF. Green lines indicate optical flow prediction, while the red dots are the final location after the refinement step. The red squares represent the area of each feature that is used for the refinement step. Features without the red square are considered as outliers.

Regarding all those features that follow this rule, their location prediction is replaced by the location of the point they are matched to. Nonetheless, some of these remaining matches can be outliers, so the *RANdom Sample and Consensus (RANSAC)* (Fischler and Bolles, 1981) technique is used to compute the affine transformation that represents the movement of the inlier features between two consecutive frames. Note that a minimum set of inlier matches (6 in the experiments) are required to validate the correctness of the current tracking.

Using this affine transformation, the new 2D locations of the 3D features computed in the initialization step are known, so this new set of 2D-3D matches

is used to update the camera pose. Moreover, these 3D features are projected with the new camera pose, and their 2D location is updated for the next frame.

Additionally, a refinement step, similar to that mentioned above, is performed. If there is a point with similar GREY descriptor in the 10x10 vicinity of the feature projection location, then its position is changed and its descriptor is updated. Otherwise, both the position and the descriptor are not changed, although they will be considered for the next frame as well. This last refinement step ensures that the descriptors evolve, and the quality of the optical flow for the next frame improves. It should be also noted that the simple projection of a 3D point using a non-accurate camera pose may fall in the middle of a homogeneous image area, which is not an optimal scenario for optical flow techniques. It is also required that at least 15% of visible features are updated using this refinement procedure to validate the correctness of the current tracking. This requirement, together with the affine restriction, identifies bad tracking frames due to fast movements or drift, and helps decide when to execute the TbD method.

3.2.4.2 Tracking-by-Detection

This phase serves for recovering the camera pose without any user intervention in the case of spatial coherence failure. For that purpose, the features extracted in the current frame are matched against those processed in the offline phase and are indexed in the database (see Figure 3.12).

Each of the four borders of the marker has been codified with a unique colour, building a different database for each one (see Section 3.2.3.3). Thus, for each feature detected in the current image, the border to which it belongs is calculated. To do this, the number of pixels of its vicinity (10x10 window) that belong to each colour is counted for each detected features. To determine the colour of each pixel, its corresponding *Hue-Saturation-Value (HSV)* levels are considered. If the saturation or value components of a pixel are in the extremes of their range, then this pixel is not considered as coloured, but rather as a white or black pixel. Otherwise, the hue channel specifies the colour of a pixel. Moreover, to consider a feature as coloured, 50% of their surrounding pixels must belong to the same colour, that is, there must be a dominant colour. This colour codification step is very useful, because it reduces the number of matching combinations by 75% for each feature and rules out too many points that do not belong to the frame of the marker.

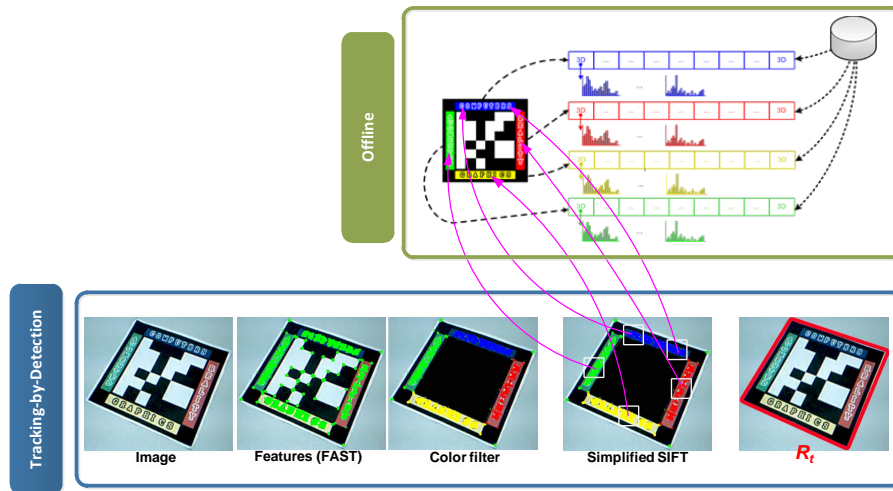


Figure 3.12: Overview of the TbD.

Once the current detected features that do not belong to the frame of the marker have been discarded, the simplified-SIFT descriptor is calculated for the remaining detected features. In addition, each of these features has been assigned to one of the four borders of the marker, and thus only the corresponding database is used to find a correspondence. The query to the database returns the two closest descriptors (d_1 and d_2 , respectively) according to the input descriptor (d_i) and the Euclidean distance is used to determine the proximity between two descriptors. This way, d_1 and d_i are considered to be positively matched if $dist(d_1, d_i) < k * dist(d_2, d_i)$, where $k \in [0..1]$ is a control parameter (set to 0.6 in this case) that discards matches with poor distinctiveness.

As a result of matching current detected features with those stored in the database, a set of 2D-3D matches are obtained. However, some of these matches may be outliers, so the camera pose can be computed using the robust hypothesize-and-verify *PROgressive Sample Consensus (PROSAC)* method (Chum and Matas, 2005). This method is based on a new sample selection criterion that favours the quality of the homography representing the transformation between the two sets of points. First, the two borders of the marker that have at least two correspondences are randomly chosen and then two matches for each of those two borders are selected according to their matching quality, which is inversely proportional to the distance between

its corresponding descriptors. This criterion overcomes the restriction of the homography generation, which states that all the points cannot be collinear. In addition, this criterion returns more stable homography transformations due to the better distribution of selected points.

Finally, the maximum number of matches has also been limited to 75 with the aim of running the application in real time. This way, the matches with the highest quality will be selected first.

3.2.5 New Interface Possibilities

The new marker design that is proposed provides more information when the marker is occluded, which can be used to develop human-machine interfaces. These new interface possibilities are complementary to those interfaces created by other authors, who have also used occlusions to implement them. For example, the interface in (Lee et al., 2004) places multiple markers at specified locations, and depending on which markers are occluded and visible, it interprets different user actions. In a similar way, the interface in (McDonald and Roth, 2003) presents a simple hand gesture recognition using the interface in (Malik et al., 2002) to treat small finger occlusions. The interface in (Uchiyama and Marchand, 2011), meanwhile, detects which region of the marker is occluded rectifying the viewpoint of the marker in the image and subtracting the rectified marker and the reference. It is noteworthy that all these interfaces can be adapted to this proposed design, although the interaction possibilities that are shown here are more difficult to obtain or computationally more expensive if other designs are used.

All the interaction possibilities presented here have to determine which features of the surface of the marker are visible. This is a simple task when the marker is not completely inside the *Field of View (FoV)* of the camera, because the projection of the occluded points falls outside the image. However, this is not the case for occlusions caused by other objects, such as the user's hands. For these cases, only those features of the surface that have a point with similar GREY descriptors in its vicinity are considered as visible.

This visibility test has a negligible computational overhead for the proposed solution because this information is already generated by the FtF (see Section 3.2.4.1), that is, this design offers the ability to efficiently develop human-machine interfaces.

Two interaction possibilities are presented as follows as a demonstration, but they can be extrapolated to numerous applications.

Occlusion Signal

The textures that have been placed along the frame of the marker imply the detection of multiple features in each border. Using this assumption, the degree of occlusion can deduce depending on the number of occluded points. Thus, an occlusion signal (see Figure 3.13) can display the degree of occlusion to the user. This is similar to the idea of a coverage signal used by mobile devices.



Figure 3.13: An example of Occlusion Signal output

Photo Viewer

Due to the presence of textures, there are multiple features in each border of the marker. Thus, the border with fewer visible features can be identified as the occluded side of the marker. Moreover, the occlusion of each border can be interpreted as a different action. To prove this usability, a photo viewer (see Figure 3.14) has been implemented, for which a different action is executed depending on which border is occluded:

- Left-border: go to the previous photo.
- Right-border: go to the next photo.
- Up-border: positive zoom to the current photo.
- Down-border: negative zoom to the current photo.

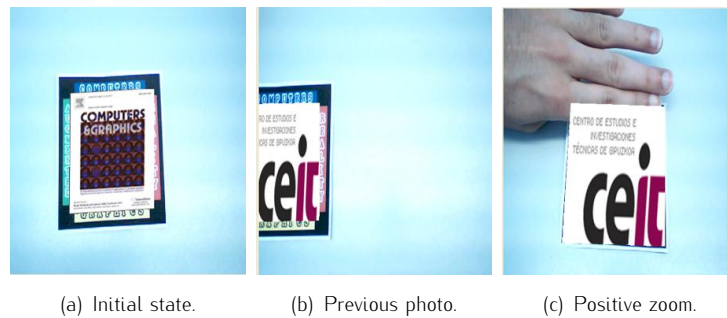


Figure 3.14: Photo viewer sequence

3.2.6 Experiments

In this section several characteristics of the new marker design that is proposed were investigated. The hardware setup consists of an Intel Core 2-Duo (Intel: Santa Clara, California, USA) at 2.40 GHz and 2 GB of RAM equipped with a Logitech QuickCam Connect (Logitech: Morges, Switzerland) webcam. All the experiments have been executed with a Windows 7 (Microsoft: Redmond, Washington, USA) operating system.

3.2.6.1 Tracking-by-Detection

First, the response of the TbD method was studied. Multiple marker designs (see Figure 3.15) were used in order to investigate the importance of choosing good texture content. More precisely, four different marker designs were developed to simulate multiple conditions:

- **Words** (see Figure 3.15(a)) represents a typical design that can be used for marketing purposes, encoding some words from a message along the frame of the marker.
- **Random-characters** (see Figure 3.15(b)) is similar to *Words* but uses randomly selected characters to demonstrate that no specific letters are required.
- **Random-shapes** (see Figure 3.15(c)) symbolizes the textures that are made using logos.

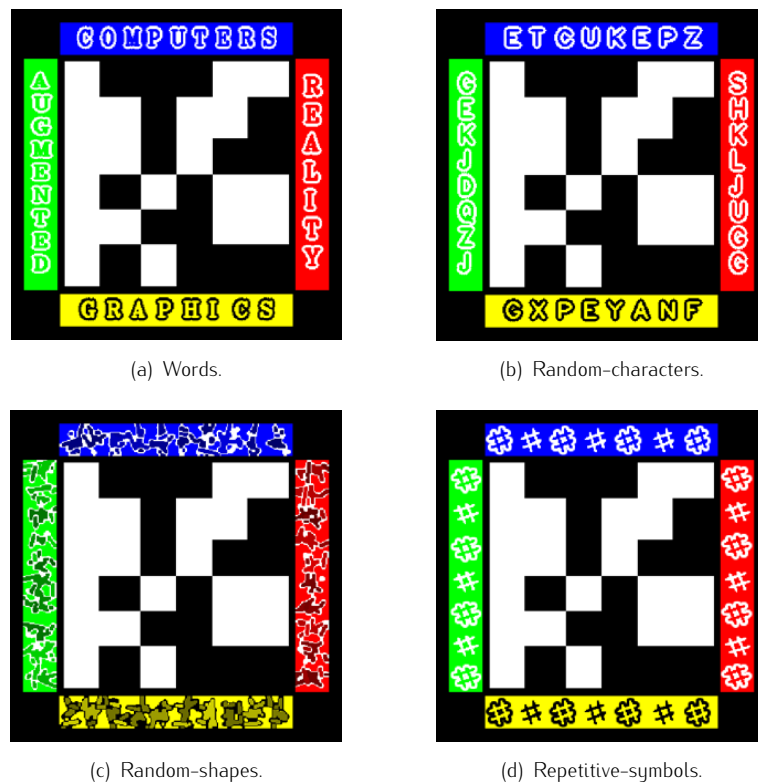


Figure 3.15: Marker designs for experiments.

- **Repetitive-symbols** (see Figure 3.15(d)) is referred to those shapes that have a repetitive pattern and do not provide distinctiveness.

To obtain these results, as well as those presented next, a set of images with a known camera pose are required to decide the success of the detection. According to this, a set of snapshots in which the marker is completely visible was acquired. These images were processed with the normal ARToolKitPlus pipeline, obtaining the corresponding camera pose for each one. Moreover, these poses were used to draw black patches on the surface of the marker, simulating marker occlusions (see Figure 3.16).

The scale was the first parameter that was examined. Nevertheless, this parameter is not dependent on the marker design, but on the size of the textures. It is noteworthy that the size of the textures is very small compared to the



Figure 3.16: Simulation of the partial occlusion of the marker

size of the whole marker, as the total increase in the size of the marker has been minimized (see Section 3.2.1). This, however, increases the difficulty of their detection, since textures appear too small in the image, without sharpness. Figure 3.17 shows the mean ratio of the TbD success for the four marker designs, according to the width of the marker and the resolution of the image.

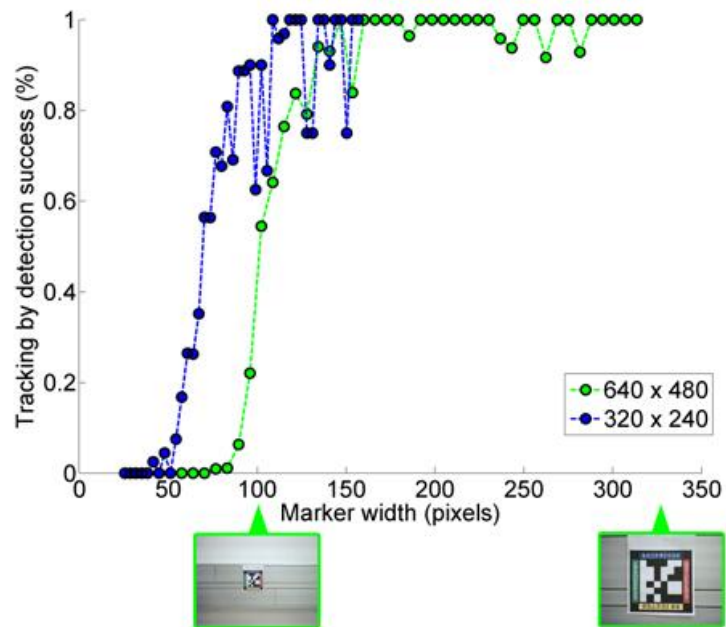


Figure 3.17: Scale study for TbD.

This ratio is increased for larger marker sizes, because textures will appear clearly and simplified-SIFT descriptors are calculated more accurately. At a 320x240 resolution, a good detection ratio was obtained when the marker width was larger than 75 pixels (22,44% of the image width), while at 640x480

resolution, a similar ratio was obtained for a width of 125 pixels (19,53% of the image width). This difference is due to the greater blur that 320x240 images have.

Notice that the detection is considered successful for an input image when the reprojection error of the marker bounding box using the real pose and that returned by TbD is below a threshold. This way, after executing TbD with different marker designs at multiple distances, it can be concluded that the marker width must be 20% of the image width to obtain a good detection. In addition, it should be pointed out that the TbD method did not have any false positives for these experiments; i.e. the presence of the marker was not been detected unless it was actually there. This is a desirable property, as this is the starting point of the FtF method that is based on temporal coherence.

Using the procedure described above, together with the scale ranges in which the TbD method is successfully executed, the simplified-SIFT was also analysed. A set of snapshots was taken for each marker design of Figure 3.15. These snapshots were used to perform several executions of the same image set, but with a different simplified-SIFT parametrization. The $i-j-k$ parametrization of SIFT (x - axis) corresponds to the number of regions (i), the number of histogram bins (j), and the patch size (k) that is used to build the simplified-SIFT descriptor. Figure 3.18 shows the success ratio of the TbD method for each marker design, according to the SIFT parametrization and for different image resolutions.

The obtained ratio of success increases when the simplified-SIFT patches are larger, since more stable descriptors are built. Nonetheless, the computational cost increases proportionally with the size of the patch, so a compromise between robustness and computational cost must be achieved. Furthermore, the image resolution also influences this decision, since at lower resolutions a smaller patch size is required in order to get similar results to those achieved at higher resolutions.

The other two parameters of the simplified-SIFT descriptors control the number of regions that divide each patch and the size of the histograms that characterize the gradients of each region (Lowe, 2004). These values are related to the size of the descriptor and they offer the possibility of establishing the degree of descriptor-distinctiveness that best suits the properties of the corresponding textures. According to Figure 3.18, the best results are obtained when the patch is divided into 4x4 regions, while the number of histogram bins that codify each region should be between 4 and 8.

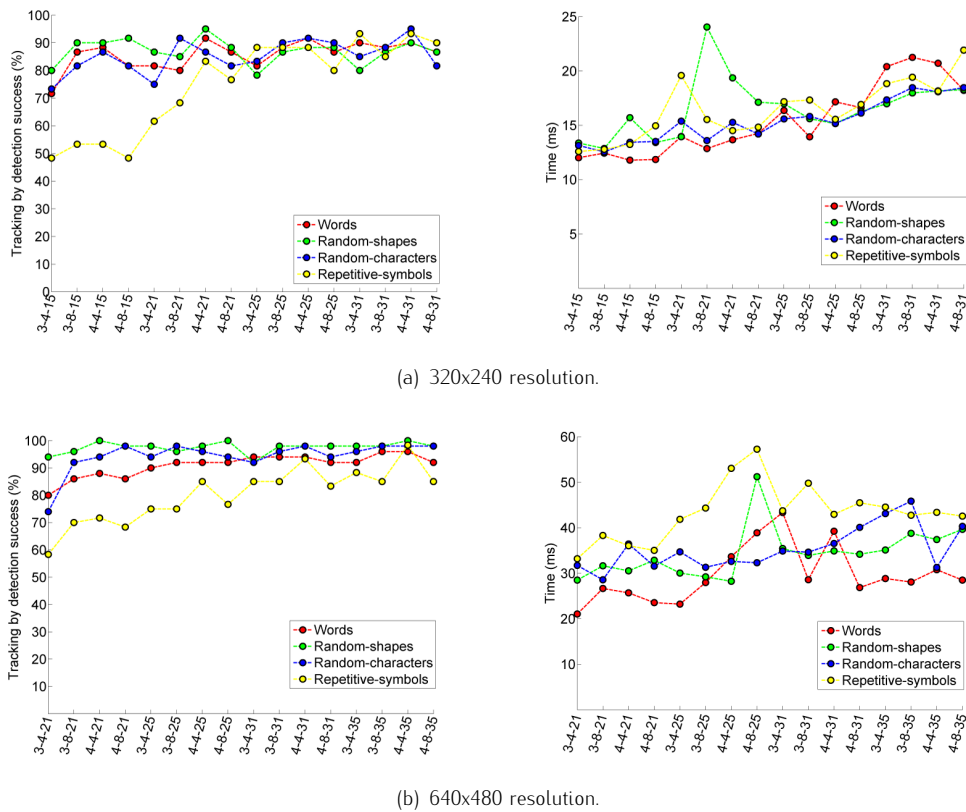


Figure 3.18: The TbD success (left) and computational cost (right) according to the simplified-SIFT parametrization and image resolution.

Additionally, although a different set of images was used for each marker design, which could explain the small differences in the success ratio, after analysing the results of Figure 3.18 for 640x480 resolution, it can be established that the textures with a repetitive pattern (*Repetitive-symbols*) harm the descriptor-distinctiveness. This in turn implies that multiple false matches appear during the detection, which increases the computational cost that PROSAC requires discarding these outliers (see Section 3.2.4.2).

The good response of *Random-characters* and *Random-shapes* designs demonstrates that textures with words and shapes are valid solutions, since they build good quality descriptors as a result of generating a lot of unique patches

around each 3D point. The response of *Words* proves that positive detection results are also obtained using textures with customized words, as long as a *repetitive-pattern* does not appear.

Taking all of this into consideration, the 4-4-25 and 4-4-31 simplified-SIFT parametrizations for 320x240 and 640x480 resolutions, respectively, are the options that generally provide the best compromise between the success ratio and the computational cost.

3.2.6.2 Frame-to-Frame Tracking

The quality of the FtF was also studied. However, the experiments were only executed for the *Words* marker design, as the results for other designs are similar. The same video sequence were executed twice, one without the refinement step presented in Section 3.2.4.1, and the other using the refinement process. It should be noted that the incremental tracking of points that is described in (Malik et al., 2002; Wagner et al., 2008) is very similar to the proposed limited version of FtF, which disables the refinement step. Thus, the main objective of this experiment was to compare the FtF with a simulated version of (Malik et al., 2002; Wagner et al., 2008). Although the video is at 640x480 resolution, the FtF was executed at 320x240 resolution to reduce the computational cost. In the video, the marker is occluded by the user's hand movement, and the marker lies on a textured surface to increase the difficulty of tracking.

As shown in Figure 3.19, this experiment demonstrates the robustness of the refinement step, which returns a good pose despite the hand occlusion; because the poorly tracked features are corrected using the correlation of the GREY descriptors (see Section 3.2.4.1). The absence of the refinement step causes an increase in tracking error due to the poor prediction of the optical flow, which is confused by the hand movement. This last option is the technique used by other authors (Malik et al., 2002; Wagner et al., 2008), so the FtF proposed here offers greater robustness compared to similar solutions.

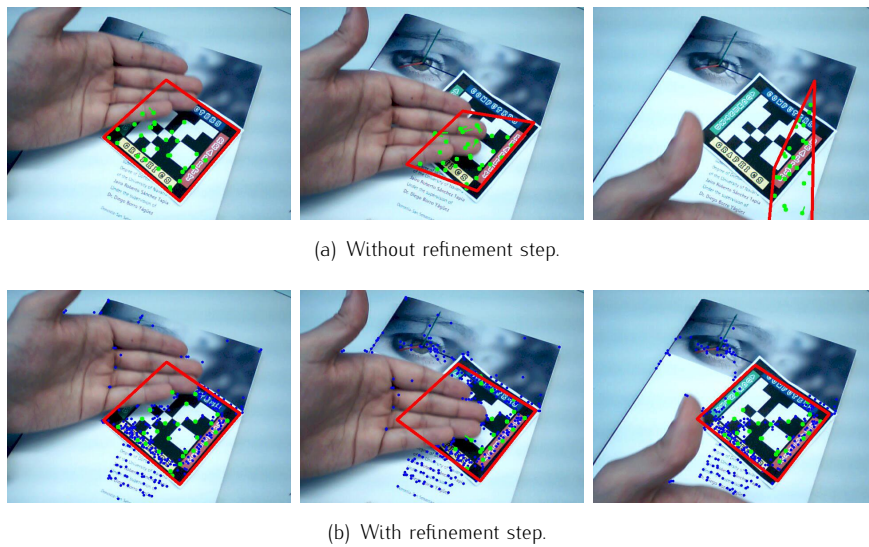


Figure 3.19: Camera pose (red) using the FtF.

The computation time for the two executions using an non-optimized code was 8.1 milliseconds and 13.05 milliseconds, respectively. The time difference is due to the cost of computing and matching the GREY descriptors that are necessary for the refinement step. Nevertheless, it should be pointed out that this would be the worst scenario for the FtF, since there are a lot of visible features that must be tracked (none of the marker points fall outside the image), and there are a lot of points around the marker (blue points of Figure 3.19(b)), whose GREY descriptors must be computed, and which can produce false matches. Despite all this, the FtF runs in real time and is able to compute a correct camera pose.

3.2.6.3 Occlusion Pipeline

Figure 3.20 shows the response of the Occlusion pipeline for a video sequence where both tracking methods are mixed. To have the correct pose for each frame of the video and compute the reprojection error (right of Figure 3.20), the marker was visible during the entire sequence in order to successfully execute ARToolKitPlus. Moreover, the procedure described above was used in order to simulate marker occlusions in each frame of the video.

The image position of the centre of the marker along the entire sequence is shown to the left of Figure 3.20(a) to indicate that the marker is continuously moving and even making fast movements that can impede tracking. The red dots in Figure 3.20(b,c) represent the execution of the TbD.

The FtF is executed for as long as possible due to its low computation time (middle of Figure 3.20). When this method fails as a consequence of a fast movement or error accumulation, the TbD is called in to compute the correct camera pose and restart the FtF. In this way, the reprojection error between the correct camera pose and that returned by the Occlusion pipeline (right of Figure 3.20) is maintained at reasonable levels most of the time. The few frames that have a higher reprojection error are those frames that the FtF takes to indicate failure. Nevertheless, the time it takes to react is very small (with non-optimized code), and the results are still visually acceptable. After executing this experiment with two different image resolutions (320x240 and 640x480), it was concluded that the results are similar. At 320x240 resolution, the computational cost is lower at the expense of higher reprojection error, whereas at 640x480 resolution, a lesser reprojection error is obtained at the expense of more computational cost.

Finally, Figure 3.21 presents the output of the Occlusion pipeline for different types of marker occlusions. Figure 3.21(a) demonstrates that the Occlusion module supports hand occlusions, including recovery from a fast camera movement (see Figure 3.21(b)). Figure 3.21(c) shows the ability to compute the camera pose when the marker is partially outside the image. Figure 3.21(d) represents the tracking success when the marker is occluded by multiple objects. Lastly, Figure 3.21(e) and Figure 3.21(f) display the robustness against occlusions made by objects with colours similar to the borders of the occluding object.

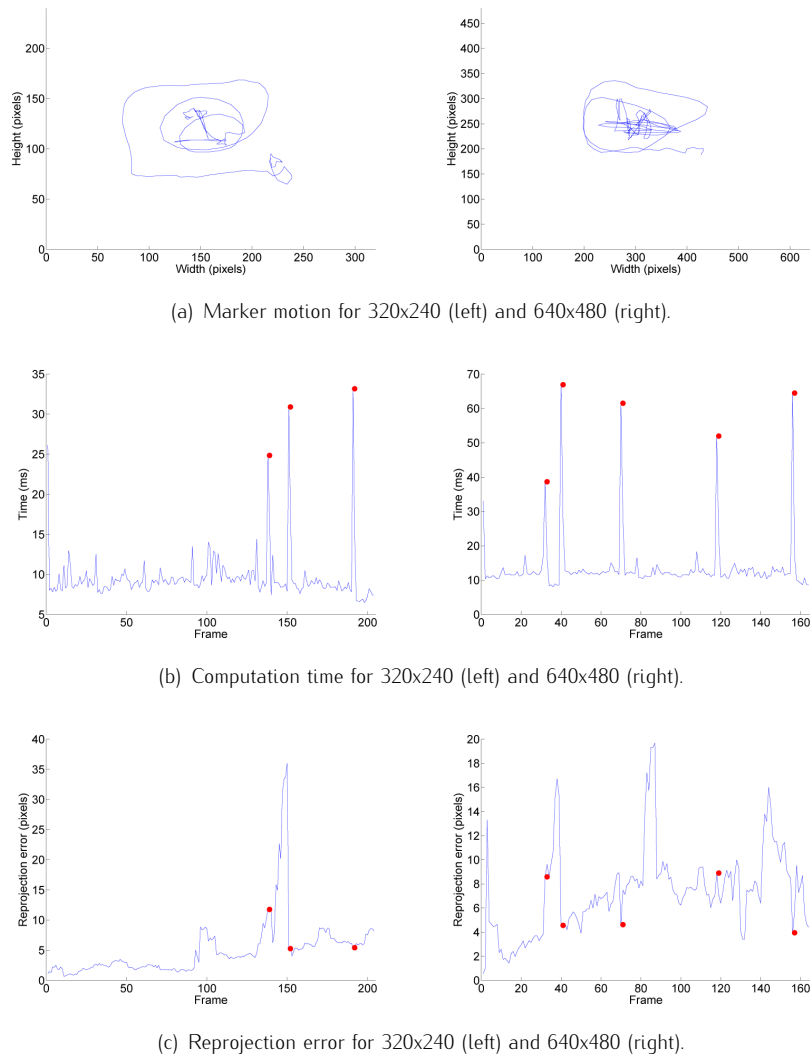


Figure 3.20: Occlusion-patches response for a video sequence.

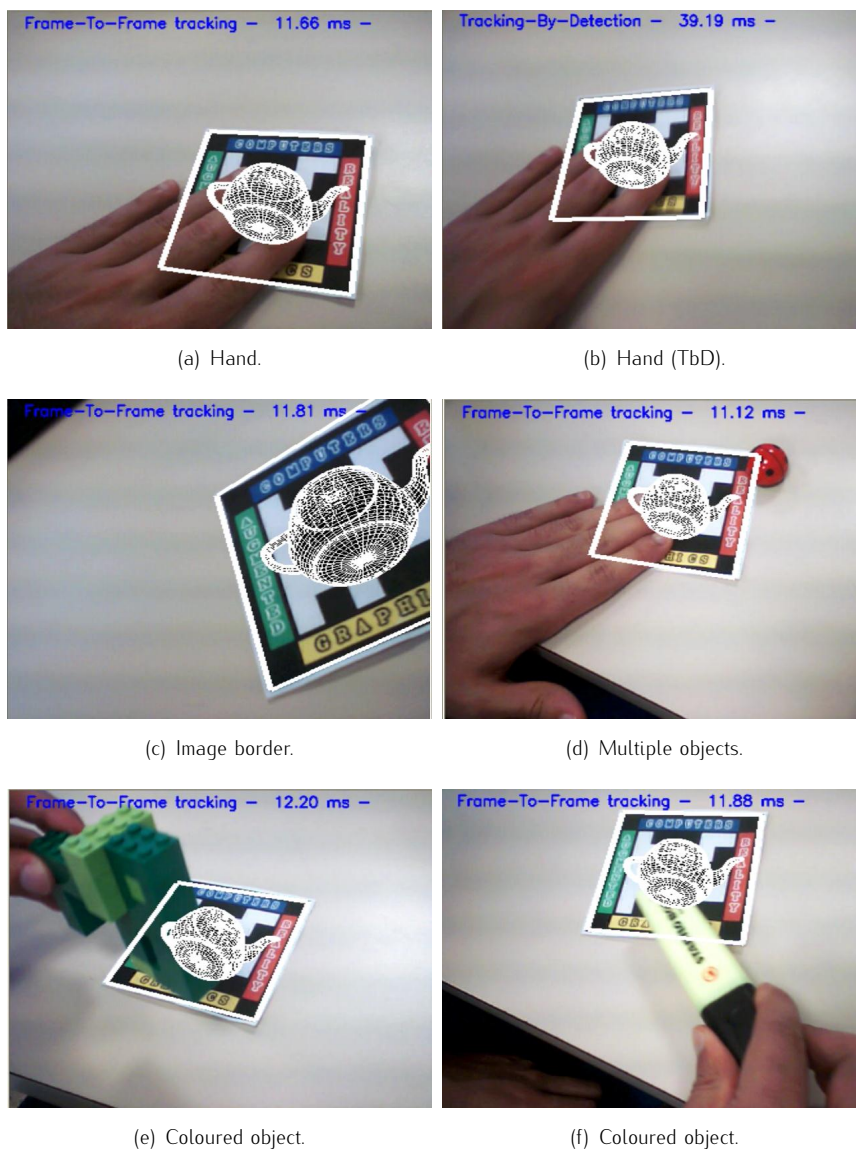


Figure 3.21: Occlusion-patches output for different occlusions.

3.3 Discussion

The use of fiducial markers has become a good alternative for solving the main problem that optical tracking systems have to deal with, that is, the registration. Such approaches are known as marker tracking systems.

With a marker and a camera it is possible to build a system based on markers in a fast, cheap and simple manner. Furthermore, the wide variety of toolkits covering on the market, facilitates the tracking process even further. Additionally, these systems offer very low execution times to the point that they are usually integrated in mobile devices. Despite these observations, the biggest handicap of these systems lies in the fact that the toolkits force up the marker to be completely visible in the image in case of an occlusion occurs. With small occlusions, the system usually fails and causes undesirable effects.

Accordingly, this chapter has presented a method to overcome the problem of marker occlusions. Using a widely used non-commercial marker tracking system ARToolKitPlus, which has the same problems with occlusions, an algorithm is presented to resolve the issue. Occlusion pipeline is based on a new marker design, which places some textures in the untapped frame that will be tracked during the marker occlusion. This solution can also be adapted to any marker tracking system that uses its central area to codify the digital identification.

To exploit the extra information provided by the presence of textures, two different tracking methods have been proposed to compute the camera pose. The first is a fast method based on temporal coherence, which works correctly as long as no fast camera movements take place (FtF). If this method fails, another tracking method is called on (TbD), which is computationally more expensive but supports rapid camera movements. Experimental results illustrate the robustness of the combination of these techniques, offering a real-time solution (with non-optimized code). Indeed, the proposed tracking outperforms other solutions, because it does not suffer from drift. An in-depth study of the parametrization of these tracking methods has also been developed to show their response against different configurations and to obtain the best balance between robustness and performance. Additionally, two novel human-machine interfaces have been presented to show the new possibilities that have arisen as a result of obtaining more information during marker occlusions.

Non-Rigid Surface Tracking

*Aurkikuntzak bazterrak astindu zituen.
Mundu guztiak frogatu nahi zuen benetan
nota horietan kantatzen ote zuen kukuak.
Baina neurketak ez zetozen bat.
Bakoitzak bere egia zuen.*

KIRME URIBE

Part of this chapter has been published in:

Leizea, I., Álvarez, H., and Borro, D. "Real time non-rigid 3d surface tracking using particle filter". Computer Vision and Image Understanding, Vol. 133, N. 0, pp. 51–65. April, 2105.

4.1 Introduction

Recovering the 3D deformable shape of a non-rigid surface from a monocular video sequence is an ill-posed problem because of the depth ambiguities that exist in a single image. That is, many 3D surfaces could have the same projection (see Figure 4.1).

Even given the intrinsic camera parameters and a well-textured surface, it is still difficult to select the best mesh between all possible configurations of a deformable surface in order to solve the depth ambiguities. The resolution to this ambiguity normally requires prior knowledge of the most probable deformations that the surface can support.

Various methods to address this problem have been proposed in the literature. Along with the standard approaches that try to solve the shape recovery problem, it can be found approaches that establish prior knowledge (Zhu and Lyu, 2007), have a reference image (Salzmann and Fua, 2009), or are even based on a set of images of the target object (Brand, 2005).

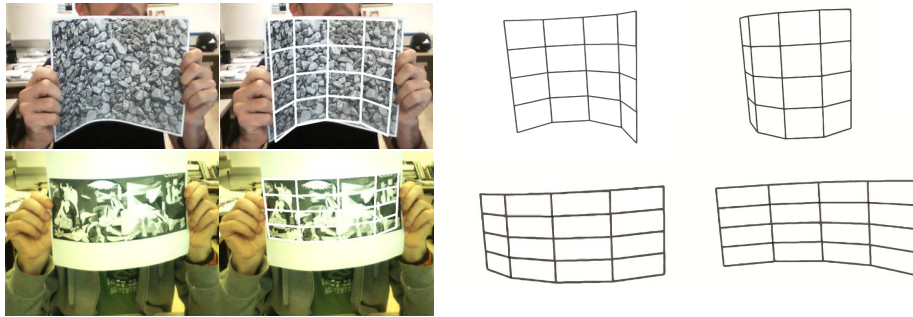


Figure 4.1: There are ambiguities when recovering a non-rigid 3D surface. The same original image can recover different meshes.

As an alternative to existing solutions, this chapter provides a novel solution that simultaneously recovers the non-rigid 3D shape and the camera pose in real time from a single image. The proposal relies on an efficient *Particle Filter (PF)* that performs an intelligent search of a database of deformations. Furthermore, an exhaustive *Design of Experiments (DoE)* to obtain the optimal parametrization of the PF is presented.

In the following sections an overview of some previous works related to recovering a deformable 3D surface will be enumerated. All the steps that have been carried out to address the problem of recovering a non-rigid 3D surface will then be detailed. Afterwards, a DoE is presented to determine the optimal values of the PF, as well as a test suite that demonstrates the visual quality and the good performance of the tracking method.

4.2 Previous Works

Some methods rely on modelling the physical properties of a surface to achieve an approximation of the physical behaviour (Fua and Leclerc, 1995), allowing *physics-based methods* to recover the 2D (Kass et al., 1988) or 3D (Terzopoulos et al., 1987) surface. Although one of the strengths of this type of method lies in obtaining accurate results, the challenge is to find the physical parameters that govern the surface behaviour. In addition, they have been limited to production and video games environments because they require complex designs such as finite element methods that imply high computational costs.

As an alternative to physical models, *learning-based models* adjust the behaviour of a surface to the movements registered in a database (Salzmann et al., 2005a), which contains the most representative deformations of the surface. This approach does not create a model based on the physical properties. Rather, the new model is generated according to the behaviour of one or more deformable objects. This approach has proved to be very effective when a valid database is available, which is not always the case. In fact, this is the major difficulty with these techniques.

Other approaches like the *template-based model* (Moreno-Noguer et al., 2009), reconstruct the deformation of a non-rigid surface by using 2D-3D correspondences between an input image and a reference image in which the shape and the intrinsic parameters of the camera are known, i.e., they fit the model to a set of visual cues extracted from the image. However, it is still necessary to use additional constraints that ensure the consistency of the reconstructed surface.

More recently, *Non-Rigid Structure from Motion (NRSfM)* methods (Bartoli and Olsen, 2007) have been proposed, which unlike the template-based models do not require prior knowledge. NRSfM receive multiple images of the target object, taken over time, generally in the form of a video sequence. Thus, they extract frame-to-frame 2D-3D correspondences by tracking points over the sequence, and seek to recover the 3D locations of the individual feature points in each input image. The main drawback with this procedure arises from the need to have a good textured surface.

4.3 Overview

As explained above, deformable techniques have their own disadvantages, making them specific to a particular field rather than being more generally applicable. In the template-based methods, for example, there are only a few approaches that are able to reconstruct the pose and the non-rigid shape simultaneously (Sánchez-Riera et al., 2010; Moreno-Noguer and Porta, 2011). Furthermore, even though in many approaches accurate 3D reconstructions are obtained, the computational cost prevents them from tracking deformable objects in real time. What is more, amongst the approaches that are able to work in real time, only 2D surface registrations are performed, which causes certain ambiguities when determining the 3D structure. The drawback of learning-based methods is that an appropriate database that includes the most significant deformation modes of the surface is required, which generally involves a laborious manual process. In addition, despite the fact that many deformable techniques solve the problem with accurate results, in most cases they are costly to implement and have a high computational cost.

For all these reasons, this section presents an approximation which is able to simultaneously reconstruct a non-rigid 3D shape and calculate the pose in real time. It is fully automatic and does not require manual intervention. Additionally, it is highly parallelizable, supporting efficient implementation. And although the surface variation is valid only for a limited *Degrees of Freedom (DoF)*, it works in an easier regime because it does not require to achieve the same level of complexity as other approaches (Salzmann et al., 2007a).

4.4 Simultaneous Pose and Non-Rigid Surface Recovery

In order to accomplish the recovery in each frame of both the pose and the non-rigid 3D shape of the target surface, a 3D PF has been designed to continuously update the camera pose and recover the 3D surface. The experimental studies in Section 4.5 demonstrate that the proposed method can reliably recover 3D structures of surfaces and the camera pose with low computational cost.

The framework of the system is divided into two phases: *offline* (see Section 4.4.1) and *online* (see Section 4.4.2).

4.4.1 Offline Phase

The main goal of the offline phase is to build the two principal databases required for the online phase. It is worth noting that this training phase is executed only once for each texture.

The first database, referred to as the *appearance database* (see Section 4.4.1.1), is used to perform the *initialization step*, i.e., to solve the first camera pose. This method is based on appearance, so it requires a database of feature descriptors that belongs to the training texture. It consists in detecting a set of 2D features from the reference image along a set of keyframes. Once this has been carried out, the corresponding 3D values are obtained through a back-projection. Together with the appearance descriptors these are then stored in the database.

The second database, the *deformation database* (see Section 4.4.1.2), is related to the *Frame-to-Frame tracking (FtF)* method that is based on temporal coherence and provides a fast and robust tracking against non-strong camera movements. At this stage, a set of deformations that represents the behaviour of the surface is stored. This can be done by applying different types of deformations to the original state of the surface.

4.4.1.1 Appearance Database

The aim of this database is to store a set of reference features (with known 3D coordinates) together with their corresponding image descriptors (see Figure 4.2), which is a well-known procedure (Xu et al., 2008). Therefore,

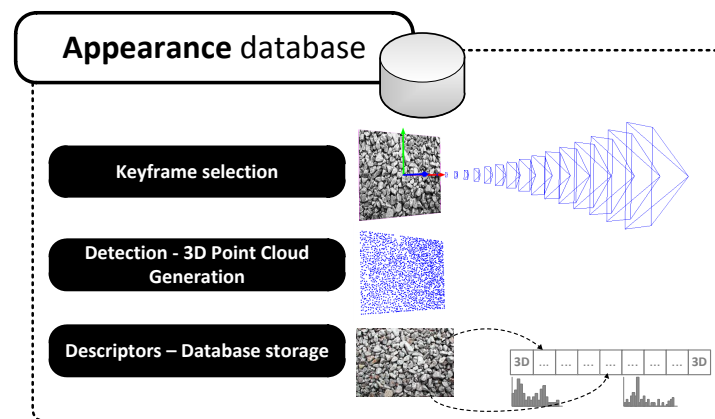


Figure 4.2: Overview of the appearance database pipeline. The texture is trained along a set of keyframes. These keyframes are processed, and the 2D features that belong to the surface are back-projected, obtaining their corresponding 3D values. These 3D points and their corresponding descriptors are indexed in a database, which is used to find a set of matches during the online phase.

following the procedure of previous chapters (see Section 3.2.3), the texture is trained along a set of keyframes and the simplified version of the original SIFT (Sánchez et al., 2010b) has been used in order to reduce the computational cost. Finally, all the reference 3D features with their corresponding descriptors are indexed in a database using a k -d tree, which allows 2D-3D matches between the input images and the reference features to be made efficiently and quickly during the tracking step.

4.4.1.2 Deformation Database

In contrast to the previous database, *the deformation database* relies on a basic knowledge of the representative deformations that the 3D model can support. Accordingly, a range of deformations is applied to the reference points that lie on the target surface (see the previous section). More specifically, this database relies on the proposal of (Hong and Chen, 2004), which presents a surface that produces a realistic page-turning effect by curling it. It is based on the behaviour of a sheet of paper, so the surface model preserves the look and the feel of the physical properties. In addition to applying deformations to the feature points,

the surface M represented as a rectangular 3D regular mesh (see Figure 4.3) divided into several planar patches (4x4) is also used in order to subdivide the problem into smaller ones (following the divide and conquer methodology). It then relies on a parallelization method that reduces the computational cost in the online phase.

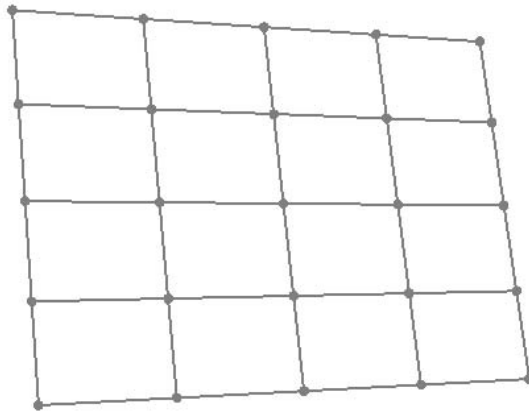


Figure 4.3: The surface of the mesh is represented as a rectangular 3D regular mesh.

In short, in order to recognize the simulated deformation that is closest to the current mesh in the online phase, a set of templates is stored in this database where each deformed mesh is expressed as the 3D position of the vertices. Below is a detailed summary of the steps that have to be followed to define it (see Figure 4.4).

Curling training. In this step some representative deformations of the original mesh are applied, i.e., deformation ranges are stored. To perform this step, a complete set of deformations that best represents the behaviour of a sheet of paper are selected. Each deformation is then classified as one of the following three types:

- **1-corner.** One corner of the sheet is folded. The surface is deformed by wrapping it around an imaginary cone model. This group includes those movements where a curl influences the whole side of the surface (see Figure 4.4 top left), like turning a page of a book. In particular, it is a process used in (Liesaputra, 2007) to make a deformation and a rotation around an imaginary cone model. It

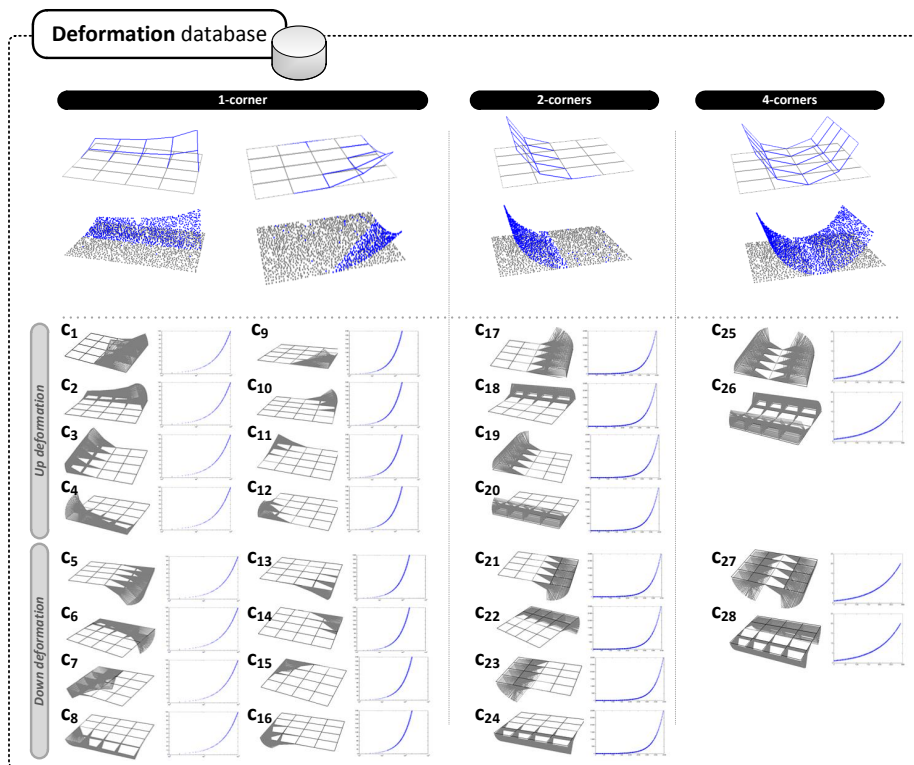


Figure 4.4: Overview of the deformation database. There are three types of deformations (*1-corner*, *2-corners* and *4-corners*). 16 clusters correspond to the *1-corner* type, 8 clusters to the *2-corners* type and 4 clusters to the *4-corners* type. Additionally each cluster has a deformation profile that represents the behaviour of the deformation.

also includes movements where the curl is a vertical movement that affects a small local area (see Figure 4.4 top left), i.e., a displacement is made from the corresponding corner to the centre of the surface. This method is called peeling and involves a process that is similar to the cone model although in this case the surface-turning point cannot be moved freely, i.e., the creased polygon is defined by a triangle.

- **2-corners.** Two adjacent corners are folded at the same time (see Figure 4.4 top middle). The surface is deformed by wrapping it around an imaginary cylindrical model.

- **4-corners.** The four corners of the sheet are folded at the same time (see Figure 4.4 top right). It also includes a cylindrical model to make the curling step and the wrapping generated on one side of the surface is applied equally to the opposite side.

Considering that a deformation degree is estimated by the maximum distance between the current position of the mesh (after applying a deformation) and its canonical location (at-rest position), the minimum and the maximum deformation degree for all types of deformation are set to 0 and 60 degrees of deformation, respectively. These values rely on the fact that higher deformations are not viable for image processing because if folds were performed on themselves, there would not be any visible internal feature, and consequently the image processing could not be done. Thus, the step between two neighbouring deformations depends on the number of folded templates, which is a user defined parameter that influences the computational cost (see Section 4.5).

The output of this module is a list of deformation templates divided into several clusters according to the type of curling movement. 28 different clusters have been identified (see Figure 4.4 down): 16 clusters that correspond to each corner of the first type of deformation (*1-corner*) with up and down deformations, 8 clusters with up and down deformations for each one of the second type (*2-corners*) and the last 4 clusters with up and down deformations for the last type (*4-corners*).

Deformation profiles. Since the behaviour of the implementation of the curling deformation that has been used is non-linear, an efficient mechanism has been created to simulate a linear response. It consists in generating deformation profiles for each cluster (see Figure 4.4 down), so that the deformation degree (the maximum distance between current and canonical position of the mesh - $\max_{1 \leq i \leq n} \|v_i - w_i\|$ where v corresponds to the deformed mesh and w the canonical mesh-) is expressed as a percentage (see Figure 4.5). This way, by knowing the deformation degree of the current mesh, in the online phase (see Section 4.4.2), the deformations that are in a range of proximity (expressed as a percentage) can be inferred, allowing a search for the same range of deformation to be performed, avoiding the non-linear behaviours.

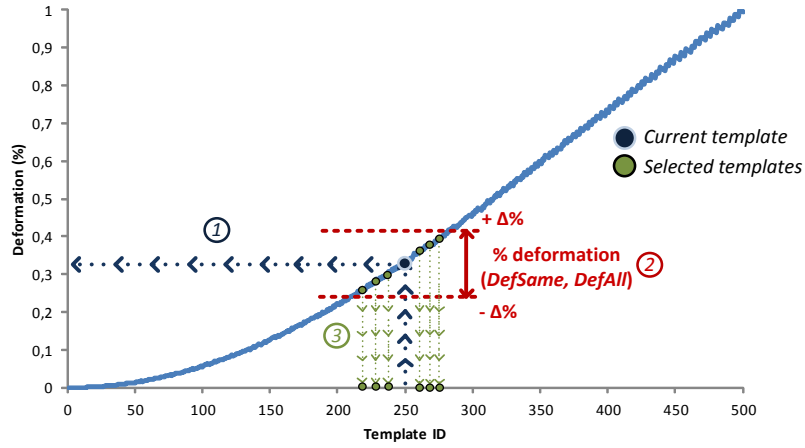


Figure 4.5: Deformation profile of one cluster. Each template of the entire database stores an identifier called *Template ID* and its corresponding deformation degree. Thus, all the clusters are ordered based on the deformation degree. Given the corresponding position (ID) of the current mesh (1), those templates that are in a close range (specified as percentage) of deformation (2) are obtained (3).

Furthermore, in an offline process all the similarities between all the templates of one cluster are precalculated against all the templates of the rest of clusters in order to obtain fast indexation in the online phase. Given a template from one cluster, it helps to determine which template in another cluster is more similar (see Equation 4.1). The main advantage of this technique is the cost reduction, which is obtained by means of queries made to the database, since the position of the current mesh is known. In this sense, the *similarity factor* between templates A and B is calculated by the distance of the vertices of each template.

$$T_i^{c_1} = \left\{ d_{t_i t_j}^{c_1 c_2}, d_{t_i t_j}^{c_1 c_3}, \dots, d_{t_i t_j}^{c_1 c_m}, \dots, d_{t_i t_j}^{c_1 c_{27}} \right\}$$

$$d_{t_i, t_j}^{c_l, c_m} = \sum_{k=1}^{n_v} \left\| v_k^{t_i} - v_k^{t_j} \right\|^2 \quad (4.1)$$

where $T_i^{c_1}$ is the collection of distances from the template i of cluster c_1 to the remaining clusters; and $d_{t_i, t_j}^{c_l, c_m}$ accumulates the Euclidean distances between the n_v vertices (v) of the templates (t) i and j that belong to the clusters l and m , respectively.

4.4.2 Online Phase

As mentioned in the introduction of this chapter, the main challenge facing non-rigid tracking methods is the combination of two requirements: that of calculating the pose while minimizing the reprojection error of the 2D-3D matches, and also the need to determine the best 3D point configuration, solving the ambiguity problem that arises from the fact that many shapes can give the same projection result. Therefore, this second phase detects the new shape of the mesh (the new 3D positions of the vertices) and calculates the correct pose that projects the corresponding 3D coordinates to the 2D cues detected in the input image.

As depicted in Figure 4.6, this phase combines two different tracking methods: *Initialization* and *Frame-to-Frame tracking (FtF)*. Both methods are combined to obtain a robust tracking. The FtF is used for as long as possible, and the *Initialization* step is used in the first frame or to reset from the FtF. It requires the surface to be at rest position, thus solving the detection problem such as a rigid alignment and approaching the visual tracking problem in the same way as other authors such as (Pilet and Saito, 2010).

4.4.2.1 Initialization

The main goal of this solution is the initialization of the data that will be used by the FtF. Let's say that the purpose is to find a planar texture in the image, i.e., a rigid problem (Lowe, 2004). The camera pose is computed at the beginning and when an error occurs. Therefore, the surface is treated as if it were a rigid planar surface. For that purpose, an incremental tracking similar to the one presented in (Wagner et al., 2008) is used. This process is divided into three parts: *obtain the pose*, *refine the pose* and *select the set of inlier correspondences*.

Obtain the pose. The input image goes through a similar detection process as explained in previous chapter (see Section 3.2.4.2). It consists in detecting a set of feature points with *Features from Accelerated Segment Test (FAST)* detector and computing their corresponding descriptors taking advantage of the simplified-SIFT method as is done in the offline phase (see Section 4.4.1). Once the current image features and the SIFT descriptors are computed, the matching process is based on the *K-Nearest Neighbour (K-NN)* approach as described in Section 3.2.4.2.

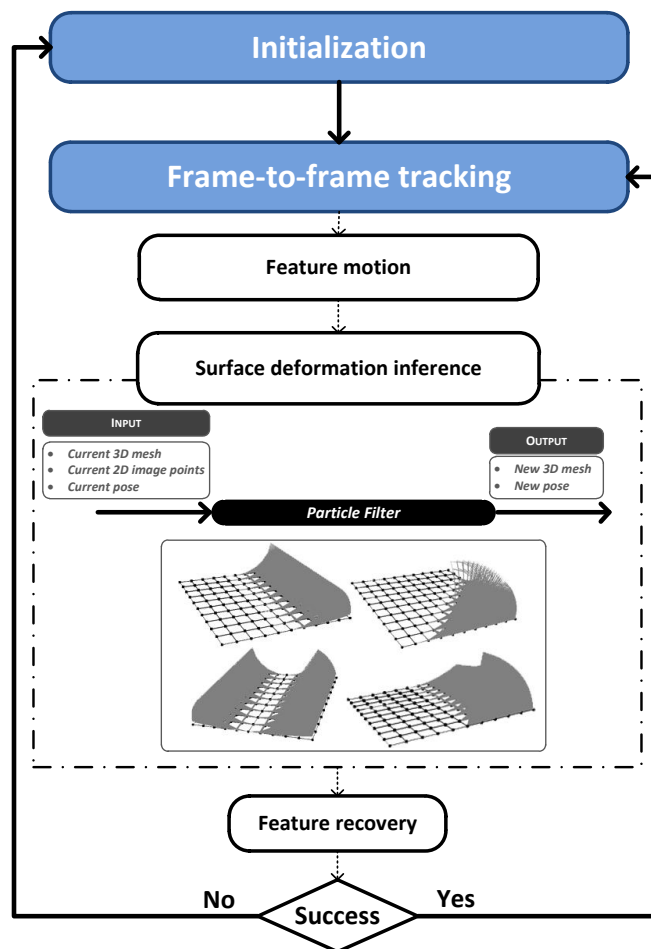


Figure 4.6: Online phase overview.

Following this, outlier filtering is executed to improve the matching step. Accordingly, the camera pose is computed (see Figure 4.7(b)) by using a robust hypothesize-and-verify *PROgressive SAMple Consensus (PROSAC)* method (Chum and Matas, 2005), which returns the best homography. This procedure selects the inlier matches depending on the reprojection error for the calculated homography. Among the underlying properties involved, it is worth noting the restrictions of the homography generation which establish that the points cannot be collinear and the

maximum number of matches to speed up the development.

Refine the pose. This second step is designed to further improve the accuracy of the camera pose. For this, the input image is warped (see Figure 4.7(c)) using the latter estimated homography, thereby helping to find more accurate homography transformation on the basis of similarity to the training texture. Hence, a new homography transformation $H_{refined}$ is obtained and combined with H_{prosac} through the matrix product (see Figure 4.7(d)). The resultant matrix H_{result} is considered to be correct if at least 25 percent of the matches are inliers.

Matching refinement. Finally, in order to initialize FtF with the highest number of features, a list of the correct matches is created at this last stage. All the 3D points of the training database are projected with the new camera pose and associated to the closest detected feature point if the distance is below a predefined threshold. An additional condition that restricts the new phase begins if the points in each patch do not exceed at least 50 percent of the features detected in the corresponding patch from the stored ones in the database.

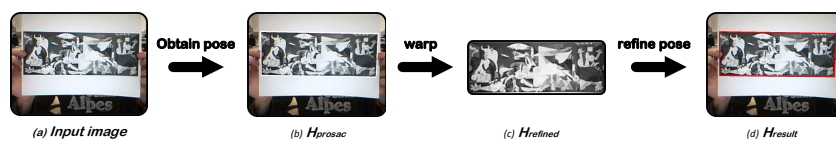


Figure 4.7: Initialization overview. The camera pose H_{prosac} is computed using the PROSAC method. The input image is then warped to find a more accurate homography transformation $H_{refined}$. And finally H_{result} is calculated by the combination of the homographies.

4.4.2.2 Frame-to-Frame Tracking

The pose and the 3D mesh calculated in the previous step, serve as input for this second stage of tracking. The FtF technique is used to recover the camera pose and 3D surface through the motion property. The FtF is divided into different steps: *feature motion*, *deformation inference* and *feature recovery*.

Feature motion. Lucas-Kanade optical-flow method (Lucas and Kanade, 1981) is used to estimate the movement of the points that lie on the texture between two successive frames. More specifically, to reduce the computational cost, the optical flow of the most significant points (those that have the strongest FAST response) is extracted only for each patch of the texture. The movements of the remaining points are inferred by applying the movement of the corresponding representative point. Furthermore, the new predicted locations are also associated with those points detected in the current input image (FAST detector) to avoid drift.

In order to avoid outlier points, affine constraints movements are set to the points of the surface. The study is made using the patches fixed in the offline phase. Thus, the pose of each planar patch is formulated as a rigid problem. This method obtains an affine transformation that maps points on one plane (the planar patch of the previous frames points) to points on another plane (the plane formed by the new image features). If the number of points is below a predefined threshold (15 points in the experiments), the patch is discarded. Then, *RANdom Sample and Consensus (RANSAC)* (Fischler and Bolles, 1981) is used to find the transformation that solves each local rigid problem, accepting only those patches that have 60 percent of inliers. Furthermore, the vertices shared between neighbouring patches should not exceed a threshold (in this case 10 pixels). As a consequence, the points that do not fit well with each of the neighbouring points, are deleted. Consequently, the filtering step is considered good as long as at least 25 percent of the patches have a good pose.

Surface deformation inference. Once robust points have been calculated, a PF is used to recover the new pose and the new 3D mesh. Given the current 2D-3D correspondences and the previous pose, the PF updates the camera pose and the shape of the surface. It is noticeable that the PF selects fewer correspondences from the centre of the mesh (inner patches), as it is a correct way to approximate the real behaviour of the model with a low computational cost. This idea is supported by the results that are shown in the experiments in Section 4.5.

The PF is therefore provided by the pose, the 2D image points and the 3D positions of the vertices of the current mesh. The key idea of

the PF is to detect a mesh stored in the database together with a pose that best resembles the input image. The functionality offered by the PF system as well as the way in which it is applied is detailed below.

Particle Filter

PFs belong to the family of *Sequential Monte Carlo (SMC)* methods, and are robust against non-static scenes in which multi-modality is likely (Pupilli and Calway, 2005). The main characteristic is that it represents the movements through a set of weighted samples known as *particles*. More detailed information about PFs can be found in (Arulampalam et al., 2002b; Salih and Malik, 2011). The use of PF is applied in many areas in the field of *Computer Vision (CV)*. One of the most noteworthy, is the head pose estimation and tracking. A more extended survey of the use of this type of techniques can be found in (Murphy-Chutorian and Trivedi, 2009).

PF is divided into two stages: *particle generation* and *particle evaluation*. In the *particle generation* step, the previous camera pose is perturbed in order to generate multiple camera pose candidates for the current frame (particles). The posterior density of the motion parameters is thus approximated by a set of particles. The *particle evaluation* step is then responsible for assigning a weight to each particle and selecting the correct one.

In this particular case, just perturbing the pose is not enough. This practice is well suited for rigid bodies, but a deformable object requires the vertices of the mesh as well as the pose to be perturbed at the same time. The particle generation step thus consists in perturbing the mesh (*perturb shape*) and the pose (*perturb pose*):

1. Perturb shape. In this perturbation step (see Figure 4.8) a filter is set for the *deformation database* in order to predict the suitable deformations. Instead of searching through the entire database (which would involve a very high computational cost), an intelligent search is made. Given the degree of deformation of the current template and pointing it in the correct position of the cluster it belongs to (see Figure 4.5(1)), the search is performed in the closer templates (in terms of degree of deformation). This search is divided

into two parts, a first search through the templates for the same type of deformation (same cluster - *DefSame*), followed by a second search that involves selecting the templates with different types of deformation (remaining clusters). In this case, the template selection starts at the corresponding position in each cluster (using the *similarity factor* (see Equation 4.1) to know which the most similar template for each remaining clusters is). A different deformation step (*DefAll*) is used here.

Given the total number of particles to process (*NumParticles*) as input, the number of particles that are assigned to the same cluster is proportional to the degree of deformation. Thus, the greater the degree of deformation, the higher the number of particles for the same cluster, since in cases where the deformation is strong, it is difficult to change to another type of deformation (another cluster).

2. Perturb pose. The pose perturbation is generated using the 3D meshes obtained in the shape perturbation step. However, all the particles are not processed from the former step, as the computational cost involved would be very high. The particles are grouped with a similar degree of deformation (*groupDef*) in each cluster (see Figure 4.9). Thus, only the new pose for those particles that is representative for each group is calculated through an efficient PnP¹ algorithm (Lepetit et al., 2009). The poses of the remaining particles are then copied from the closest representative particle.

After finishing the particle generation process, the particle evaluation phase weights each particle in order to subsequently select the best one. It is assumed that each particle is formed by the set of vertices that compose the 3D mesh, the camera pose and its associated weight. The likelihood (w) of each particle (P_i) is proportional to the percentage of inliers (those correspondences with a reprojection error lower than a predefined threshold; ~ 9 pixels for the experiments) (see Equation 4.2).

¹The aim of the *Perspective-n-Point (PnP)* problem is to determine the position and orientation of a camera given its intrinsic parameters and a set of n correspondences between 3D points and their 2D projections

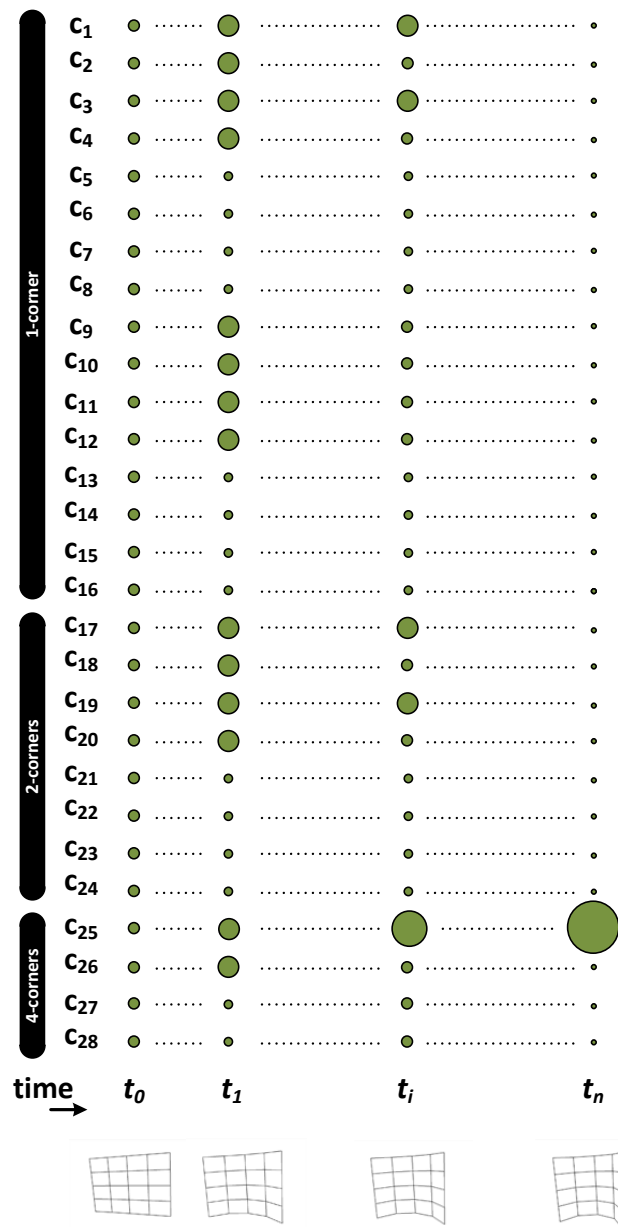


Figure 4.8: Particle generation process and particle evaluation phase over time. At one point in time, the evolution of the particles of each cluster is shown.

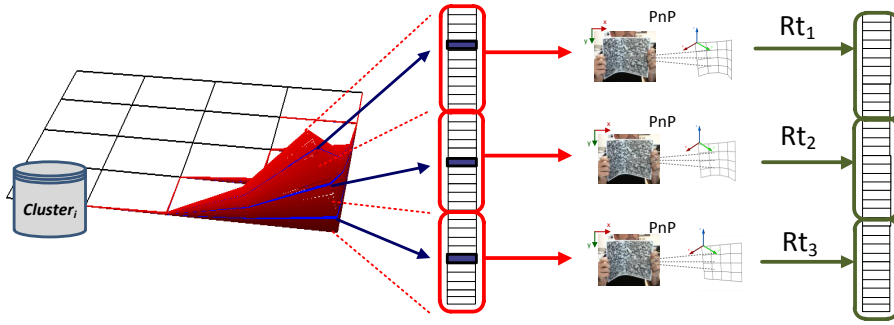


Figure 4.9: Templates are grouped (identified by red circle) based on the similarity of the deformation degree ($groupDef$) and each group is represented by a particle (located in the middle, blue cell). The pose of this particle (middle diagram) is applied to the other particles of the group (identified by green circle).

$$w = \sum_{p \in patches} \frac{f_p}{N_p} \text{ and} \quad (4.2)$$

$$f_p = \sum_j \left(\left\| \begin{pmatrix} u_j \\ v_j \end{pmatrix} - KR_t \vec{v}_j \right\| < thr \right)$$

where f_p is the number of inliers related to patch p and N_p is the number of points in patch p . An inlier is defined by f where (u_j, v_j) are the 2D image position of the visual cue j , \vec{v}_j are their corresponding 3D coordinates, R_t the current camera pose and K represents the intrinsic parameters that describe the characteristics of the camera (focal length, skew, etc.), which can be extracted through a camera calibration process.

The results achieved are satisfactory enough (see Section 4.5). The idea of introducing a new term to preserve the length of the edges has thus been rejected because it would increase the computational cost. Moreover, the templates generated in the offline phase offers a close approximation to the real physical properties of the model, so they impose restrictions inherently.

Finally, to set the inliers and refuse the outliers after the PF evaluation, the new 3D points are projected with the new pose and checked against the FAST points from the detection phase. If the distance to the closest FAST point is higher than ~ 7 pixels, the current point is set as outlier and the list of points for the next frame is updated.

Feature recovery. This last phase is called in when the number of current tracked points is below a threshold. Two major restore stages can be distinguished. The first stage recovers points along the whole mesh (a high number of points are recovered), while the other stage is more specific to each patch that composes the mesh (in this case only the points of a specific patch are recovered). The procedure is the same for both stages. It goes over the list of points detected in the initial state (*Initialization* step), and if a point has not been tracked yet, it is projected with the new pose and a FAST point close to it is verified. If these conditions are met, then this point is added to the list of tracked points for the next frame.

4.5 Experiments and Results

This section presents the results achieved by the proposal. The DoE process that was used to deduce the optimal parametrization of the PF is also shown.

To perform the set of experiments a sheet of paper was used with different types of textures as depicted in Table 4.1.

4.5.0.3 Parametrization



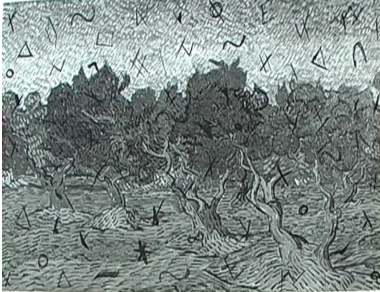
Several points of the methodology presented in (Tanco et al., 2009) were applied to deduce the optimal parametrization for the PF. The main goal is to find a robust, efficient and real-time PF through a DoE. Nowadays there is a variety of commercial software packages to help with the experimentation, especially for making calculations and creating graphs for the interpretation. In this instance, the statistical analysis software package Minitab was selected. The key steps of the DoE followed in order to obtain an optimal parametrization of the PF are outlined below. For more specific details, Appendix C analyses the steps that were carried out during the entire case study process as well as the theoretical method.

So, it is derived from the data in Appendix C that the variables shown in Table 4.2 are the most relevant ones in the PF. These variables are also known as *primary factors*. A factor is defined as a variable that can affect the *response*. In this case the response corresponds to two different output variables such as the execution time and error of the PF. Moreover, Table 4.2 shows the *p*-values associated to each one of the identified factors. This indicates the effect of all factors with regard to each of the two responses. Thus, an effect is considered as *significant* as long as its *p*-value is less than 0.05. In view of the results found in Appendix C it is deduced that the best parametrization is 3500 for *NumParticles*, 5 for *DefSame*, 5 for *DefAll*, 1 for *DefGroup* and 14000 for *Granularity*.

Furthermore, the following graphs (see Figures 4.10 and 4.11) show for each of the textures and setting the behaviour of the *NumParticles* and *Granularity* factors. *GroupDef*, in spite of being a significant factor, was not considered to perform a study of it, as well as those values that are not currently considered to be significant.

In the error case, looking at both texture graphs (see Figure 4.10 and 4.11 left), as the value of the *Number of Particles* and the *Granularity* increases the

Table 4.1: Different textures used in the experiments. The first column shows the texture, and the second column shows the number of detected points; the number of points in each patch and the percentage in relation to the total number of feature points. The colour red corresponds to a small number of features and blue and green correspond to a medium and high number of features, respectively.

Texture	Feature Points																
<p>Stones</p> 	<p>(1657)</p> <table border="1"> <tbody> <tr> <td>110 (6%)</td> <td>85 (5%)</td> <td>118 (7%)</td> <td>111 (6%)</td> </tr> <tr> <td>114 (7%)</td> <td>108 (6%)</td> <td>102 (6%)</td> <td>101 (6%)</td> </tr> <tr> <td>106 (6%)</td> <td>104 (6%)</td> <td>104 (6%)</td> <td>103 (6%)</td> </tr> <tr> <td>106 (6%)</td> <td>85 (5%)</td> <td>100 (6%)</td> <td>100 (6%)</td> </tr> </tbody> </table>	110 (6%)	85 (5%)	118 (7%)	111 (6%)	114 (7%)	108 (6%)	102 (6%)	101 (6%)	106 (6%)	104 (6%)	104 (6%)	103 (6%)	106 (6%)	85 (5%)	100 (6%)	100 (6%)
110 (6%)	85 (5%)	118 (7%)	111 (6%)														
114 (7%)	108 (6%)	102 (6%)	101 (6%)														
106 (6%)	104 (6%)	104 (6%)	103 (6%)														
106 (6%)	85 (5%)	100 (6%)	100 (6%)														
<p>Guernica</p> 	<p>(870)</p> <table border="1"> <tbody> <tr> <td>48 (5%)</td> <td>70 (7%)</td> <td>33 (4%)</td> <td>57 (6%)</td> </tr> <tr> <td>35 (4%)</td> <td>58 (7%)</td> <td>33 (4%)</td> <td>55 (6%)</td> </tr> <tr> <td>71 (8%)</td> <td>81 (9%)</td> <td>50 (6%)</td> <td>24 (3%)</td> </tr> <tr> <td>76 (9%)</td> <td>68 (8%)</td> <td>53 (6%)</td> <td>56 (6%)</td> </tr> </tbody> </table>	48 (5%)	70 (7%)	33 (4%)	57 (6%)	35 (4%)	58 (7%)	33 (4%)	55 (6%)	71 (8%)	81 (9%)	50 (6%)	24 (3%)	76 (9%)	68 (8%)	53 (6%)	56 (6%)
48 (5%)	70 (7%)	33 (4%)	57 (6%)														
35 (4%)	58 (7%)	33 (4%)	55 (6%)														
71 (8%)	81 (9%)	50 (6%)	24 (3%)														
76 (9%)	68 (8%)	53 (6%)	56 (6%)														
<p>Salzmann</p> 	<p>(1630)</p> <table border="1"> <tbody> <tr> <td>108 (6%)</td> <td>117 (7%)</td> <td>121 (7%)</td> <td>134 (8%)</td> </tr> <tr> <td>99 (6%)</td> <td>108 (6%)</td> <td>88 (5%)</td> <td>112 (7%)</td> </tr> <tr> <td>99 (6%)</td> <td>101 (6%)</td> <td>107 (6%)</td> <td>101 (6%)</td> </tr> <tr> <td>87 (5%)</td> <td>76 (5%)</td> <td>84 (5%)</td> <td>88 (5%)</td> </tr> </tbody> </table>	108 (6%)	117 (7%)	121 (7%)	134 (8%)	99 (6%)	108 (6%)	88 (5%)	112 (7%)	99 (6%)	101 (6%)	107 (6%)	101 (6%)	87 (5%)	76 (5%)	84 (5%)	88 (5%)
108 (6%)	117 (7%)	121 (7%)	134 (8%)														
99 (6%)	108 (6%)	88 (5%)	112 (7%)														
99 (6%)	101 (6%)	107 (6%)	101 (6%)														
87 (5%)	76 (5%)	84 (5%)	88 (5%)														

error values decrease. An error can sometimes be seen to fluctuate. This is due to the fact that the parametric configuration allows some particles in the PF to be introduced while others are not. It may therefore happen that the particle that best fits the feature points is not being used. This is also why the interactions between different factors are given strategic importance, allowing for the possible

Table 4.2: *Analysis of Variance (ANOVA) for error and time.*

Texture	FACTOR	p -value error	p -value time
Stones	<i>NumParticles</i> (A)	0.060	0.000
	<i>DefSame</i> (B)	0.037	0.431
	<i>DefAll</i> (C)	0.678	0.000
	<i>GroupDef</i> (D)	0.000	0.000
	<i>Granularity</i> (E)	0.000	0.000
Guernica	<i>NumParticles</i> (A)	0.000	0.000
	<i>DefSame</i> (B)	0.598	0.941
	<i>DefAll</i> (C)	0.621	0.000
	<i>GroupDef</i> (D)	0.000	0.000
	<i>Granularity</i> (E)	0.000	0.000

appearance of some noisy factors that cannot be controlled. Nevertheless, the overall performance is always descendent. It should also be mentioned that the behaviour from 6000 number of particles and 8000 of granularity, the error values are no longer decreasing because it achieves a saturation state where the number of particles and the granularity do not affect the response. Similarly but in the converse sense occurs in the time response (see Figure 4.10 and 4.11 right). The reaction in the number of particles increases at all times while in the granularity case, starting from the 6000, time remains constant

4.5.1 Experiments

This section presents the results obtained with the proposed method and its optimal parametrization. A hardware set-up that consisted of an Intel Core 2-Quad Q9550 at 2.83 GHz and 3 GB of RAM with a Logitech QuickCam Connect webcam was used to obtain the results.

4.5.1.1 Robustness

The first type of experiments involved applying a Gaussian noise with a normal distribution to the acquired matches. These matches are those that are provided to the PF. This experiment thus allows us to evaluate the robustness of the PF results. Three main tests were developed as follows.

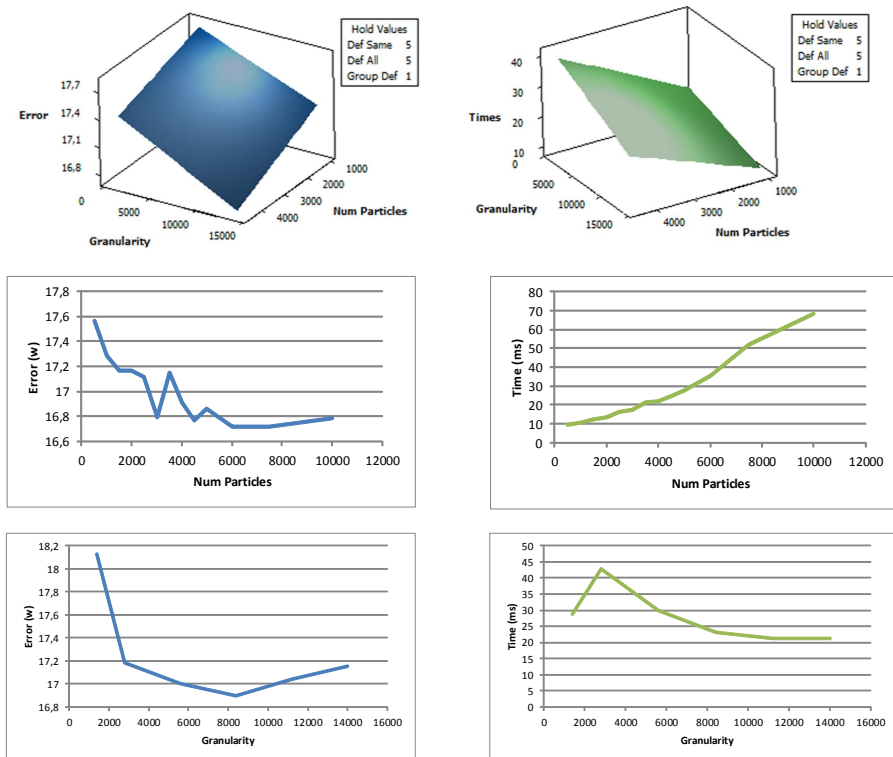


Figure 4.10: *Guernica* factors behaviour. The behaviour of the *Granularity* factor and *NumParticles* are studied, fixing the factors which are not considered as significant (with the exception of the *GroupDef*).

The first test consisted in introducing noise in the input data. More specifically, a Gaussian noise from 5 to 20 pixels was set to a percentage number of random matches that varied from 1 to 20. By analysing the results plotted in Figure 4.12, it can be stated that the PF kept the reprojection error (in pixels) at reasonable levels despite the presence of some outliers. Thus, it could clearly be seen that once the noise exceeded 10 pixels the results started deteriorating. For example, with 15 and 20 of noise for the 15 percent of matches, the reprojection errors are 5.11 and 7.72, respectively. Moreover, there are no more results presented when the noise is higher than 20 pixels because from there on, the deformations are degraded as can be seen in the last example of

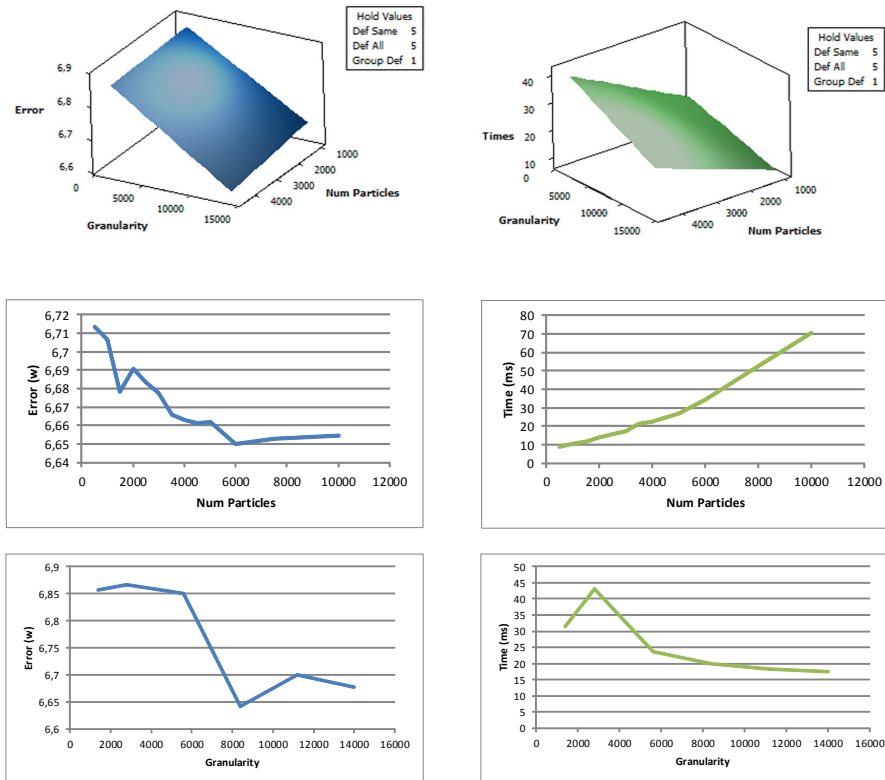


Figure 4.11: *Stones* factors behaviour. The behaviour of the *Granularity* factor and *NumParticles* are studied, fixing the factors which are not considered as significant (with the exception of the *GroupDef*).

Figure 4.12. Nevertheless, although the results deteriorated when the number of noisy pixels exceeded 10, the projection was not bad at all and the PF deduced correctly the type of deformation that was being applied.

The second test introduced noise to all correspondences but within reasonable limits. Thus, the Gaussian noise varies from 1 to 5 pixels of standard deviation. The Graph 4.13 (left) shows that the reprojection error (measured in pixels) increases provided that the noise increases too. Even so, it should be noted that the results are sufficiently valid to obtain a correct visualization.

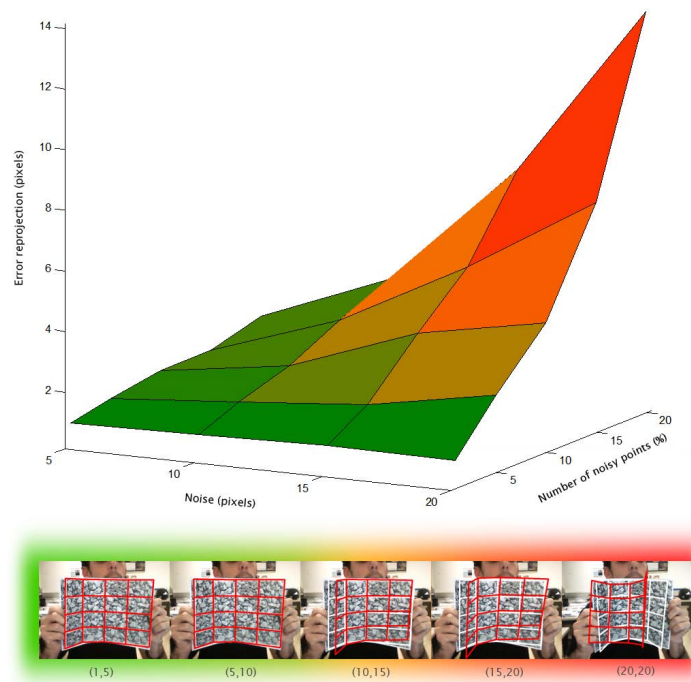


Figure 4.12: Reprojection error when input data is perturbed by adding Gaussian noise. The white mesh of the photography represents accurate results while the red represents the noisy results. The resultant examples are detailed with their corresponding configuration (i, j) , where i is the number of noisy points and j represents the noise.

And finally, the last type of evaluation of performance consisted in introducing a small noise to the inliers while for the considered outlier correspondences (randomly selected as in the first test) a very large noise was added, in this case 20 pixels of standard deviation. Figure 4.13 (right) indicates the results for each of the percentages of matches. Unlike the first test where the noise was only added to various matches, in this case the error is larger but did not differ a great deal from the first. For instance, for the case where the number of selected matches was 15 and the noise 20, the error was about 10.08 while in the first test achieved 7.72 pixels.

It is equally important to highlight the fact that the points most affected by the noise are those that suffer a deformation. In other words, the majority of the

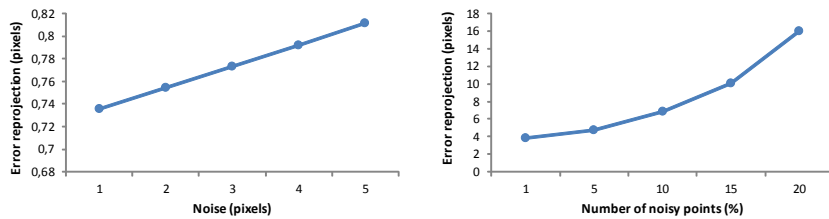


Figure 4.13: Reprojection error when input data (all correspondences) is perturbed by adding Gaussian noise (a) and reprojection error when introducing small amount of noise to the inliers and very large noise to the outliers from the input data (b).

points that are located in the centre of the texture, practically are not altered since it can be considered as a rigid problem due to little or no deformation.

4.5.1.2 Visual Quality

Figure 4.14 shows the correct visual output obtained by this method on each of the three texture examples (*Stones*, *Guernica* and *Salzmann*). The *Stones* example represents the reconstruction of a deforming sheet of paper after a video sequence of 89 frames, while the *Guernica* and *Salzmann* examples performed the reconstruction after a video sequence of 51 and 71 frames, respectively. These texture examples were selected based on the uniformity of the features detected in each. *Stones* is composed by a large number of features distributed in a uniform way, while *Guernica* has both fewer points and, a more irregular distribution. *Salzmann*, in turn, looks like *Stones*. The video sequence² was taken from (Salzmann et al., 2007a). It was used to compare the quality of this proposal against existing solutions and to check the visual behaviour of this method in an environment that is equal to that used in other works. It should be noted that the results of this proposal are as good as those provided by other approaches.

Finally, a set of experiments with materials such as clothes has accordingly been performed. Even though false positives may appear (middle Figure 4.16), sometimes the results are visually acceptable (on the right of Figure 4.16, although a forward movement is being applied, the model fits backward). In

²Paper_bend.zip example from <http://cvlab.epfl.ch/data/dsr/>

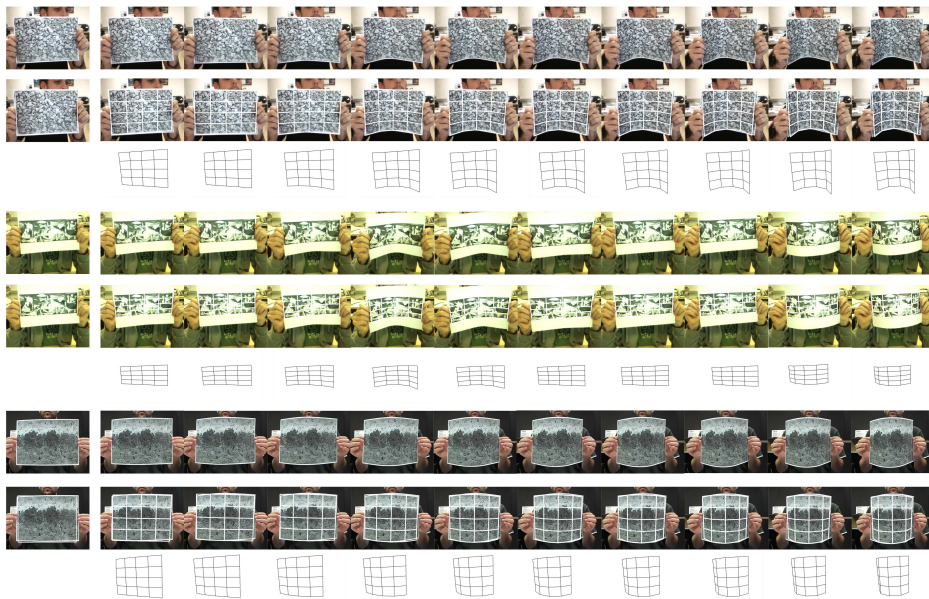


Figure 4.14: Results of 3D recovering surfaces. These are the three experiments performed: *Stones* (top), *Guernica* (middle) and *Salzmann* (bottom). The following information is presented for each of the three examples: the original image (first row), the projection of the recovered 3D mesh in that image (second row) and the recovered 3D mesh (third row).

other cases, though there are many deformation possibilities, the system returns satisfactory results (see Figure 4.15).

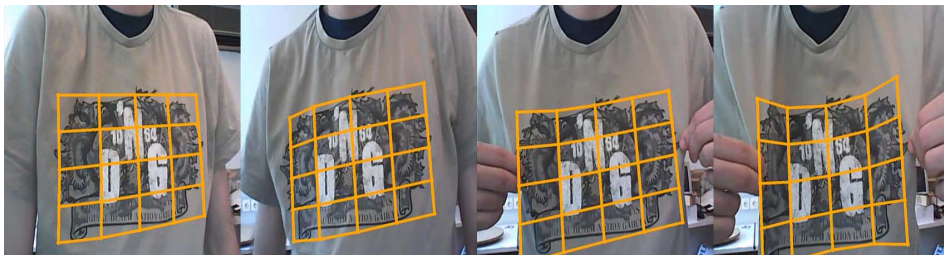


Figure 4.15: Response of the proposed recovering non-rigid 3D surface for clothes model.

4.5.1.3 Performance

The PF was also tested with different hardware configurations. Three different PCs were used: an Intel Core 2-Quad Q9550 at 2.83 GHz and with 4 GB of RAM, an Intel Core 2-Duo E8500 at 3.16 GHz and with 3 GB of RAM and an Intel Core i7-930 2.8 GHz and with 3 GB of RAM. Table 4.3 presents the times for the PF method and the FtF step. The execution time of the FtF includes both the visual detection and the PF method.

Table 4.3: Execution times (in *ms*) for FtF and PF using different PCs.

	Frame-to-Frame tracking	Particle Filter
<i>Core 2-Duo</i>	48.81	33.62
<i>Core 2-Quad</i>	34.61	19.72
<i>Core i7-930</i>	30.94	17.23

The analysis of these values allows inferring that this method executes in real time and is highly parallelizable, as demonstrated by the better results obtained with better configurations. Indeed, a future GPU implementation of this proposal will allow obtaining lower computation times, and consequently improve other aspects such as more detailed databases or better image processing.

Finally, regarding the Initialization step, it is also worth mentioning that the computational cost of the described stage runs in average time from ~ 75 *ms* to ~ 150 *ms* depending on the selected hardware configuration. Since it is a single computation of the camera pose at any given time, it does not affect the performance of the whole process, so it guarantees the execution in real time.

4.6 Discussion

The problem of recovering the 3D shape of a deformable surface has become an important issue in different areas such as *Computer Graphics (CG)* and *CV*. This chapter has presented a study of the different methods that can be found in the literature in order to solve the problem of the ambiguities that arise when reconstructing these types of surfaces.

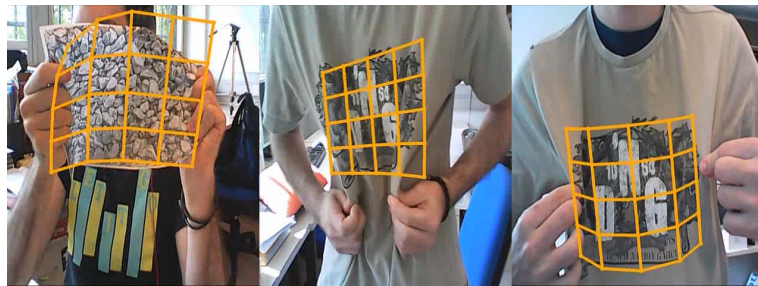


Figure 4.16: False positives for two types of textured planar models: sheet of paper and a shirt.

Since the range of deformations is limited, complex deformations may cause false positives (see Figure 4.16). If this happens, a simple shake will be enough to restart the recognition. As with the majority of the non-rigid tracking solutions that are based on storing a set of deformations in a database, the main problem is to include a feasible set of deformations that best fits the physical behaviour of the model. At the same time, it should be pointed out that extrapolating this methodology to other types of planar objects could be complex (see Figure 4.16 clothes examples). So, the creation of a more complex database, containing as the composition of various deformations, could solve this issue.

On the other hand, the composition of deformations is not addressed. Notice that the left example in Figure 4.16 fits to one of the two deformations that are being applied. By the term composition is meant the combination of various types of parametrization of deformations at the same time in the same 3D structure. The main goal of this solution is oriented to handle simple deformations that do not achieve a high degree of accuracy but instead correct visual feedback, i.e., achieve a real-time performance over accuracy sufficiently acceptable for the human eye. Incorporating composition of deformations will involve increasing the size of the database and, consequently, the computational cost would be higher as well as the reserved memory. In this respect, it shall seek an appropriate

balance between the performance of the system and the size of the database. Moreover, the presented category of clusters is considered complete enough to cover a set of simple deformations to meet the requirements mentioned above.

Additionally, the design selected to create the database based on clusters is applied in a supervised manner. In other words, in cases where the deformation is very low, the search is almost always performed along all the clusters giving similar probabilities. However, as the deformation achieves higher levels, the same cluster is given higher priority over the rest. This is based on the assumption that the greater the degree of deformation, the higher the probability of being close to the same cluster. By this assumption, when the deformation pattern (*1-corner*, *2-corners* and *4-corners*) begins to appear, the particles of the same cluster are more likely to be closer than the rest of the remaining clusters. This ensures that no abrupt changes appear with regard to the different parametrizations of deformations. Furthermore, the selected cluster structure together with the current function of similarity gives as a result the correct visual feedback and the real-time performance that is wanted to achieve.

Consequently, this framework is a complete solution that detects and tracks a texture in real time with a set of deformations that best fits the logical behaviour of a sheet of paper.

In addition, a refined DoE has allowed to deduce the optimal parametrization of a PF that achieves correct visual results in line with a low computational cost. The robustness of the system has also been tested by adding noisy matches and comparing it with other approaches.

Finally, Figure 4.17 shows the visual results from different viewpoints. It is not only check the output 3D reconstruction of the mesh from a frontoparallel view of the camera, but also from more side-views. In this sense, as can be appreciated in Figure 4.17, the side-view varies from medium (left) to extreme (right). Moreover, the error in pixels varies from 4.68 to 12 for the *Stones* example and from 4.7 to 11.8 for the *Guernica* one in the case of medium and extreme side-views, respectively. In conclusion, it is possible to check that when a deformation is applied from an extreme side-view of the camera a degree of occlusion of the texture is caused and consequently the visual feedback is not as good as being in the frontoparallel case. Even so, it can be seen that the visual feedback is correct enough for the human eye.

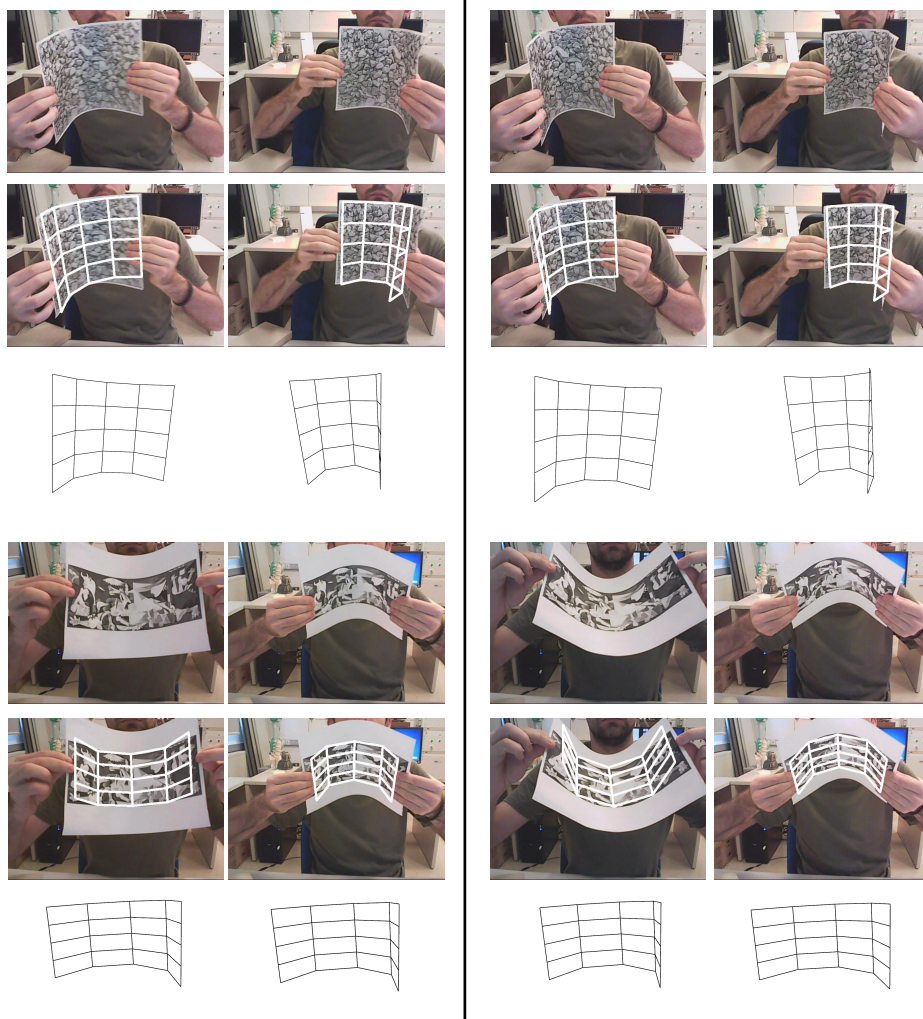


Figure 4.17: 3D reconstruction from different points of view. Different side-views of the camera for two different types of textures (*Stones* at the top and *Guernica* at the bottom). Medium (left) and extreme (right) side-views are shown.

Chapter 5

Deformable Object Tracking

*Para algunos, la vida es galopar un camino
empedrado de horas, minutos y segundos.
Yo más humilde soy, y sólo quiero que la ola que
surge del último suspiro de un segundo me
transporte mecido hasta el siguiente.*

LA MALA GANA, SANTOS ISIDRO SESEÑA

The content of this chapter has been published in:

Leizea, I., Álvarez, H., Aguinaga, I., and Borro, D. "Real-time deformation, registration and tracking of solids based on physical simulation". In Proceedings of the 13th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 165–170. Munich, Bavaria, Germany. September, 2014.

Leizea, I., Mendizabal, A., Álvarez, H., Aguinaga, I., Sánchez, E., and Borro, D. "Tracking of deformable objects based on a visual approach and physics simulation for robot-assisted surgery". Submitted to Computer Graphics and Applications, IEEE, 2015.

5.1 Introduction

Recovering the 3D shape of a non-rigid object presents the same challenge as occurs with the deformable surfaces (see previous Chapter 4). This challenge entails a highly ambiguous problem because many deformable objects (with completely different shapes) can have the same projection. In *Computer Vision (CV)* there are few solutions that solve the registration of a non-rigid 3D object and the number of approaches decreases when a real-time constraint is set. This is mainly caused by the complexity of the recognition and non-rigid registration steps.

Based on this assumption, there has been significant research in recent years in the field of tracking and recovering deformable objects. Many of these works (Haouchine et al., 2013a) are focused on very specific fields such as medicine, where the models addressed have a particular geometric appearance. Because of this narrow focus, all these approaches have difficulties dealing with objects that have different shapes.

With respect to image registration, two kinds of approaches can be found: those based on detecting features of the objects (Salzmann and Fua, 2011), and those that use optical-flow (Munoz et al., 2009) (e.g. determining the intensity differences between two consecutive images). Both techniques require that the objects be textured which is not always possible. In addition, these works are mainly focused on the robustness of determining the shape of the deformable object.

For all the reasons cited, this chapter presents two methods for registering deformations of 3D non-rigid objects. These solutions differ mainly in the physics formulation used to obtain a realistic behaviour of the deformations applied to the bodies. The first solution is based on a *Mass-Spring Model (MSM)* system in order to achieve a real-time non-rigid object registration method with a correct visual feedback. The second, is focused on a *Finite Element Method (FEM)* to achieve more accurate results than the former. Nevertheless, these two solutions share the same concept of tracking and registering 3D models regardless of their geometric shape. They can be used, for example, to provide visual feedback for applications such as an assembly of flexible components in the industry or even medical surgery. Additionally, the detection and tracking system is not based on features, but is based on templates instead (LINEMOD method (Hinterstoisser et al., 2011)). They can therefore operate with textured or untextured objects. This

makes these methods robust against sudden illumination changes. Furthermore, they not only obtain the deformed structure, but also the camera pose for each frame. In fact, these methods can handle large camera movements since, unlike other solutions, they do not require a static camera.

The following lines provide an overview of some works related to recovering a deformable 3D shape. The various steps that have been carried out to solve the problem of recovering a non-rigid 3D object and deducing the camera pose that best fits the projection for both types of physical formulation will be then detailed. The third section describes some experiments that validate the performance of the physical methods in terms of time consumption and stability, as well as giving a comparative account of the two methodologies. And finally, the last section presents the basis for the development of a surgical *Augmented Reality (AR)* system.

5.2 Previous Works

Obtaining the deformation of a non-rigid object is a complex task in the field of CV. Amongst the majority of approaches that recover the 3D structure of a non-rigid object; the problem is divided into two steps: *image registration* and *shape inference*. The first step deals with the resolution of the visual part of the problem (e.g. detecting the object in the image), while the second step deals with handling the deformation of the shape of the object.

As far as *image registration* is concerned, the issue can be tackled using two main approaches: *feature-based methods* or *direct methods*. *Feature-based methods* (Ahmed et al., 2008; Hayashi et al., 2012) focus on detecting points (also called *features*) that are distinguishable from their neighbourhood. Some correspondences are generated between a reference model and an input image to solve the registration problem. *Direct methods* (also known as *pixel methods*) (Jordt and Koch, 2011), in turn, use the intensity differences between two images to calculate the correspondences.

The *shape inference* process adjusts the acquired visual information to the shape. Physics-based systems adjust the shape according to its physical properties. Others (Brunet et al., 2011), however, integrate geometrical or temporal constraints to the system and solve it through optimization techniques such as *Second Order Cone Programming (SOCP)*. Similarly, there are approaches that learn the principal deformations of the objects to tackle the problem.

On the other hand, there are many ways to capture images. There are solutions where the image is captured by *monocular cameras* (Shen et al., 2010), *stereo cameras* (Haouchine et al., 2012) or more recently *RGB-D cameras*, such as the Microsoft Kinect. This last type of approach combines the colour information from the camera with the depth information obtained from an infrared sensor. The main advantage of the RGB-D approach relative to methods based on conventional cameras is that the former captures a depth map (a matrix that contains distance estimations for each pixel) in addition to conventional colour images. However, the depth images captured by these kind of devices contain noisy data and produce areas with no information.

Image registration as well as shape inference solutions are combined in several ways with different types of devices to capture images in many of the approaches presented below.

In terms of *feature-based vision methods*, various examples can be found in (Salzmann and Fua, 2011) and (Salzmann et al., 2007c). These works perform the reconstruction of non-rigid surfaces by learning the deformation-modes of the shape. They use *Principal Component Analysis (PCA)* to compute modes of deformation and generate a database of feasible shapes (Allen et al., 2003). In (Salzmann et al., 2007c), the 3D deformation modes are performed by varying angles between facets, while in (Salzmann and Fua, 2011), the mesh is subdivided into a set of patches that are combined linearly. In contrast, there are other solutions that do not obtain the deformation-modes of the shape but still solve the visual part by using a feature strategy (Schulman et al., 2013). In this sense, local rigidity constraints (Shen et al., 2010; Perriollat et al., 2011) or unconstrained quadratic optimization (Zhu et al., 2008) surface reconstruction can be carried out. However, the primary focus of many of these approaches (Salzmann et al., 2007c) is on planar surfaces. The main drawback being that most of them have a high computational cost.

Following the idea of using visual cues, there are real-time approaches like (Pilet et al., 2005), even some using RGB-D cameras (Hayashi et al., 2012; Shimizu et al., 2013), which solve the registration problem. Nonetheless, they are focused on textured surfaces such as T-shirts. Similarly, approaches like (Haouchine et al., 2013a; Haouchine et al., 2013b) work with stereo cameras and use a FEM formulation in order to capture the behaviour of the object and calculate the deformation. These approaches, however, are very oriented to applications in medical surgery, which usually involves specific formulation, for instance, the position of the camera should be static, and in some cases, they are not executed in real time (Haouchine et al., 2012). Furthermore, (Haouchine et al., 2014a) proposes a real-time method to register the non-linear elastic model deformations using the image points acquired from a monocular camera. However, this solution is based on an orthographic projection since the minimization is easier than with the perspective projection. Like the former approaches, (Shen et al., 2008; Shen et al., 2010) also use a stereo camera to simplify the recovery process, but instead of a physics-based model, they use iterative processing (Haehnel et al., 2003) or mathematical tools like SOCP which involve a complex optimization task to be carried out.

Another well-known technique that makes use of features is *Non-Rigid Structure from Motion (NRSfM)* (Bregler et al., 2000; Torresani et al., 2001; Torresani et al., 2003; Del Bue and Agapito, 2006; Del Bue et al., 2007; Fayad et al., 2010; Taylor et al., 2010; Russell et al., 2011), which is based on the factorization method (Tomasi and Kanade, 1992). In contrast to the conventional

feature-based methods cited above, in this case, it is not necessary to have a reference model because it simultaneously tracks and recovers the shape of non-rigid 3D surfaces. An example of a mechanical-based tracking method as (Agudo et al., 2012b), presents a FEM elastics thin-plate solid within a Bayesian *Extended Kalman Filter (EKF)* to estimate the deformation. The main drawbacks of NRSfM are that it requires good textured surfaces and that the amount of modes limits the deformations to linear modes, excluding non-linear deformation modes such as bending.

However, the main drawback in most of the above cases lies in the fact that they have to overcome several challenges such as working with a limited set of feature points.

In contrast to feature-based methods, *direct-based* or *Analysis-by-Synthesis (AbS) methods* (Jordt and Koch, 2011; Fugl et al., 2012; Jordt and Koch, 2013b) use the image data in combination with depth and colour information. In (Jordt and Koch, 2013a) the reconstruction of the model is based on a *Non-Uniform Rational B-Spline (NURBS)* based deformation. In contrast to physics-based methods, other approaches (Delingette et al., 1991; McInerney and Terzopoulos, 1993; Salzmann et al., 2007b; Salzmann et al., 2008b) use temporal constraints to minimize the error. There are also studies that focus on estimating the deformations for the human body (Koch, 1993; Shotton et al., 2013) or specialize in faces (Cai et al., 2010; Munoz et al., 2009). These methods are highly correlated with finding differences between two consecutive frames. As with most of the existing solutions in the literature, they are oriented toward cloth deformations (Rosenhahn et al., 2007; Hilsmann et al., 2010). Nevertheless, the main drawback of these methods is that they usually depend on a good initialization and require to make corrections over time in order to not be prone to drift.

5.3 Overview

Having presented the advantages and disadvantages of the existing techniques, the next sections present two methods with different physical formulations that are able to recover deformations of non-rigid 3D models regardless of their geometric shape.

The proposed methods not only obtain the deformed structure, but also take advantage of a template-based tracking method (LINEMOD method (Hinterstoisser et al., 2011)) to continuously update the camera pose. Unlike other solutions, these methods do not require a static camera. In fact, the methods handle large camera movements.

Additionally, the 3D registration and tracking of texture-less 3D objects is not based on features but templates, and can therefore operate with textured or untextured objects. For this purpose, an RGB-D camera that obtains both the colour and depth data is exploited, avoiding using other formulations that are not usually robust for texture-less surfaces or objects. This makes the solutions robust against sudden illumination changes.

5.4 Proposed Method using a Mass-Spring Model

Compared to existing solutions, this solution does not achieve the same level of accuracy. On the other hand, it returns correct visual results for the deformations of any kind of object (in terms of its geometrical shape) in real time.

For this purpose, the geometry of the model, represented as a triangulated mesh, is adjusted to the raw information acquired from an RGB-D camera. This triangle mesh in turn, is captured through a scanning process (see Figure 5.1) performed by the 3D Sense™ Scanner.

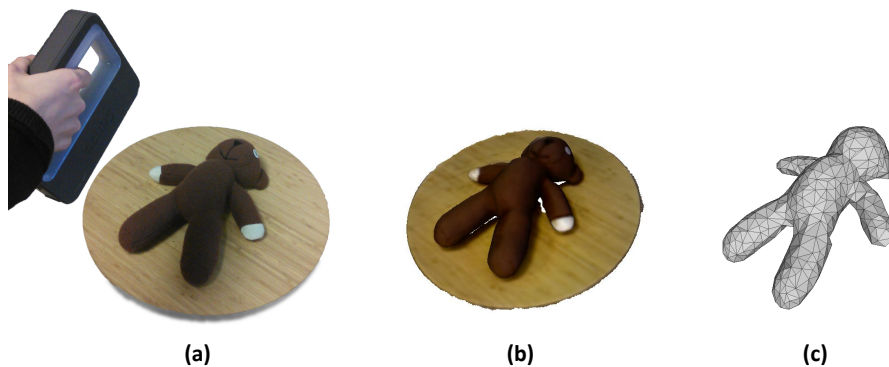


Figure 5.1: Triangle mesh acquisition. The object is scanned by the 3D Sense™ Scanner (a). Once the entire cloud is captured (b), the final mesh is computed as a triangle mesh (c).

The RGB-D camera obtains both the colour and depth data that is incomplete since it has a lot of noise with large holes (as Figure 5.2 shows) providing an incorrect visual feedback. In addition, the cited registration and tracking of texture-less 3D object system based on templates exploits the use of the RGB-D camera.

This camera information, in turn, is used to feed the input of a physics-based method, as is the case of MSM, which gives a realistic behaviour to the solid in order to obtain the deformations. Since the aim of the presented method is not to achieve an accurate deformation of the mesh, an MSM system has been chosen in order to obtain a visually correct solution that can be easily integrated with other AR technologies. Compared to other formulations such as FEM, MSM uses a simple structure and offers real-time performance (see Section 5.4.2). Thus, the experiments presented in Section 5.4.3 demonstrate that the method recovers the



Figure 5.2: The raw data acquired from the depth camera. The first column contains the original model while the adjacent columns present front and lateral views of the camera's raw data.

3D structure of a deformable solid with a low computational cost. Furthermore, this physical model supports multiple deformation modes at one time.

In conclusion, this chapter presents a complete framework for tracking and registering deformations of objects using a physics-based formulation. Figure 5.3 illustrates the framework which is divided into two main phases: *preprocessing* and *online execution*. These two phases, in turn, are subdivided into two main stages. The first focuses on the process of detecting and tracking rigid objects (presented in Section 5.4.1). To this end, it takes advantage of an RGB-D camera

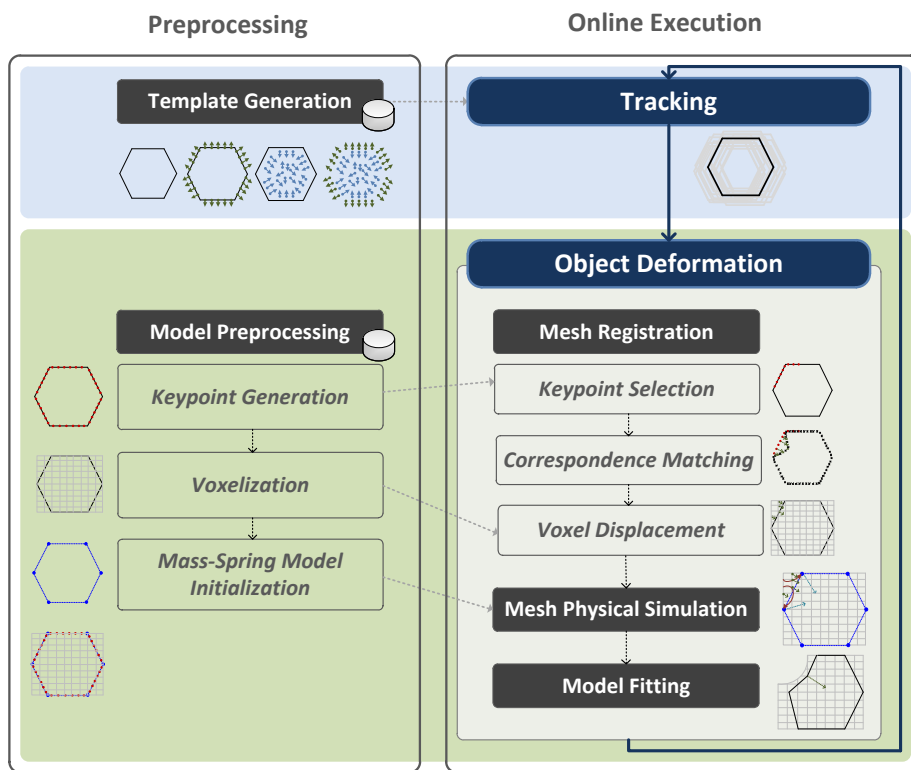


Figure 5.3: Overview of the proposed method (preprocess and online phases).

to obtain both the colour and depth information. The second stage retrieves the deformations of the objects (explained in Section 5.4.2), i.e., integrates the colour and depth information into a physical model. Therefore, each of these two steps generates preprocess information that is subsequently used in the online phase. Following is a description of each step.

5.4.1 Tracking

The objective of this phase is to recognize and track 3D objects from a point cloud captured by a camera. The camera pose is obtained for each frame, as it would be with a rigid solid problem. A template-based LINEMOD approach

(Hinterstoisser et al., 2011; Hinterstoisser et al., 2013) is used, as it guarantees the detection of texture-less objects in cluttered scenes.

This method requires an RGB-D device and is divided into two major phases: *learning* and *detecting* the objects. The following overview is a brief description of this approach for the sake of completeness. More detailed information can be found in (Hinterstoisser et al., 2013).

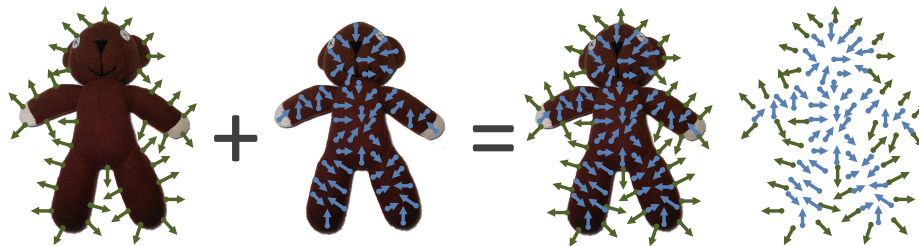


Figure 5.4: Template Learning. Two types of features are detected to generate a template: the colour gradients from the silhouette of the shape and the surface normals from the interior of the shape.

5.4.1.1 Template Generation

This first phase trains and stores in a database a range of templates of the object in order to detect similarities with the input image given in the detection phase (see Section 5.4.1.2). Each template corresponds to a different camera point of view. The exact number of templates varies depending on the geometry of the training model (no rotations for sphere objects), but no more than 10 templates are stored at a time. The types of features that are extracted to generate a template are colour gradients and surface normals (see Figure 5.4) which are robust against changes of illumination and noisy inputs. The colour gradient of largest magnitude for each colour channel is extracted from the RGB image. These gradients are obtained only for the silhouette of the model to deal with texture-less objects. On the other hand, the depth map retains the surface normals, in particular, the orientation of the normals from the interior of the model.

Finally, the 2D bounding box for each template and its corresponding pose is stored in addition to the purely visual information (colour and depth). This pose is obtained through a marker tracking system and represents the transformation

between the camera and the model coordinate systems (see Figure 5.5). This means that in following operations (see Section 5.4.2.2) of the online phase, this reference pose must be refined applying a delta transformation in order to adjust the reference 2D bounding box to the bounding box detected in the input image.

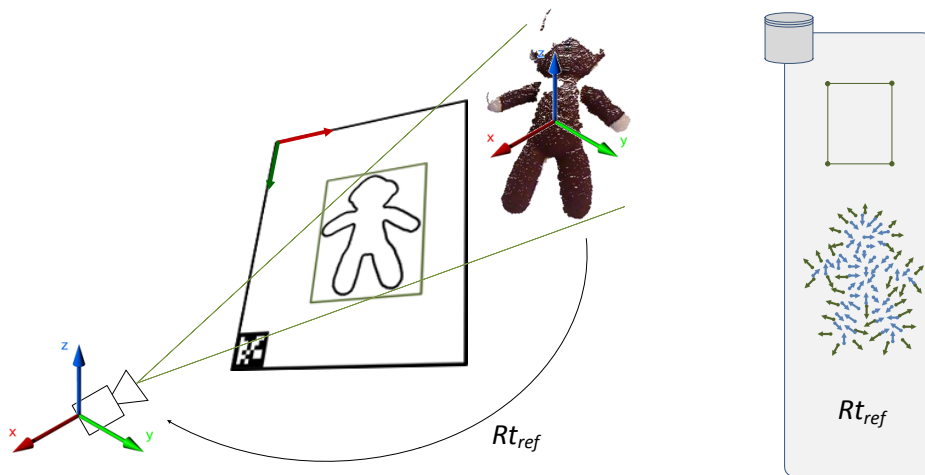


Figure 5.5: Each template in the database is composed of a 2D bounding box, the current processed point cloud, and the corresponding pose. This pose is obtained using a marker tracking system.

5.4.1.2 Online Execution

The online or detection step generates a number of hypotheses and selects the template that best fits the input data from the database (see Figure 5.6). In order to estimate the correct position of the object in the scene two tests are completed.

First, a search for correspondences between the input image and the reference image is performed. This search only takes the colour information into account, i.e., it computes the number of pixels (as far as colour is concerned) that the input cloud and the reference template have in common.

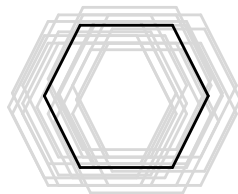


Figure 5.6: Tracking online execution. It generates hypotheses and selects the template that fits best.

The second test consists in computing the orientation and position of the object according to the previous correspondences. Since the reference and input clouds may not be completely aligned (the templates stored in the training step have been generated from a discretized point of view), the *Iterative Closest Point (ICP)* algorithm is used to obtain an accurate alignment between the two point clouds (see Figure 5.7). For all the hypotheses that have passed the colour check, the last pose is retrieved using the input depth map information (working with normals). It is worth noting that to carry out this process, the implementation that is available in the *Point Cloud Library (PCL)* (Aldoma et al., 2012) was used.

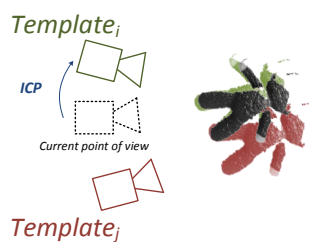


Figure 5.7: The *Iterative Closest Points (ICP)* algorithm is required to obtain accurate alignment since the templates are extracted from a discretized point of view.

The two tests mentioned are carried out in each frame. It should, however, be noted that once the object is detected after a search over the entire image (called *Tracking-by-Detection (TbD)* step), the following searches are bounded searches. This means that the search is performed in a neighbourhood close to that found in the previous frame (called *Frame-to-Frame tracking (FtF)*). The reason for this strategy lies in the fact that the computational cost of the tracking

process is reduced. Moreover, in the case that a satisfactory search is not found in that bounded search, the process resumes, starting from the initial state of the tracking.

5.4.2 Object Deformation

The main challenge of the non-rigid problem is to know what the transformation of the mesh will be after suffering a deformation. Therefore, this second phase is responsible for performing the deformable registration of the non-rigid model. As in the previous tracking step (see Section 5.4.1), here the stage is also divided into a preprocess task and online execution. The preprocess consists in defining a properly parametrization (see Section 5.4.2.1) of the physical model. The online execution in turn, calculates the correspondences between the model and the point cloud (see Section 5.4.2.2) acquired from the camera. These correspondences are then adjusted to a physical behaviour (see Section 5.4.2.3), and finally, the displacements are extrapolated to the triangle mesh (see Section 5.4.2.4).

5.4.2.1 Model Preprocessing

The model is preprocessed to generate the data structures required for its physical simulation. This procedure is divided into several steps: *Keypoint Generation*, *Voxelization* and *Mass-Spring Model Initialization*.

Keypoint Generation. Since features are not used to calculate deformations, a list of keypoints uniformly distributed on the surface of the model (see Figure 5.8(b)) is defined in order to relate the visual part with the 3D model. These keypoints operate as control points in order to calculate the deformation in the steps that follow. These keypoints are used to calculate the correspondences between the vertices of the triangle mesh and the raw point cloud (see Section 5.4.2.2). Thus, these keypoints act as a reference in order to extrapolate the movements and apply it to each of the vertices of the mesh.

To carry out this task, a ray casting technique is developed. It traces rays from the bounding box to the centroid of the model and calculates the intersection points of the rays with the triangles that make up the mesh. Moreover, this phase performs a filter in order to remove the keypoints

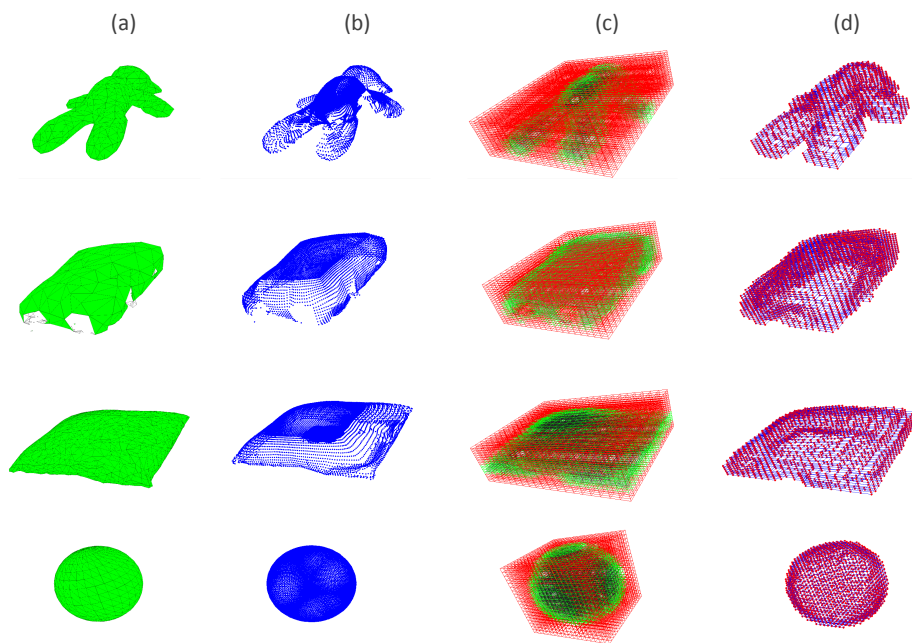


Figure 5.8: Preprocessing. From the input triangle mesh (a), a list of keypoints is extracted (b) and the shape is subdivided into voxels (c) to approximate the structure to the physical model (d).

that could affect the correct functioning of the deformation process. This means that the keypoints that are relatively close (based on a threshold) to the bounding box of the object are discarded to avoid noisy movements along the corners.

Voxelization. This step subdivides the volume into cells (also called *voxels*). A space partition is generated from the original geometry of the model. This data structure gives all the information required in order to adjust the object shape to the underlying MSM.

First, the global bounding box of the model is computed based on the triangle mesh. This bounding box is then divided uniformly into equally sized cubes (voxels) and the required memory is reserved. Each voxel is assigned to the triangles contained therein, and viceversa.

In order to identify which voxels are intersected by a given primitive object, it may be possible to compare each triangle of the scene with each cell of the voxelized mesh. However, this solution has a high

computational cost. Consequently, to speed up the calculation and identify which voxels are partially or totally included in a triangle, a test based on a triangle-AABB (*Axis Aligned Bounding Box*) is performed. This intersection test (Akenine-Möller, 2005) analyses only the voxels that are in collision with the AABB of the triangle to reduce the number of intersection tests.

Once an intersection between a triangle and a voxel is detected, a pointer to the triangle is stored within the data of the voxel. This way, each voxel stores a pointer list to all the triangles that intersect its volume. The different stages of the process are graphically represented in Figure 5.9.

Finally, the classification process is performed. In this phase, apart from the allocation of the voxels to triangles which intersect, a classification process is performed of each of the cells. Three different types of voxels are distinguished: *interior*, *exterior* and *boundary* (see Figure 5.8(c)). The *interior* voxels are classified as the voxels that are in the inside of the model and do not intersect with the triangles of the mesh. The *exterior* voxels are outside the model and they too do not intersect with the triangle mesh. The *boundary* voxels are those that intersect with the triangles of the mesh representation of the object.

The classification process starts classifying all the cells as internal and once the intersection between the voxels and triangles is computed, those cells are set as boundary cells. Finally, it is necessary to make a final run to identify the external cells.

Figure 5.10 shows an example of the steps that are performed to obtain the voxel classification.

Mass-Spring Model (MSM) Initialization. As previously mentioned, MSM offers advantages over other deformation simulation methods such as the FEM. MSM is a physical simulation model with a straightforward implementation and with a relatively small computational cost. Furthermore, it works easily with a voxel structure, which is a simple structure that can be easily obtained for an object of any shape (see Figure 5.8(d)).

Following the idea of classical mechanics, MSM is a particle system that is composed of a set of particles (each one characterized by its mass, position and speed) and a set of springs that link them together in pairs. These springs exert an elastic force on each particle when the mesh is deformed (see Equation 5.1).

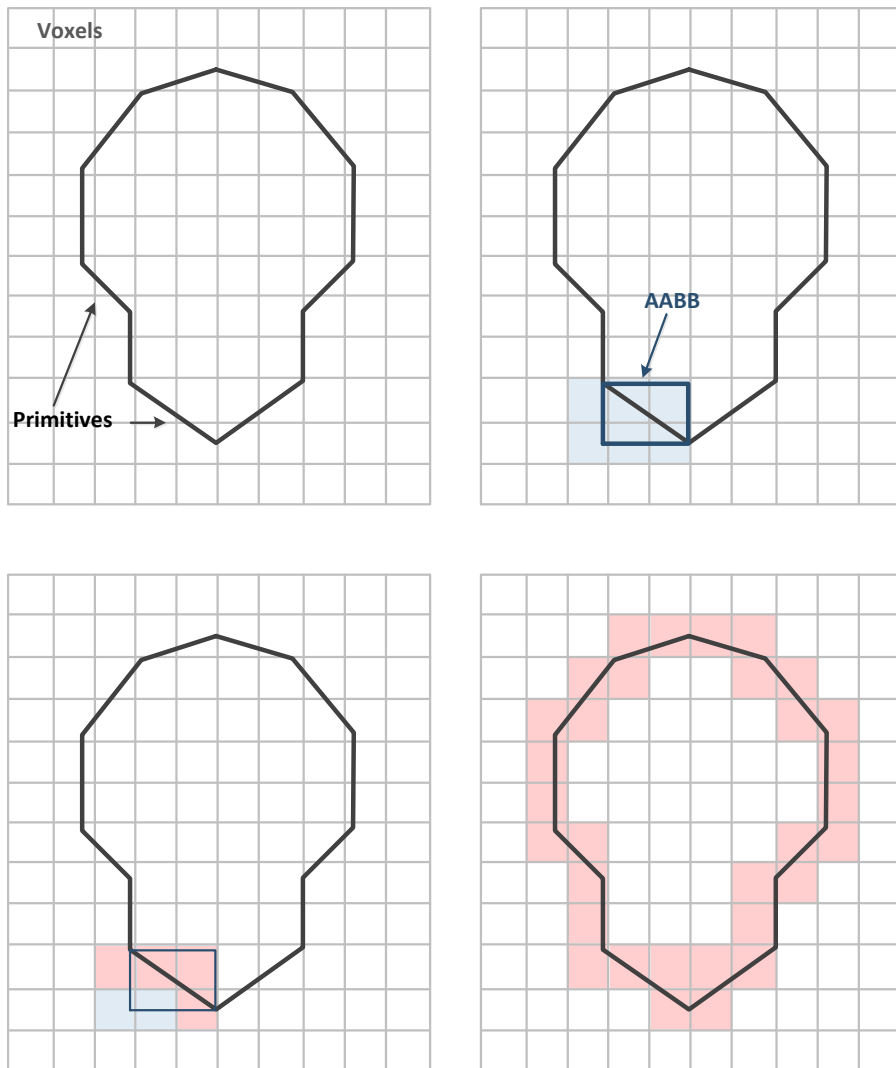


Figure 5.9: 2D representation of the voxelization process. Memory allocation (a), voxels that intersect with the bounding box of every primitive are selected (b), a triangle-AABB test is performed between the primitive and every selected voxels (c) and the coloured voxels contain at least one primitive (d).

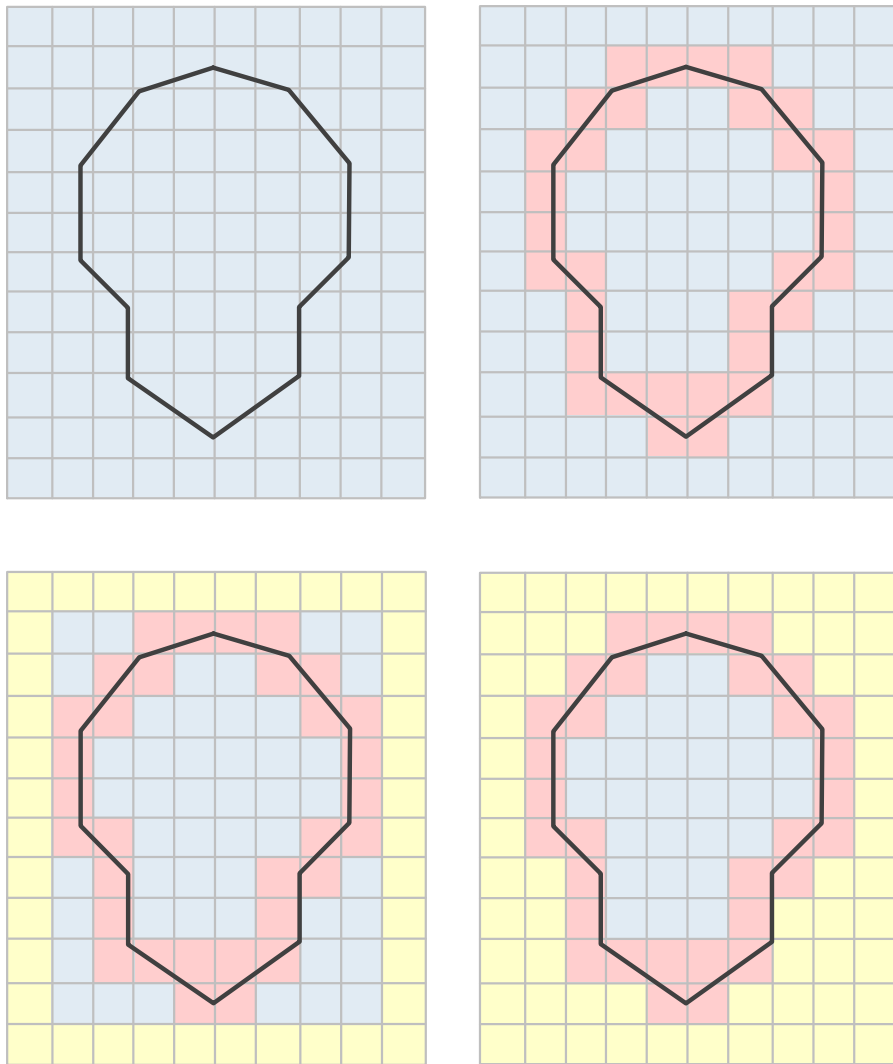


Figure 5.10: Voxel classification: *interior* voxels in blue, *external* in yellow and *boundary* ones in pink. Initially, all voxels are classified as *interior* (a); the voxels that contain a triangle are classified as *boundary* (b). The process starts finding the *exterior* voxels (c) and finally, the obtained classification (d).

$$\mathbf{f}_i = -\mathbf{f}_j = k_s \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|} (|\mathbf{x}_j - \mathbf{x}_i| - l_0), \quad (5.1)$$

where \mathbf{f}_i and \mathbf{f}_j are elastic forces on particle i and j , respectively. k_s , is the stiffness of the spring and determines the behaviour of the model, l_0 is the initial distance between the pairs of nodes, and \mathbf{x}_i and \mathbf{x}_j are the positions of the particles.

The forces are proportional to the elongation of the spring and the k values (the stiffness of the spring) determine the behaviour of the model, i.e., the higher the k values, the higher the stiffness of the simulated body, and vice-versa. In this context, the k values are computed based on (San-Vicente Otamendi et al., 2011). This work uses an eigenproblem approach to find the parameters of a cubical MSM. Taking the FEM model as a reference, it is able to define the parameters k_1 , k_2 and k_3 for the stiffness of the springs for each edge, face diagonal and internal diagonal of a cube, which reduces the difference between the modes to a minimum. The parameters of MSM can be rewritten as Equation 5.2 (in the experiments Poisson's ratio is 0.0 and Young Modulus is 29 *KPa* to simulate a soft sponge-like material).

$$k_i = \frac{L_0}{2} k'_i E, \quad (5.2)$$

where k_i is the stiff value and k'_i is the pseudo-stiffness values for edge springs, face diagonals and internal diagonals. L_0 is the initial length of the spring and E is Young's modulus.

5.4.2.2 Mesh Registration

Once the camera pose is obtained in the tracking step (see Section 5.4.1), mesh registration is used to provide an estimation of the correspondences between the offline keypoints (see Section 5.4.2.1) and the current point cloud captured by the RGB-D camera for each input frame. These matches serve as input to the MSM, which are responsible for calculating the new shape of the object. This stage belongs to the online execution and involves three steps: *Keypoint Selection*, *Correspondence Matching* and *Voxel Displacement*.

Keypoint Selection. Not every keypoint selected in the preprocessing model is visible all the time. In addition to the keypoints discarded in the

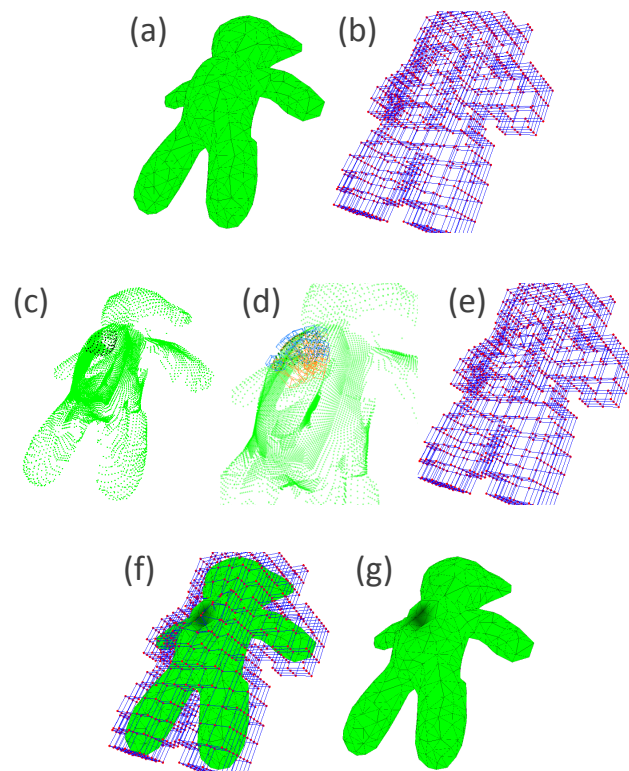


Figure 5.11: Object deformation. The process of registering the deformation of a triangle mesh (a). An input point cloud (c) is given to match the points with the corresponding keypoints. The matching step (d) is divided into two octree searches (blue *OBB test-1* and orange *OBB test-2*). Once the matches are known and extrapolated to the voxel structure (b), a physics-based model is introduced to estimate the deformation of the voxels (e) and an isoparametric formulation is used to deduce the interpolation between the voxels and the mesh triangles (f) to know the final structure of the model (g).

preprocessing step, the system must take into account that all keypoints are not visible during the online execution due to the point of view of the camera. Therefore, all the keypoints that are not visible must be discarded, whether deformation is being applied at that point or not. Although the visible keypoints can be extracted from the selected tracking template (precalculated in the preprocessing step), it may be the case that other tracking methods do not return this information. Therefore,

this framework is composed of two totally independent and self-sufficient stages: detection/tracking and deformation. For this reason, in this step, a series of tests are run in order to filter and select suitable keypoints. More specifically, there are two main types of tests: *occlusion query* and *points per triangle*.

- **Occlusion query.** The points that are not in front of the camera are set as not visible and consequently discarded. The procedure computes the normal vector of the keypoint (the normal of the triangle it belongs to) and calculating the difference (in degrees) with respect to the vector that represents the point of view of the camera (see Figure 5.12). If the difference is below a threshold value, the point belongs to a back triangle and it is discarded. Otherwise, it lies on a front triangle and it is retained for further processing.

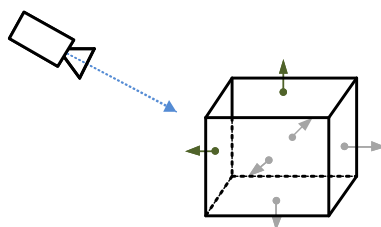


Figure 5.12: Occlusion query. The difference (in degrees) respect to the vector that represents the point of view of the camera.

- **Points per triangle.** This test discards the keypoints whose triangles are not completely visible. A triangle is defined as not completely visible, if at least one of the points that lie on it is not visible. This rule is applied because the keypoints that lie on the triangles that are not completely visible are likely to be a source of errors.

Correspondence Matching. Once the 2D position of the model's bounding box is defined by the tracking method, its corresponding pose is obtained from the trained template. However, it is necessary to apply a delta transformation to calculate the final pose (see Figure 5.13). First, a homography transformation (H) is calculated between the reference bounding box and the detected bounding box in the image. In this way the resulting 3D delta transformation ($\Delta R t$) is deduced from the stored reference 3D pose in order to match the projection of the reference bounding box to the currently detected bounding box.

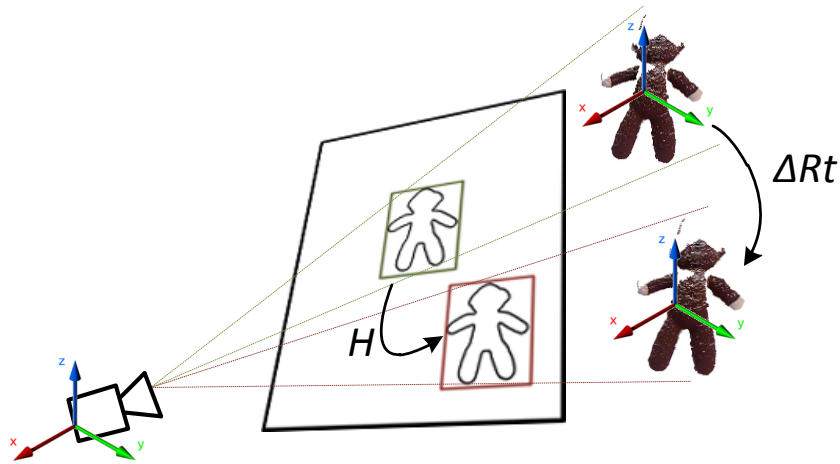


Figure 5.13: The reference pose is obtained by calculating a delta transformation between the reference bounding box and the detected one.

Nonetheless, although an initial 3D pose is estimated, it is necessary to make a registration alignment between the input cloud and the reference model because it may be possible that they are in different scales (in terms of CAD modelling), and thereby compute a more refined pose. Thus, by projecting the 3D model keypoints with the former calculated pose, the corresponding 3D coordinates of the input frame are obtained, establishing 3D-3D correspondences between the two point sets. Instead of using algorithms such as ICP that are computationally expensive for this registration refinement, the algorithm of (Umeyama, 1991) is used to estimate the transformation between the two point clouds. This algorithm returns the transformation between two point sets according to:

$$e^2(R, t, c) = \frac{1}{n} \sum_{i=1}^n \|y_i - (cRx_i + t)\|^2, \quad (5.3)$$

where given two sets of points x_i and y_i , the mean squared error is minimized to obtain the transformation parameters R , t and c , which correspond to rotation, translation and scale values, respectively.

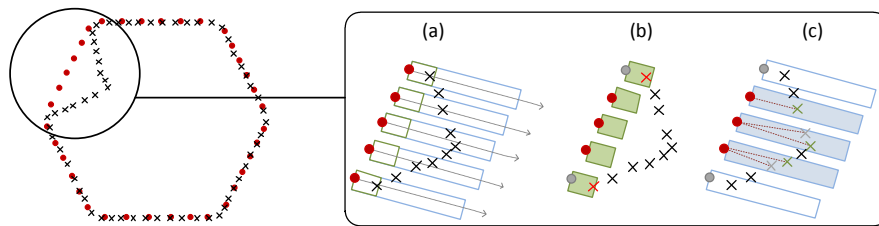


Figure 5.14: Correspondence matching. Once the search areas are defined (a), two tests are applied in order to relate the keypoints to the point cloud: *OBB test-1* (b) and *OBB test-2* (c).

After the transformation is obtained, the matching (finding correspondences between the point cloud and the keypoints) process begins. The main goal here is to find the associations between the visible keypoints with the points in the point cloud. Therefore, an intelligent search structured as a scale search is performed to achieve a low computational cost (see Figure 5.14 (a)). This search is divided into two main levels. The first level is used to discard the keypoints that do not have any deformation, and the second serves to make the association between the keypoint and a sample of the input point cloud.

In order to carry out this search, two search areas are defined for all keypoints. These areas are represented by two different *OBBs* (*Oriented Bounding Box*), each of which is related to two different tests (see Figure 5.11 (d)). These *OBB* areas differ from the size of the height value. The values for these boxes are fixed according to the measures of the global bounding box and calculated offline. The orientation, in turn, is defined as negative to the normal of each keypoint in order to find the deepest point.

The first test (defined as *OBB test-1*) corresponds to a search at the smallest *OBB* (see Figure 5.14 (b)). The main goal of this test is to verify if there is any point inside a small box around its position. If there is one, the keypoint will be discarded since it is understood that there is no deformation for it or that the deformation is so small that it can be disregarded for the global deformation. This also offers robustness against possible noise in the data of the cloud as captured from the camera.

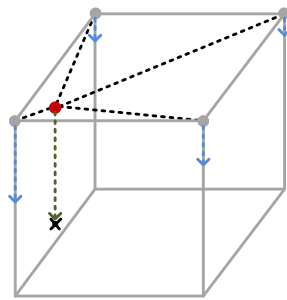


Figure 5.15: The displacements of the vertices of the voxel are applied in a weighted manner.

Once the non-deformed keypoints are discarded with the first test, the following search (defined as *OBB test-2*) finds a correspondence over the second box see Figure 5.14 (c)), more concretely it provides the most distant point.

Both searches are carried out through an efficient octree search based on a recursive search of the boxes (implementation provided by the PCL library). This search allows an octree representation to be obtained for a given input cloud point and, consequently, performs fast intersection tests.

Voxel Vertex Displacement. Finally, the displacement of the vertices of the voxels must be calculated in order to extrapolate the nodes of the physical structure (see Figure 5.11 (e)). This process detects the voxel to which the keypoint belongs and applies, in a weighted manner, the displacement to the vertices of the nearest face of the voxel (see Figure 5.15). This weight is represented by an interpolation between the keypoint and the vertices of the voxel that contains it. Following this, all the displacements are obtained, i.e., the total movements of each node of the mesh.

5.4.2.3 Mesh Physical Simulation

After computing the displacements of the nodes of the voxel structure from the previous step (see Section 5.4.2.2), a MSM model is fed with the distances to simulate the deformation. The MSM model is dynamically simulated and the motion equations of the model are integrated in time using an explicit Verlet integrator. The system computes the elastic forces exerted by each spring on the nodes, i.e., the physical deformation of the voxels. This process runs in parallel for

each spring, accumulating the resulting forces for each node. Once the forces of the springs are computed, the displacements of the nodes are directly computed from the forces acting on each node:

$$\mathbf{x}_i(t + \Delta t) = 2\mathbf{x}_i(t) - \mathbf{x}_i(t - \Delta t) + \Delta t^2 \frac{\mathbf{f}_{springs}}{M_i}, \quad (5.4)$$

where \mathbf{x}_i is the position of node i , M_i is the mass of the associated particle and $\mathbf{f}_{springs}$ is the total force exerted by the springs connected to the particle. Δt is the integration time step.

The Verlet integrator is only conditionally stable, and the value Δt is limited by the stiffness of the system. Stiffer materials require a lower time step for the simulation to be stable. In this case the deformation is usually applied to objects of low rigidity such as foams, which can be easily deformed by hand. The low stiffness allows a time step value that is high enough to guarantee the real-time execution of the simulation. In any case, since the objective of the framework is to provide a visually accurate representation of the body, the simulated stiffness can always be chosen with a value low enough to achieve real-time performance. In this case, if the material of the tracked object is stiffer than the material of the simulated object, which is unlikely in the context of the proposal, the system tracks deformations that are smaller than those that would be expected as a result of the simulation for the same force. However, since the simulation is guided by the points captured by cameras, the difference in behaviour is only extended to internal points and points far from the deformed surface.

Moreover, this simulation is performed with a number of iterations per each frame until the deformation converges to a stable configuration.

5.4.2.4 Model Fitting

The position of the nodes of the voxels obtained in the previous step (see Section 5.4.2.3) allows determining the new position of all the vertices of the triangle mesh (see Figure 5.11 (f)). The vertices of the triangle mesh are interpolated with regards to the 8 nodes of the containing voxel. For that purpose, in the preprocessing step the isoparametric coordinates of each vertex are computed. These coordinates relate the mesh represented by the voxels and the triangle mesh. The isoparametric coordinates represent a mapping from

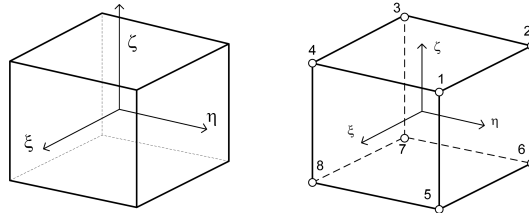


Figure 5.16: Isoparametric coordinates. 8-knot linear element that is defined like a 2-sided cube.

local coordinates to a general coordinate system. Therefore, given a point in the isoparametric coordinates, the corresponding global coordinates of the point are obtained using the mapping equation. Knowing this, the interpolation function is defined by three coordinates ξ , η and ζ with a range between -1 to $+1$, i.e., setting this local system like a 2-side cube (see Figure 5.16), using trilinear interpolation. More concretely, an 8-knot linear element is used where each of its faces is a rectangular element of 4 knots (see Equation 5.5). See (Wilson, 1998) for more details about the formulation.

$$N_i = \frac{1}{8} (1 + \xi \xi_i) (1 + \eta \eta_i) (1 + \zeta \zeta_i), \quad (5.5)$$

where N_i is the interpolation function for each of the knots in the cube and ξ_i , η_i and ζ_i are the coordinates in knot i .

5.4.3 Experiments

The following section presents the results of the proposal. A set of experiments that evaluate performance (primarily the error estimation and execution times), robustness against noisy data and the adaptability of the method are shown for some geometrically shaped objects. The hardware setup consists of an Intel Core 2-Quad Q9550 at 2.83 GHz and 3 GB of RAM equipped with a Kinect XBOX 360. Also, 4 different models were used to evaluate the performance of the framework: *Teddy Bear*, *Sponge Bob*, *Ball* and *Cushion*. The representation of *Teddy Bear* can be defined as a highly complex deforming shape with a homogeneous texture (texture-less). *Sponge Bob* has a simple geometrical shape since the limbs of the object are discarded. It is a squared model with a rich texture (that is not used). *Cushion* is along with the *Sponge Bob* model the simplest geometric shape. It is a parallelepiped object with texture. Both the

Table 5.1: Execution times (in *ms*) for the tracking module.

STEP	TIME (<i>ms</i>)			
	Teddy Bear	Sponge Teddy	Ball	Cushion
<i>Detection</i>	270.01	271.47	262.99	275.82
<i>Frame-to-frame</i>	90.47	61.77	51.73	114.91

Sponge Bob and the *Cushion* models, consist of box-like shapes (though *Cushion* is simpler than *Sponge Bob*) with different kind of textures (irregular texture for *Sponge Bob* case and regular texture for *Cushion*). And finally, the *Ball* example provides another geometric shape that is like a sphere to make the registration. It is a curvilinear and texture-less object.

5.4.3.1 Performance

The first set of experiments presented in this section shows the performance of the system in terms of computational cost and accuracy level.

Computation Time

Table 5.1 presents the execution times of tracking module, while Figure 5.17 shows the performance of the physics registration modules for the four models detailing the computational cost of each step. As can be seen in the results, the deformation framework is able to obtain the non-rigid transformation of the shape between 8.3 and 12.65 *ms* depending on the model.

The most important bottleneck is the process of matching keypoints with the point cloud acquired from the RGB-D camera. However, it remains quite low due to the optimal octree search that is a scalable way to reduce the computational cost. Thus, the second OBB search is notably faster than the first.

In terms of the physics simulation module, Figure 5.18 displays the times of the mesh fitness step, according to the number of iterations of integration that the system requires in order to achieve good physical behaviour. In that sense, it is observed that as the number of iterations increases, the computational cost also increases. Nevertheless, the experiment depicts that the visual adjustment of the deformable mesh is not influenced in an excessive way by the number of iterations (see Table 5.2). These errors were estimated

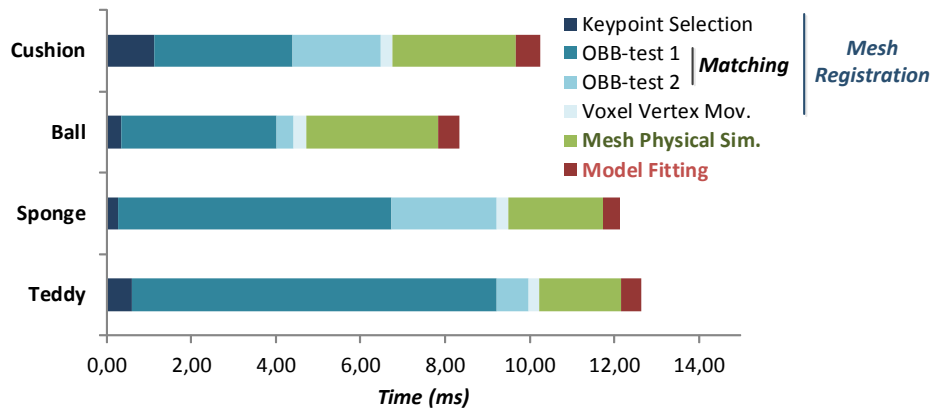


Figure 5.17: Execution times (in *ms*) for the different steps making up the physical registration stage.

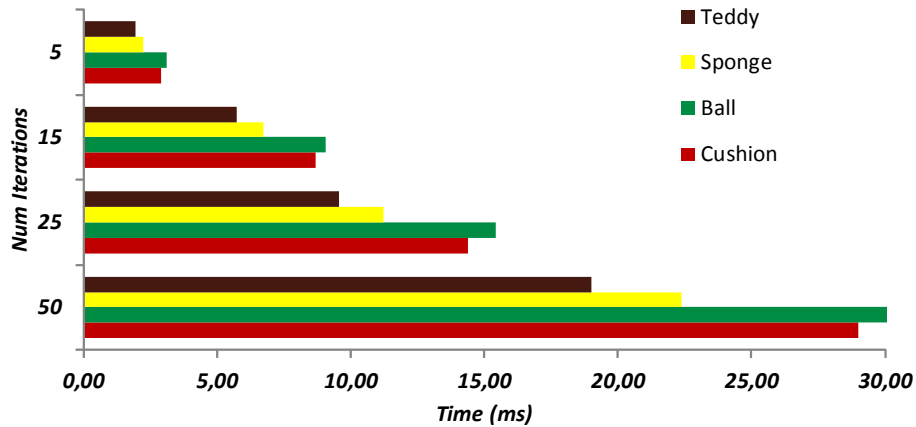


Figure 5.18: Execution times (in *ms*) of the mesh physical simulation step according to the number of iterations in the physical integration step.

using synthetic data (deformed meshes). For more detailed explanation please refer to next paragraph (see Section 5.4.3.1-*Accuracy level*). Consequently, it was not considered necessary to use a large number of iterations to perform the task in order to achieve acceptable visual results as observed in Figure 5.22.

Table 5.2: The mean error and the standard deviation (in *mm*) for one point of view of the camera of the mesh physical simulation step according to the number of iterations in the physical integration step of.

	Number of iterations				
	5	15	25	50	
μ - error (<i>mm</i>)	9.31	6.75	6.6	6.15	<i>Teddy Bear</i>
σ - error (<i>mm</i>)	2.35	2.1	1.95	1.5	
μ - error (<i>mm</i>)	6.3	5.85	5.55	5.3	<i>Bob Sponge</i>
σ - error (<i>mm</i>)	1.35	1.5	1.8	1.75	
μ - error (<i>mm</i>)	6.61	6.45	6.05	5.9	<i>Ball</i>
σ - error (<i>mm</i>)	2.35	2.45	2.02	2.25	
μ - error (<i>mm</i>)	6.19	6.03	5.55	5.4	<i>Cushion</i>
σ - error (<i>mm</i>)	2.78	2.25	2.2	2.1	

Finally, regarding computational cost, it is also worth mentioning that the described tracking algorithm runs in an average time of ~ 270 *ms* for the detection step and the FtF step varies from 60 to 114 *ms* depending on the density of the point cloud for each of the four models. However, in order to achieve a complete solution, this proposal is well suited to the framework because it uses a template-based methodology instead of using features (not required in the physical simulation step).

Accuracy Level

An error estimation was also performed in order to check the response of the method. For this purpose, taking into account the results described above, a configuration of 300 keypoints was chosen (see Section 5.4.2.2), while the physical simulation step ran in 5 iterations. In addition, a set of synthetic data was used to evaluate the accuracy of the registration for the different models. A set of 25 random deformations of the mesh for each of the 4 different models was built, and the accuracy of each deformation was calculated from 9 different points of view (see Figure 5.19). Each point of view sets the visible keypoints, and consequently, the result of the deformation can be different. Thus, Table 5.3 depicts (in *mm*) the mean errors, the standard deviation and maximum error from

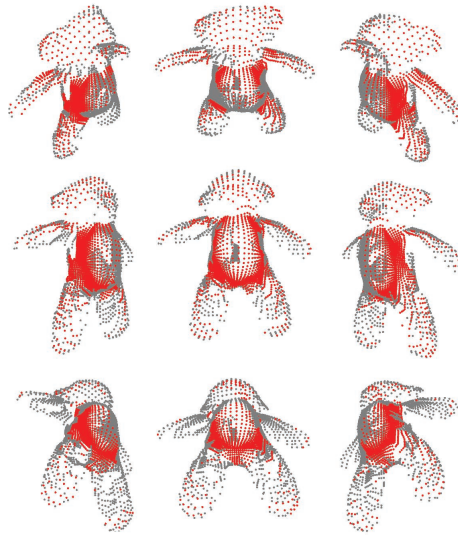


Figure 5.19: Different points of view of the model in order to estimate the visible (red) and not visible keypoints (grey).

the ground truth vertices mesh and the resulting mesh. These results demonstrate the accuracy of the system for different types of objects and camera points of view. The reason there are error differences between the models lies in the fact that the geometric shapes are different. Even so, the errors remain low enough for every case where the visual quality of the reconstructions is good.

Table 5.3: The mean error, standard deviation and the maximum error (in *mm*) for four types of models.

	Mean error	Stdv error	Max Error
<i>Teddy Bear</i>	9.31	2.35	15.1
<i>Sponge Bob</i>	6.3	1.35	7.8
<i>Ball</i>	6.61	2.35	8.9
<i>Cushion</i>	6.19	2.78	8.9

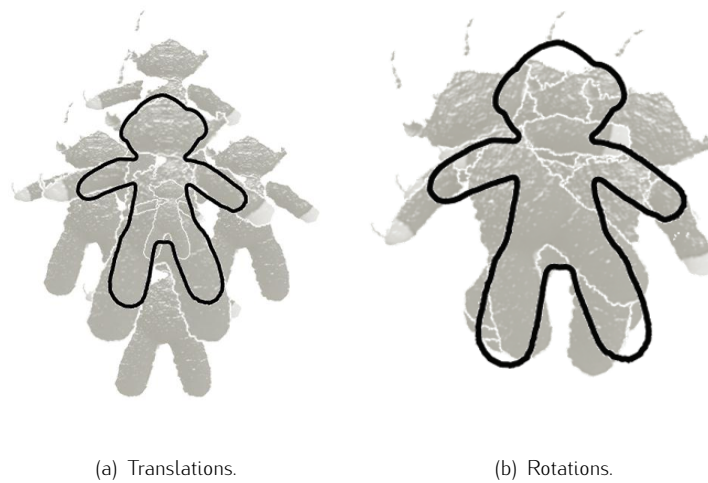
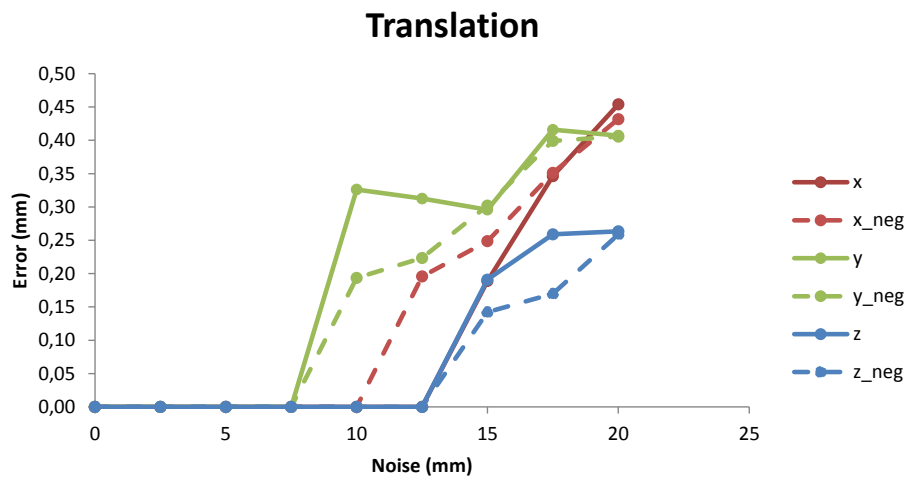


Figure 5.20: Transformations applied to the *Teddy Bear* model.

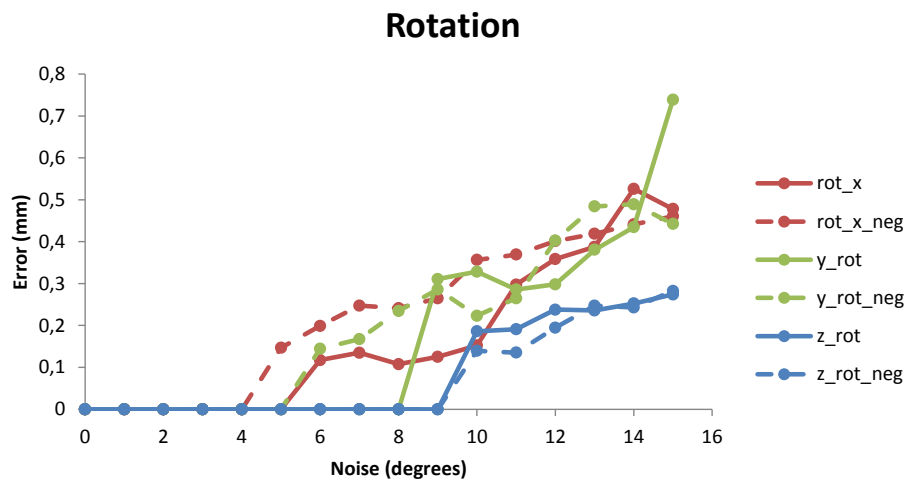
5.4.3.2 Robustness

This second experiment shows the limits of the method in terms of robustness when applying some noise, i.e., while distorting the movement of the points. The methodology consisted in applying a set of transformations to the input cloud in order to disorder it and then testing the behaviour of the method. The transformations applied to the *Teddy Bear* model were divided into two main types: *translations* (see Figure 5.20 (a)) and *rotations* (see Figure 5.20 (b)), doing each of the three axes x , y and z . Figure 5.21 indicates the results for each of the transformations.

For translations, movements from 0 to 30 mm were executed with a step of 2.5 mm along the three axes. Resulting errors varied from 0.2 to 0.5 mm in case of a 20 mm displacement. In the rotation transformation, in turn, the strategy used consisted in rotating the point cloud from 0 to 15 degrees. In this case, resulting errors varied from 0.2 to 0.8 mm in the case of 15 degrees. Analysing the results, it can be concluded that the method maintains the error at reasonable levels despite the presence of noise in the data and that it correctly deduces the deformation being applied.



(a) Noise translation.



(b) Noise rotation.

Figure 5.21: Error (in *mm*) when the input data cloud is perturbed (for the *Teddy Bear* model) by adding two different types of noises: *translations* and *rotations* in each of the three axes.

5.4.3.3 Adaptability

Figure 5.22 shows the correct visual output obtained by this solution and stresses the adaptability to many different types of geometrical shapes. In this case, the four different models were used to test the adaptability of the framework: (*Teddy Bear*, *Sponge Bob*, *Ball* and *Cushion*). The *Teddy Bear* example represents the reconstruction after a video sequence of 490 frames. *Sponge Bob* is a 325 frame video sequence. In the same way, *Cushion*, is a 565 frame video sequence reconstruction. Finally, the *Ball* example, is a 410 frame sequence. It should be noted that in all of these examples the results are sufficiently valid to obtain a correct visualization and develop some AR applications.

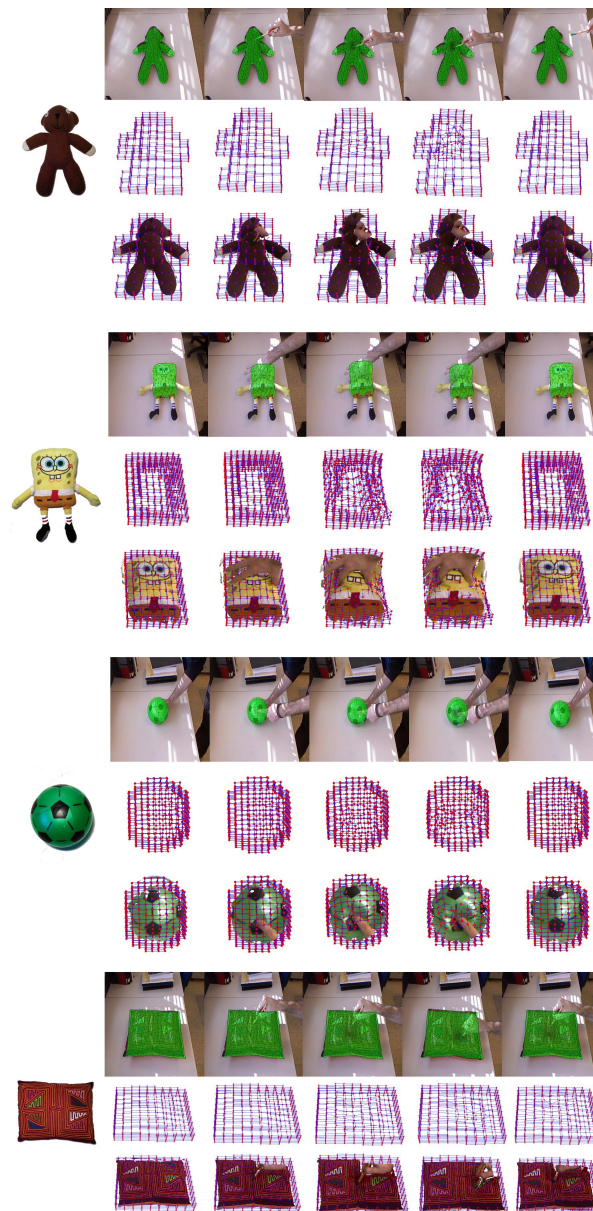


Figure 5.22: Visual results of 3D recovery shapes. Four experiments were performed: *Teddy Bear* (a), *Sponge Bob* (b), *Ball* (c) and *Cushion* (d). For each of the four examples, the following information is presented: the model, the original image with the projection of the recovered 3D mesh (first row), the recovered 3D mesh in terms of the physics model (second row) and the same recovered mesh with the raw data acquired from the depth camera (third row).

5.5 Proposed Method using a Finite Element Method

The method described in this section follows a similar technique to the former (see Section 5.4). The main difference between the solutions lies in the mathematical model that simulates the physical behaviour of the bodies. In this case, the solution takes advantage of a formulation based on FEM. The objective is to demonstrate that the level of accuracy achieved with this formulation is higher than with MSM, although the computational cost increases.

The modularity of the pipeline proposed in the previous section (see Section 5.4) allows the incorporation or replacement of certain functionalities. Therefore, the following section only describes the new modules or the altered ones. More precisely, the main stages associated to the object deformation are explained. Furthermore, since this section is only focused on analysing the physical simulation process, the detection and tracking steps are carried out through a marker tracking system.

On the other hand, it is worth mentioning that the phases involved in the physical model simulation, require specific type of models (different geometry mesh and material properties of the bodies). Unlike the MSM proposal, this solution requires a tetrahedral mesh. To that end, the model represented as a triangle mesh is converted to a tetrahedral mesh. Moreover, in order to deduce an optimal parametrization of the material, this process requires models composed by homogeneous material. The objects such as *Teddy Bear*, *Sponge Bob* and *Cushion* are therefore discarded, since they are composed of different materials (they have outer sheath material which differs from the interior one) and the test samples that can be extracted from the model are not uniform (cotton material). The *Ball* is also discarded since it does not contain any type of interior material (it is filled with air). The models that have been used to perform the experimentation process are those shown in Figure 5.23, whose physical parametrization is detailed in Section 5.5.2.1.

5.5.1 Tracking

The tracking and detection steps are based on a marker tracking system instead of using the system presented in Section 5.4.1. More concretely, a multiple marker configuration is used, with the aim of minimising the tracking failure when an occlusion occurs in any of the markers.

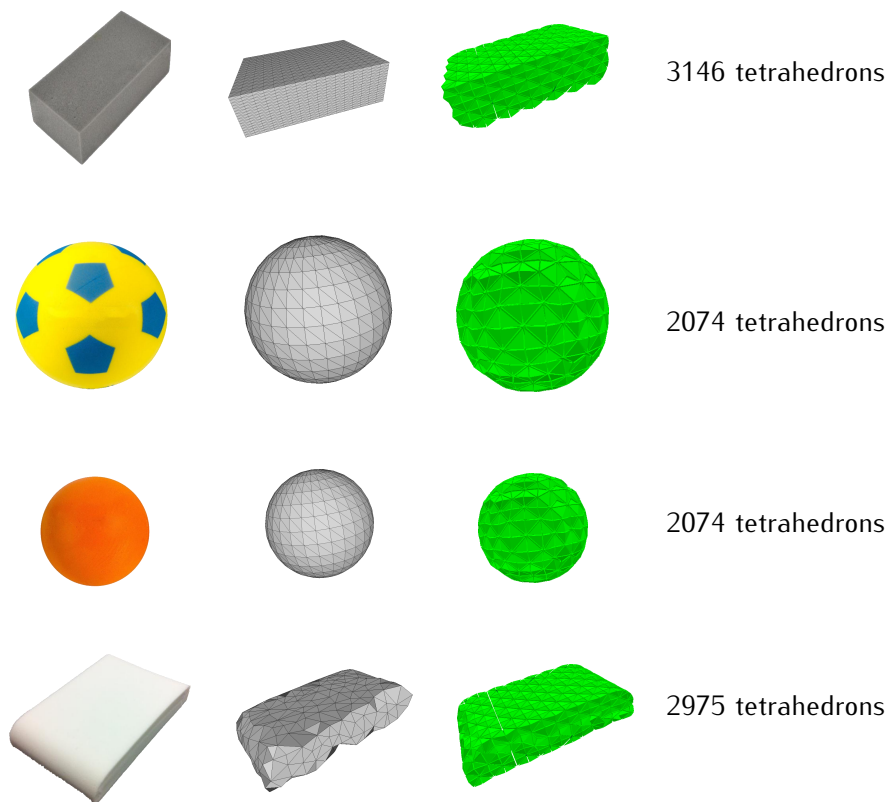


Figure 5.23: Transformation from triangle mesh to tetrahedral mesh. The model (left) represented as a triangle mesh (middle) is converted to a tetrahedral mesh (right) for the FEM formulation.

In a preprocessing stage, a calibration process is performed in order to obtain the relation transformation between the markers and the corresponding 3D object. ARToolKitPlus marker tracking system has been used to perform the detection of the markers.

5.5.2 Object Deformation

As in the previous solution, here the object deformation is divided into two main steps: *Model Preprocessing* and *Mesh Physical Registration*. The *Model Preprocessing* (see Section 5.5.2.1) is focused on defining the proper

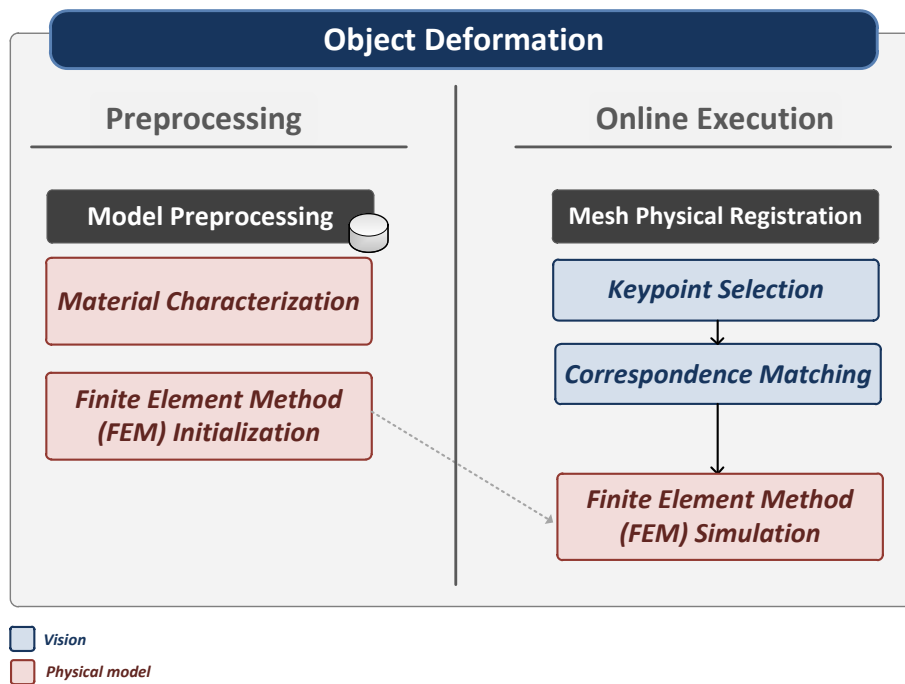


Figure 5.24: Overview of the proposed object deformation method. Offline and online phases are subdivided into a visual and physical simulation processes.

parametrization of the physical model. Moreover, simple shear tests in a rotational rheometer were performed in order to define the mechanical properties of the material. The online execution or *Mesh Physical Registration* (see Section 5.5.2.2) feeds the input of the physical simulation module with the information acquired from the RGB-D camera. However, unlike the previous solution, at this stage there is not *Model Fitting* phase (see Section 5.4.2.4) since the deformation of the mesh is directly computed by the *FEM Simulation* (see Section 5.5.2.2). Furthermore, both preprocessing and the online execution are divided into two main phases that are related to the visual part and the physical simulation model, respectively. Figure 5.24 illustrates the entire process.

5.5.2.1 Model Preprocessing

The offline phase, which is essential in order to carry out the *Mesh Registration* (see Section 5.5.2.2) in the online execution, is executed only once for each model. In order to complete this preprocessing task, it is necessary to generate information from the visual part of the system as well as the physical parametrization of the model. The first is responsible for generating a set of control points to determine the correspondences between the raw information acquired from the camera and the 3D mesh of the model (*Keypoint Generation*). In the case of the visual part, the keypoints correspond directly to the vertices of the tetrahedral mesh and hence it is not necessary to perform a *Keypoint Generation* step (see Section 5.4.2.1). The physical model procedure in turn, is divided into two main parts which correspond to the description of the physical formulation based on FEM as well as the range tests performed to define the material properties (*Material Characterization*).

Material Characterization. In order to obtain the mechanical properties of the deformable objects, a sample of each material was tested in a parallel-plate rheometer (Anton Paar Physica, MRC 301). The rotational rheometer characterizes the material under shear loads, that are common loads in soft tissues at surgical procedures (Cheng et al., 2008; Misra et al., 2010; Horgan and Murphy, 2010).

Figure 5.25 shows the schematic representation of the experimental set-up. The top plate of the rheometer was lowered until it contacted the upper surface of the sample.

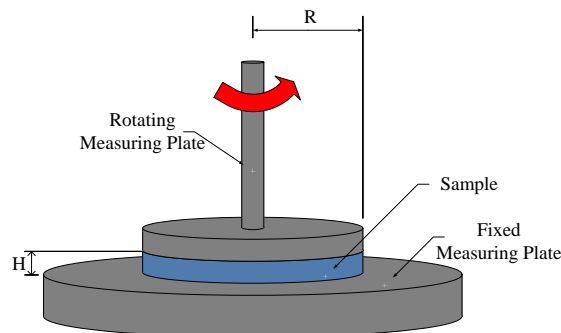


Figure 5.25: Parallel plate rheometer.

Measured strain (γ) is not constant along the sample, since it is a function of the radius of the plate (R), the gap (H) and the deflection angle (φ), as follows:

$$\gamma[1] = \frac{\varphi[\text{rad}]R[\text{mm}]}{H[\text{mm}]} \quad (5.6)$$

Therefore, maximum deformation and maximum shear rate occur at the edge of the plate, and reported data is related to this position.

Amplitude sweeps have been performed in a cylindrical samples (diameter = 25 mm and thickness = 2 - 4 mm). Amplitude sweep is an oscillatory test performed at variable strain amplitudes, keeping frequency at a constant value. As long as strain amplitudes are still below the limit of linear viscoelastic range, the values of the dynamic properties remain steady and the material shows reversible-viscoelastic behaviour. It deforms elastically and returns to its original shape when the applied stress is removed. However, at amplitudes higher than this limit, some fraction of the deformation will be permanent and non-reversible, or the structure of the sample can even be completely destroyed.

Mechanical properties of the tissues are defined within linear viscoelastic range. A strain ramp was imposed from 0,001% to 100%, at 1 Hz. The value of shear modulus, G , within linear viscoelastic range is obtained. Poisson's ratio, ν , has been selected for each material. Linear elasticity, isotropy and homogeneity of the material has been assumed. Therefore, Young modulus, E , is defined by

$$E = 2G(1 + \nu) \quad (5.7)$$

Density, ρ , has been considered constant within the tissue sample and it has been determined by measuring its mass and volume. Table 5.4 shows the values of the mechanical properties established for each material.

Table 5.4: Mechanical properties of the tested materials

	Yellow Ball	Orange Ball	Square Sponge	Pillow
ρ [kg/m ³]	57.4	75.2	14.3	100
E [Pa]	50000	37800	76261	25000
ν	0.3	0.3	0.3	0.3

Finite Element Method (FEM) Initialization. The system uses a non-linear Total Lagrangian explicit FEM using a tetrahedral mesh. This kind of formulation is well suited to surgery simulators since it provides a very compact and efficient implementation. The formulation can handle easily non-linear material and large deformations. Additionally the method does not require the computation of a stiffness matrix, which allows an easy adaptation to topological changes.

In this case, for simplicity and computational efficiency, a Saint-Venant-Kirchhoff is implemented using the properties for a linear material model described above.

This formulation allows a direct computation of the elastic forces acting on each node when the model is deformed. The force is divided into two terms, one that only depends on the initial geometry of the mesh (as is constant during the simulation) and a second that depends on the current configuration through the first-Piola-Kirchhoff stress tensor. See (Irving et al., 2004a) for additional details of the method. During the initialization the first term is computed and stored.

5.5.2.2 Mesh Registration

As with the solution explained above (see Section 5.4.2.2), this phase estimates the correspondences between the offline keypoints and the current point cloud captured by the RGB-D camera. In addition, these matches serve as the input displacements to the physical module, which is responsible for calculating the new shape deformation. This stage is divided into a visualization step (*Keypoint Selection* and *Correspondence Matching*), and the mesh physical simulation module (*Finite Element Method (FEM) Simulation*).

Notice that the *Keypoint Selection* and *Correspondence Matching* phases (see Figure 5.26) follow the same steps as those presented in Section 5.4.2.2. For more information, please refer to the corresponding technical details in each part of that section.

Finite Element Method (FEM) Simulation. Once the nodes of the simulation mesh have been matched to the deformed surface, they are used to control a dynamic simulation of the deformation of the solid. During this simulation, the nodes are displaced from their original position to the distance detected using the vision system. The simulation runs through a

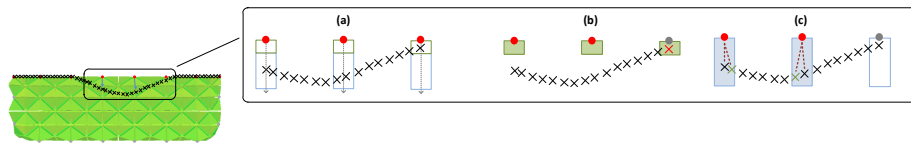


Figure 5.26: Correspondence matching. The keypoints in the FEM formulation correspond to the vertices of the tetrahedral mesh. The process for the correspondence matching consists of a two different type of tests: *OBB test-1* (b) and *OBB test-2* (c).

virtual time until it achieves its final configuration that it is considered, that is, until it represents the deformed state of the solid. The simulation is performed with a time step that has been computed previously and guarantees the stability of the semi-implicit Euler integrator that is employed.

5.5.3 Experiments

This section presents the results of the proposed method using the FEM formulation. These experiments consist in evaluating the performance (error estimation and computation time) and the adaptability of the system for different kind of models. To develop this set of experiments, a mechanical robot was used, the commercial Mitsubishi PA-10-7C robotic arm.

The hardware setup was the same as the previous solution: Intel Core 2-Quad Q9550 at 2.83 GHz and 4 GB of RAM equipped with a Kinect XBOX 360. Moreover, as explained above, the models used to test the performance are different from the previous: *Yellow Sponge Ball*, *Orange Sponge Ball*, *Rectangular Sponge* and *Pillow*. The first two models can be defined as sphere shapes with different sizes and texture types. The *Yellow Sponge Ball* is textured, while the *Orange Sponge Ball* contains a homogeneous texture (textureless). Likewise, *Rectangular Sponge* and *Pillow* are textureless simple geometric shapes. All the materials of the objects differ from each other as Section 5.5.2 depicts. In other words, all the sponge materials are different while the *Pillow* is composed of a viscoelastic material.

5.5.3.1 Performance

The next section provides the evaluation of the performance in terms of the accuracy level and the computational cost for the object deformation step using the FEM formulation.

Accuracy Level

The error estimation was performed in order to validate the proposed system and it has different stages.

Primarily, a force was applied with the robotic arm and through the FEM formulation the deformation was computed. Simultaneously, the body was scanned by the 3D Sense™ Scanner during the deformation. The reconstruction of this scanner served as reference for the error estimation, that is, it was the ground truth. According to the technical specifications of the 3D Scanner reported by Cubify®, the accuracy at 0.5m depth resolution is 1mm, and the spatial x/y resolution at 0.5m is 0.9mm.

Subsequently, a filter of subdivision of the surface was applied to each model (FEM and Scanner models) in order to obtain two dense point clouds. Then, the interest region of each point cloud was defined (the area where the deformation was being applied was selected manually).

Table 5.5: The mean error, standard deviation and the maximum error (in mm) for four types of models.

	Mean error	Stdv error	Max Error
<i>Yellow Sponge ball</i>	5.01	2.38	12.63
<i>Orange Sponge ball</i>	1.75	1.11	6.09
<i>Rectangular Sponge</i>	1.49	0.99	7.19
<i>Pillow</i>	0.24	0.17	0.87

Once the two regions of the two point clouds were selected, the open source *CloudCompare* software¹ was used to compute the error estimation. This is an application for managing and comparing 3D point clouds. More concretely, two main functions were used to complete the process: *Register* and *Distance* procedures. The first, *Register*, aligns two points clouds, and the

¹Open source implementation of this method can be found in <http://www.danielgm.net/cc/>.

second, *Distance*, calculates the distances between two point clouds. The error was computed by the Euclidean distance from one point of the first cloud to the nearest point of the second cloud. The mean errors (in *mm*) are displayed in Table 5.5. As can be seen, the error varies from 0.24 to 5.01*mm*, with standard deviation from 0.17 to 2.38*mm*.

These error values are low enough to achieve a correct visual feedback. Figure 5.27 illustrates the visual results for each of the deformations applied to the four models alongside the ground truth.

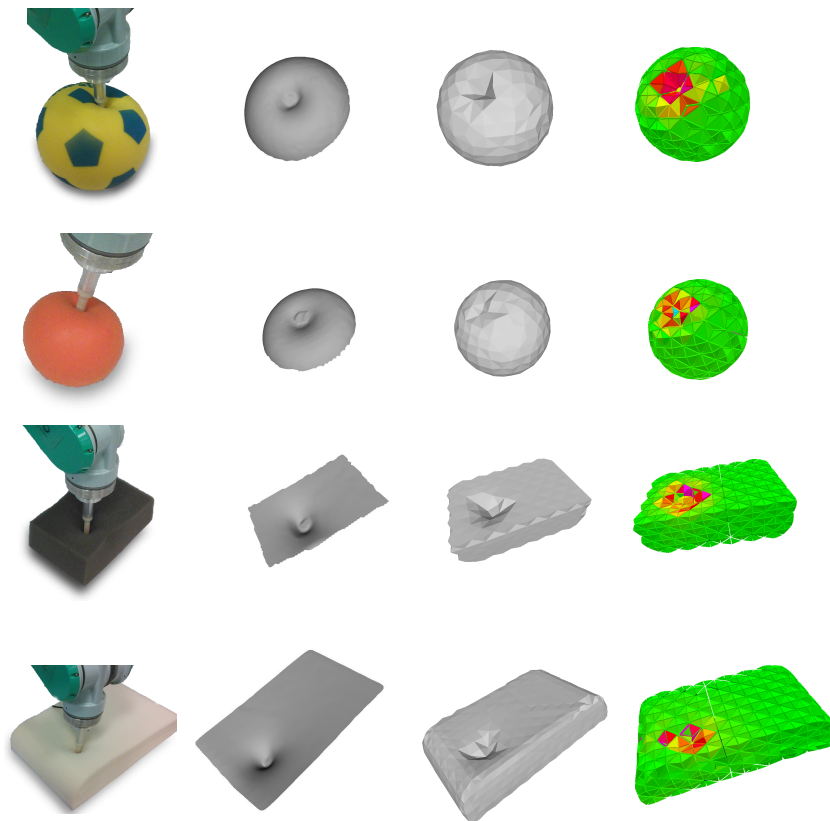


Figure 5.27: Visual accuracy level when a force is exerted on the model (first column). Scanner reconstruction (second column), FEM deformed mesh (third column) and visual feedback of the strains of the mesh (fourth column).

Computation Time

This section shows the execution times of the object deformation module for the four models. More concretely, Figure 5.28 presents the times of the mesh physical simulation step, that is, the FEM physical simulation process. As can be seen, the execution times vary from 37.82 to 62.16 *ms* depending on the model.

Though it has been proven that an acceptable level of accuracy is achieved, the performance is still not as high as required.

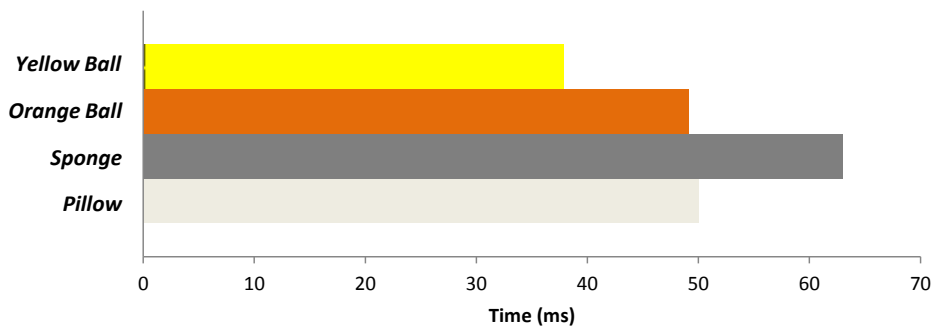


Figure 5.28: Execution times (in *ms*) of the FEM physical formulation for the four models.

5.5.3.2 Adaptability

Figure 5.29 shows the visual results obtained for different kind of objects with different material characterization. The *Yellow Sponge Ball* example is a video sequence of 498 frames, which represents the reconstruction of curvilinear deforming shape with a texture that is not used. The *Orange Sponge Ball* example is also a curvilinear model without texture after a video sequence of 461 frames. The *Rectangular Sponge* is a 236 frames video sequence which reconstructs a parallelepiped textureless object. Finally, the *Pillow* example is a 383 frames video of a viscoelastic material object without texture. These results conclude that the visualizations of the deformations are sufficiently correct in order to develop AR applications.

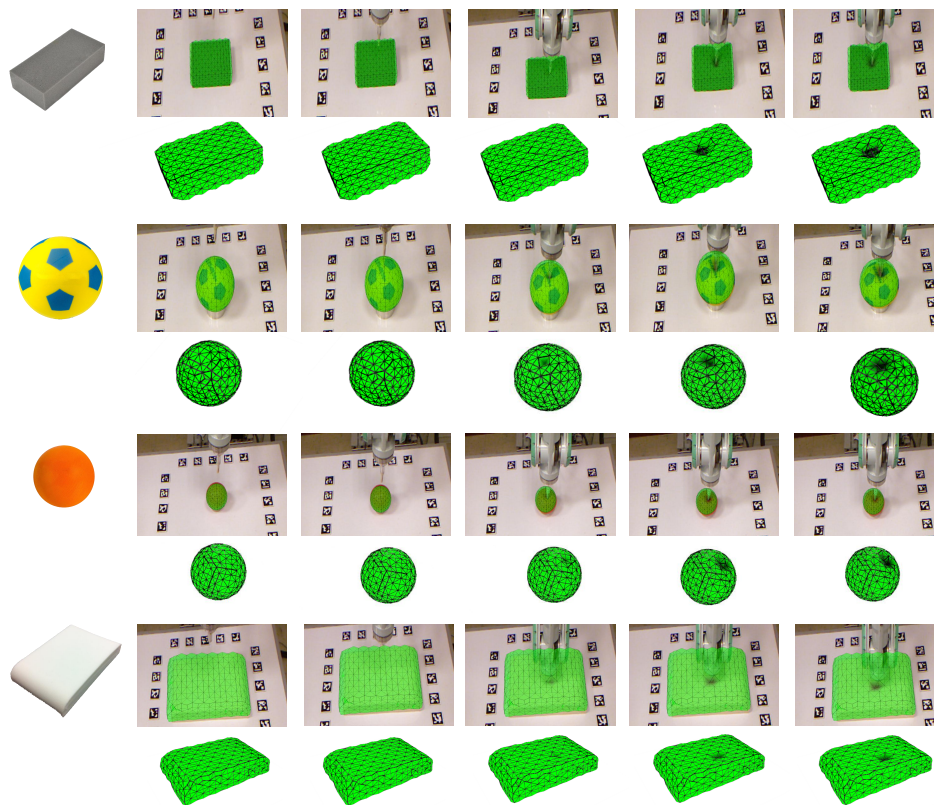






Figure 5.29: 3D recovery shapes for different objects: *Rectangle Sponge*, *Yellow Sponge Ball*, *Orange Sponge Ball* and *Pillow*. For each of the four examples, the following information is presented: the model (left), the original image with the projection of the recovered 3D mesh (first row), the recovered 3D mesh in terms of the FEM physical model (second row).

5.6 MSM versus FEM

The following section demonstrates the differences between the two physical methods. In practice, an experimental set was designed to show the results with regard to the computational cost and error estimation. In line with the experimental strategy applied for the FEM formulation, the same hardware system and configuration of models (valid for both methods) was used.

5.6.1 Accuracy Level

Table 5.6: The mean error, standard deviation and the maximum error (in *mm*) for four types of models between FEM and MSM simulations.

	FEM			MSM		
	μ	σ	max	μ	σ	max
	5.01	2.38	12.63	5.97	3.62	15.11
	1.75	1.11	6.09	4.34	2.14	9.33
	1.49	0.99	7.19	2.01	1.4	11.12
	0.24	0.17	0.87	0.45	0.34	1.91

The physical formulation based on FEM achieves more accurate level results than MSM. Table 5.6 carries out a comparison between the two methods for the four different models. In this case, the process to compute the error is the same as that described in Section 5.5.3.1. Note that FEM values have been copied again for completeness of the table and facilitate the comparison.

On the other hand, Figure 5.30 shows a colour map (extracted from the *CloudCompare* software) that represents the differences between the scanner and both physical formulation clouds (FEM and MSM).

On the basis of the above, it can be concluded that the postulated premise is realized. The FEM formulation returns more accurate results than MSM. Even so, it must be emphasized that both solutions results are good enough for deceiving the human eye, i.e., they offer a correct visual feedback. This means that, when a force is exerted on the object, the physical behaviour is realistic enough.

5.6.2 Computation Time

The computational cost is another factor that was also tested to compare the performance of both types of configurations. Table 5.7 validates that the computation time of the physical deformation is higher for the FEM formulation. In order to perform this evaluation, the same set of models was used: *Yellow Sponge Ball*, *Orange Sponge Ball*, *Pillow* and *Rectangular Sponge*. There are significant differences between both solutions. FEM formulation execution varies from 37.82 to 62.95 *ms*, while MSM varies from 3.04 to 3.85 *ms*.

Table 5.7: Comparison between FEM and MSM formulations in terms of computation time (in *ms*).

	FEM	MSM
<i>Yellow Sponge ball</i>	37.82	3.27
<i>Orange Sponge ball</i>	49.1	3.04
<i>Pillow</i>	49.9	3.85
<i>Rectangular Sponge</i>	62.95	3.81

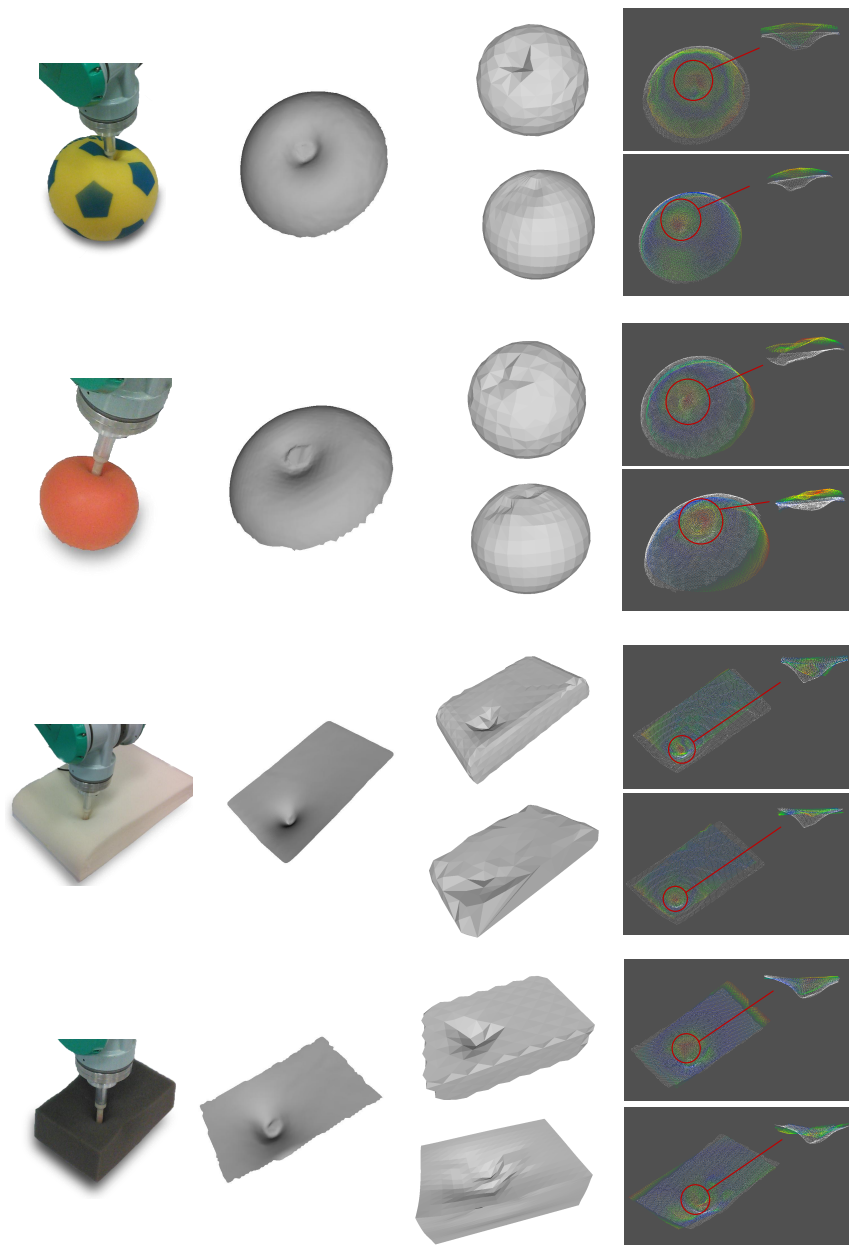


Figure 5.30: Error comparison between FEM and MSM formulations for the four different models. For each of the four models, the scanner mesh, FEM (up) and MSM (down) meshes and a colour map that represents the error between each of the solutions with respect to the scanner result (extracted from *CloudCompare*) are presented.

5.7 Surgical Simulation System

Robotic systems integration in medicine and more specifically in the surgical field entails a remarkable improvement in the delivery of healthcare services. Nowadays, robotic devices and computer-assisted technologies are included as a means of guidance, diagnosis, verification and general assistance during the performance of a surgical intervention (Landeira, 2014).

However, the use of robotic devices in surgery creates new challenges that must be overcome for their successful deployment. A robot acting on a tissue exerts a force and produces deformations over which the surgeon has no direct feedback. The tracking of the deformation of the organs is required for assistive technologies such as AR, for example to support the surgeon by a visual feedback of the deformations that are taking place on the tissues, or to support the surgeon by locating the position of a malignancy to be removed from an organ. In most cases, the inclusion of sensors to track the position of the organs of the patient is not usually feasible. Thus, the use of alternative approaches such as CV techniques has become a real need to solve this kind of problem.

Nevertheless, the very presence of the robot produces occlusions that limit how much information can be obtained using only CV. The combination of CV with a suitable physical simulation can be used to reduce the missing information. In this case, the accuracy required for the surgical simulations remains important. In contrast to deformable models used in video games and animations, the purpose of soft tissue models in medical simulation is to model realistically the behaviour of biological tissues. Consequently, the simulation of the deformation of the tissue should be controlled using real material parameters. Thus, these parameters should be obtained from biomechanics experiments instead of intuitively adjusted parameters. However, in robotic guidance and surgery assistance systems not only accuracy is essential, but also computational performance must be taken into account, in order to provide a fast enough visual feedback to the surgeon. Therefore, in this application it is critical to obtain a compromise between accuracy and computational cost. Being able to provide surgical realism at interactive rates of simulation is one of the most challenging problems in robot surgical assistance systems (Cotin et al., 2000).

The results obtained in the proposed FEM solution (see Section 5.5) make it possible to develop a novel visual tracking method based on physical simulation. This procedure is capable of obtaining the deformation produced by a robotic system on patient tissues, and therefore, it obtains the basic information to

provide assistance and guidance to surgeons using visual or haptic feedback. More concretely, this method has been integrated within the framework of robotic surgery system through a surgical module developed in (Landeira et al., 2014).

The process is divided into two main phases. The first deals with handling robotic surgical arm movements, while the second computes the deformations applied from the first step through a vision system.

5.7.1 Previous Works

Regarding the modelling of the tissues material behaviour to be simulated, there is no agreement in the literature on how to define the most suitable material models required for surgical simulation. Model selection is often a very subjective process; different modellers choose different models to describe the same phenomenon (Miller et al., 2010). A key question in modelling tissue behaviour for simulation is the complexity level of the model required for a certain application. More complex models can simulate better the behaviour of the tissue at the cost of computation cost. For surgical simulation of biological tissues, models must be simple enough to solve a broad range of problems, but complete enough to realistically describe the behaviour under a variety of load conditions (Kerdok, 2006).

Linear FEM models are commonly used for the modelling of deformable materials, mainly because the equations remain simple and the computation can be optimized. Several authors use linear elasticity to model brain deformations (Ferrant et al., 2001; Warfield et al., 2001; Skrinjar et al., 2002; Clatz et al., 2007) or to simulate muscle and soft tissue (Gladilin et al., 2001). However, simulations built upon linear elastic models can only be applied to small deformations, while the most surgical procedures involve organs being subjected to large ones.

Some recent works in the literature present robotic systems complemented by vision techniques in surgery. Authors like (Haouchine et al., 2013a; Haouchine et al., 2013b) show an image-guided biomechanical model that captures the complex deformations undergone by the liver during surgery. They work with stereo cameras and use a FEM in order to capture the behaviour of the object and calculate the deformation. The FEM is based on a co-rotational formulation which allows for large displacements while relying on a linear expression of the stress-strain relationship, which is taken from the literature. However, these kind of approaches are very oriented to medical surgery application, which usually involve specific formulation, the position of the camera should

be static, and in some cases, they are not executed in real time (Haouchine et al., 2012). Furthermore, (Haouchine et al., 2014a) proposes a real-time method to register the non-linear elastic model deformations using the image points acquired from a monocular camera. Nevertheless, this solution is based on an orthographic projection, whose computation is easier compared to the perspective projection. (Hamarneh et al., 2014) presented pre-operative surgical planning, intra-operative image registration and AR visualization for image-guided tumour identification. They focused on kidney cancer cases with robot-assisted partial nephrectomy performed with a da Vinci surgical robot (Intuitive Surgical, Inc.). They built the biomechanical model of kidney tissue and tumour with FEM using a co-rotational tetrahedral formulation. They also determined the mechanical properties of the tissue from the literature.

5.7.2 Justification

This section presents the basis for the development of a robot-assisted surgical system (Landeira et al., 2014) combined with a vision module through which it is possible to obtain the deformations of a non-rigid object when a force is applied to it. A visual feedback system for the surgeons could be developed based on the information obtained from this system.

In environments such as medicine or industry, it is sometimes difficult to perform a tracking of visual cues (often essential for a vision system), either by the lack of textured areas and models or the environmental conditions (as blood in medicine). In this sense, compared to existing solutions (Haouchine et al., 2014b; Haouchine et al., 2014a), the presented vision module avoids the use of formulations that are not usually robust for texture-less surfaces or objects. This makes the solution robust against sudden illumination changes and adaptable for tracking any type of object (in terms of its geometrical shape). The visual feedback of the deformation state of the object relies on a non-linear total Lagrangian Finite Element formulation, that provides a fast and accurate enough biomechanical model of the object, while being able to handle complex material models. Furthermore, compared to other works (Haouchine et al., 2013a; Hamarneh et al., 2014), the mechanical properties of the models are obtained experimentally, in order to simulate a more realistic behaviour of the materials.

5.7.3 Proposed System

This section presents a complete framework for registering deformations of non-rigid objects when a surgical robot is applying a force and therefore, offering the possibility of providing an extra visual feedback to the surgeon.

The selected robot-assistant module has been developed as a prototype for a cooperative robotic platform aimed at assisting in surgery for lumbar transpedicular fixation in (Landeira et al., 2014).

The vision module is composed by a RGB-D camera whose raw information is used to feed the input of the physical model. Two different models have been used to perform the experimentation process: a *Porcine Kidney* and a *Calf Brain*. These models are shown in Figure 5.31 and their physical parametrization is shown in Table 5.8.

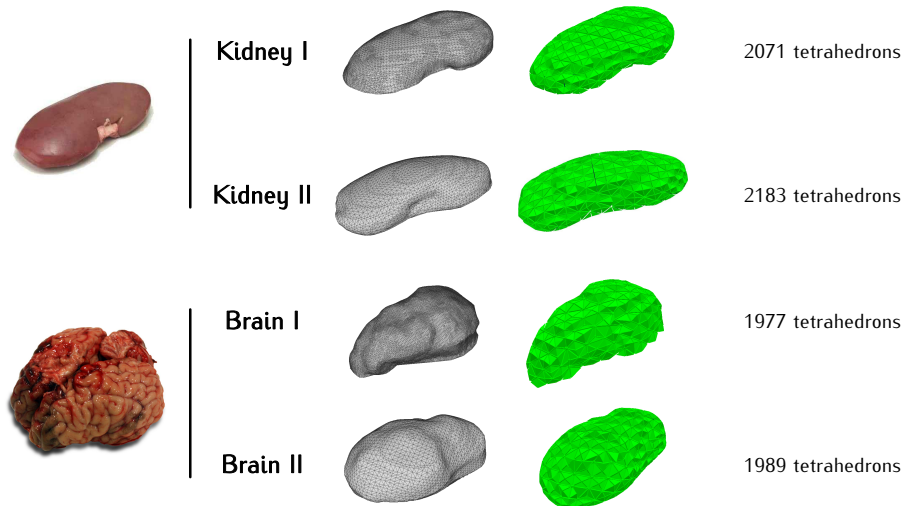


Figure 5.31: Two different types of models are used for the surgical robot system process: *Porcine Kidney* and *Calf Brain*. The model (left) represented as a triangle mesh (middle) is converted to a tetrahedral mesh (right) for the FEM formulation.

In order to obtain a compromise between computation time and accuracy of the reconstruction, and based on the conclusions of (Wittek et al., 2009), deformable objects are modelled as Saint-Venant-Kirchhoff material within a non-linear FEM formulation. Materials are defined by the *Young modulus* E ,

Poisson's ratio ν and *density* ρ . In order to define the mechanical properties of the materials with a realistic physical behaviour, simple shear tests have been performed in a rotational rheometer (see Section 5.5.2.1).

Figure 5.32 illustrates the configuration of the system, whose description is divided into two main sections: *Robotic Platform* and *Object Deformation*. Furthermore, since the analysis is focused only on the deduction of the deformations of the organ, the process of computation of the camera pose (*camera tracking problem*) is carried out through a marker-based system. This provides a level of accuracy high enough to ignore the error. Following is a description of the *Robotic Platform*.

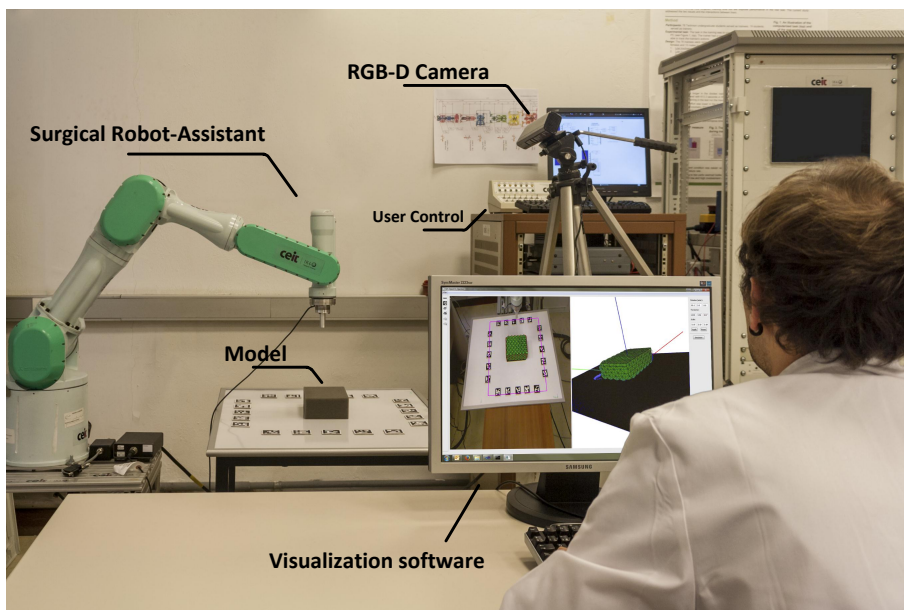


Figure 5.32: Configuration of the system. Robot-assisted surgical system combined with a vision module.

5.7.3.1 Robotic Platform

The surgical assistant used is composed by a commercial PA10-7C robotic arm (Mitsubishi Heavy Industries Ltd., Kobe, Japan) with open control architecture. This implies that, using a generic programming language, it is possible to

develop, in an independent way, the control algorithms to be implemented in the robot control PC. It is a 7DoF open chain serial manipulator, with all revolute joints that have a well-defined rotation axis. Its maximum load capacity is 98 N and it can achieve a distance of 1.03 m when fully extended. The robot has a force/torque sensor, Mini40 (ATI Industrial Automation, NC, USA). In that way, the force performed by the robot in the soft tissue is recorded. Figure 5.33 shows the robotic arm used as indenter (left) and its exploded view (right). The end of the robotic arm consists of a force/torque sensor, a grip, a big indenter and other small one.

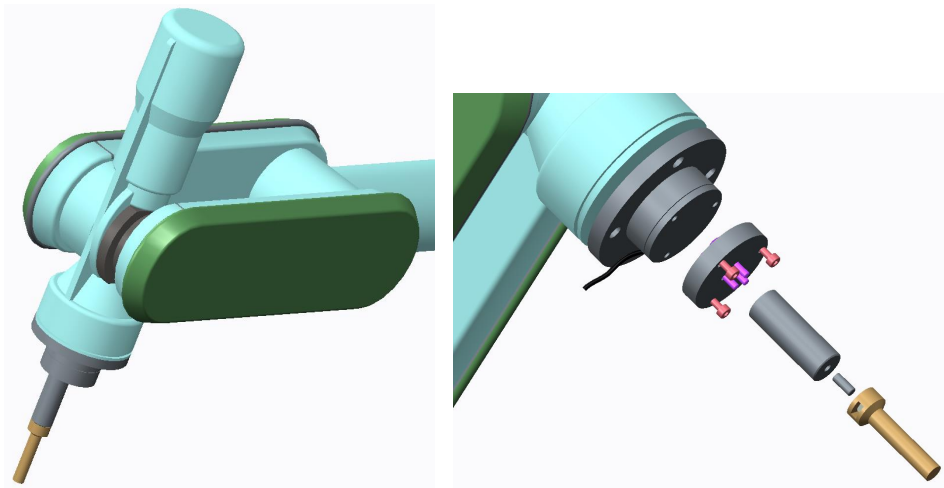


Figure 5.33: Representation of the robotic arm (left) and exploded view of its end (right).

5.7.4 Experiments

This section presents the results of the proposed method. These experiments evaluate the performance (error estimation and computation time) and the adaptability of the system for different kind of models. To develop this set of experiments, the robotic arm described above was used. Moreover, two cylindrical indenters were used to deform the tissue (see Figure 5.33). These indenters differ from the size: one has a diameter of 15 mm and 45 mm length, while the smaller one has a diameter of 8 mm and a 36 mm length.

The hardware setup consists of an Intel Core 2-Quad Q9550 at 2.83GHz and 4GB of RAM equipped with a Kinect XBOX 360. Porcine *Kidney* and

Table 5.8: Mechanical properties of the tested materials for the surgical system.

	Kidney	Brain
ρ [kg/m ³]	1000	1000
E [Pa]	1500	1085
ν	0.45	0.48

calf *Brain* models were used to evaluate the performance of the framework. *Brain* and *Kidney* provide alternative geometric shapes with different kind of textures. The texture of the surface was not used for recognition in any case. Furthermore, for each of these two categories of organs two different type of simulation models were used (overall two different models for each organ). The aim was to check that the behaviour of the system was the same using the same material parametrization and making incisions in different areas. For each simulation model a force was applied in different areas: *Brain I*, *Brain II*, *Kidney I* and *Kidney II*. For the *Kidney II* example in turn, the smaller indenter was used in order to test the deformation with different tools. Figure 5.31 depicts the previous categorization as well as the number of tetrahedrons of each model since it is a determining factor in the experimental results.

5.7.4.1 Accuracy Level

Two different techniques were used to validate the level of accuracy of the online FEM formulation. These techniques consist in comparing the results obtained with the online FEM simulation with a simulation of the same experiment using Abaqus and with the 3D reconstruction obtained through a 3D scanner. Both Abaqus simulation and 3D scanner outputs have been considered to be suitable ground truths because the physical deformations achieved are really accurate.

5.7.4.2 Abaqus

The experiments were simulated in Abaqus 6.13. The mechanical properties of both tools were considered rigid enough compared with the indented tissues ($\rho = 7850 \text{ kg/m}^3$, $E = 3000000 \text{ Pa}$, $\nu = 0.2$). Deformable objects were modelled with the same mesh as the one used in the online FEM simulation. Materials were defined by the properties presented in Table 5.8. The curve of the displacement

Table 5.9: The mean error, standard deviation and the maximum error (in *mm*) between FEM simulation and Abaqus simulation.

	Mean error	Std dev	Max error
Kidney I	1.28	2.23	14.97
Kidney II	0.40	0.68	6.42
Brain I	1.09	1.10	7.24
Brain II	1.08	1.37	10.36

versus time recorded by the robot in each experiment was imposed to the tool in the Abaqus simulation. Simulations were defined in a *Dynamic, Implicit* step with *Quasi-Newton* solution technique.

Deformed meshes from Abaqus and from the online FEM simulation were compared. The error between both meshes was obtained calculating the point-to-point Euclidean distance. Table 5.9 shows the values of this error for each material. The visual results of this comparison are displayed in Figure 5.34.

5.7.4.3 Scanner

Table 5.10 displays the mean errors (in *mm*) using the methodology of Section 5.5.3.1 to estimate the error. The mean errors (in *mm*) are displayed in Table 5.10 and Figure 5.35 shows the visual results of this experiment.

In conclusion, the error values are low enough to achieve a correct visual feedback. In this experiments the visual feedback consists in showing a gradient colour to emphasize the degree of deformation. Figure 5.36 illustrates the visual results for each of the deformations applied to the four simulation models alongside the ground truth.

Table 5.10: The mean error, standard deviation and the maximum error (in *mm*) between FEM simulation and scanner mesh.

	Mean error	Std dev	Max error
Kidney I	1.46	0.78	4.87
Kidney II	1.33	0.86	4.00
Brain I	2.26	2.00	11.68
Brain II	2.30	1.21	4.44

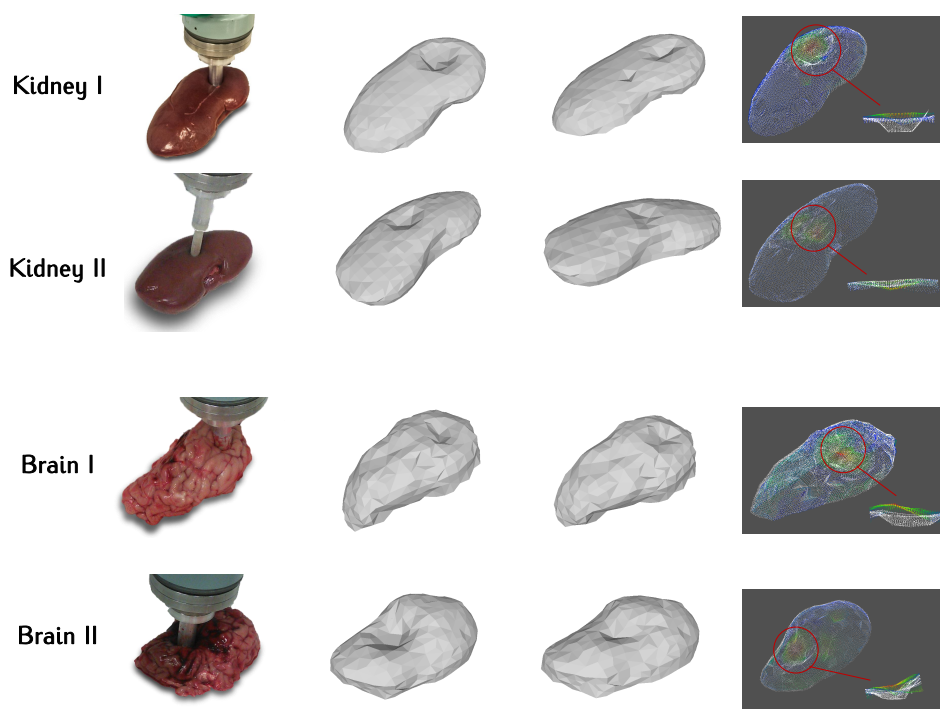


Figure 5.34: Comparison between FEM formulation and Abaqus simulation for different models. For each model: reconstructed mesh with Abaqus (second column), FEM returned mesh (third column) and a colour map (fourth column) that represents the error between the solutions (extracted from *CloudCompare*).

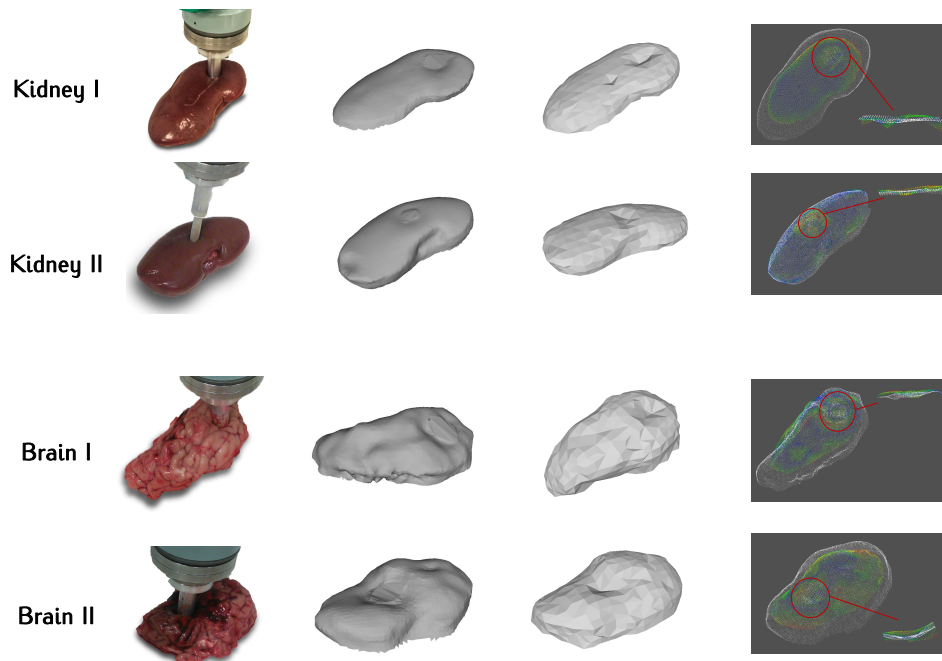


Figure 5.35: Comparison between FEM formulation and scanner reconstruction for different models. For each model: the mesh of the scanner (second column), the mesh of FEM (third column) and a colour map (fourth column) that represents the error between both meshes (extracted from *CloudCompare*).

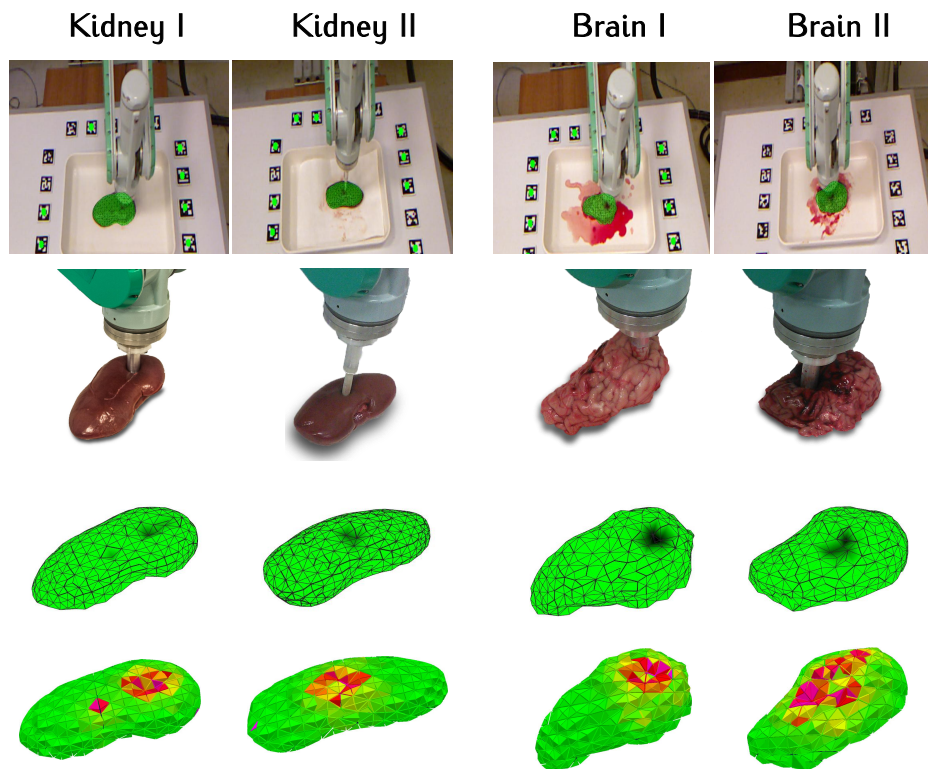


Figure 5.36: Visual feedback to surgeon when the soft tissue is deformed in a robot-assisted procedure. Two type of examples: *Kidneys* and *Brains*.

5.7.4.4 Computation Time

This section shows the execution times of the object deformation module for all models. In this case an exhaustive study of the computation times was not been taken into account for the marker tracking system as well as the matching step between the keypoints and the input point cloud. These results vary between 4 and 10 *ms* depending on the model. Therefore, this computation time is not a bottleneck for the performance of the method. The attention was focused on the physical module. More concretely, Figure 5.37 presents the execution times of the mesh physical simulation step, that is, the FEM physical simulation step. The process consisted in calculating 10 different times of the physical module and compute the mean time deleting the extreme values in order to discard the outliers. As can be seen in the results, the execution time varies from 14.13 *ms* to 39.98 *ms* depending on the model.

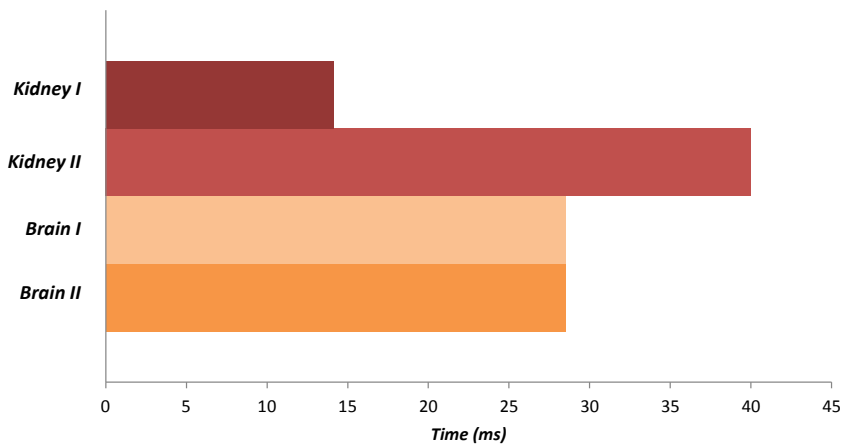


Figure 5.37: Execution times (in *ms*) for the physical simulation step for four models: *Brain I*, *Brain II*, *Kidney I* and *Kidney II*.

This range on the execution time is due to two reasons. Firstly, the execution time depends on the number of tetrahedrons of the model. As might be expected, the larger number of elements, the larger the execution time required. Secondly, the execution time depends on the time step set for each model. The smaller the time step, the larger the total execution time. Besides, the time step depends on the total simulation time of the experiment and on the mechanical properties of the material. The time step has been selected in each case in order to ensure the stability of each simulation.

5.8 Discussion

Recovering the 3D shape of 3D non-rigid objects is an ill-posed problem due to depth ambiguities. In contrast to the existing solutions, the solutions described in this chapter carry out deformable registration for non-rigid 3D objects. These systems rely on two different physics-based model simulations that allow preserving the physical behaviour of the solid, that is, they provide realistic behaviour to the vertex movements of the models. Specifically, the two main physical solutions that have been used to register the deformations are MSM and FEM.

The mass-spring mechanical model captures the natural behaviour in order to recover the non-rigid transformations in real time. Furthermore, the proposed system uses an RGB-D camera that, together with the physical simulation, enables the deduction of the new transformation of the 3D triangles of the mesh without the need of well-textured objects. In addition, it includes a tracking module based on templates to deal with scenes that lack texture and to retrieve the camera pose. Thus, the object can be moved while its deformations are calculated.

Additionally, a set of experiments has verified the quality of the results, demonstrating a correct visual representation of the solid in line with a low computational cost. The robustness has also been tested by adding noise to the data. Regarding occlusions, it is possible that with large occlusions the camera pose may be lost (as with non-textured tracking systems). Nonetheless, the method handles deformations with small occlusions (for example tools manipulating an object (Haouchine et al., 2013b)).

The FEM solution, in turn, consists in a more complex system that achieves higher levels of accuracy at the expense of increasing the computational cost. Unlike MSM, in this case the mesh is composed of tetrahedrons. Moreover, a characterization stage has been performed for optimal parametrization of the materials, which ensures that the level of realism is higher when a deformation is applied. Similarly, a set of experiments has presented to evaluate the robustness (error estimation) and computational cost of this second proposal.

Apart from evaluating the two physical formulations independently, a comparison between MSM and FEM solutions has been performed in terms of computational cost and error estimation. It has been concluded the theory that FEM is more accurate than MSM, but MSM offers a real-time performance

with an easier system. In addition, both solutions return a correct visual feedback in order to develop interactive applications related to AR.

Finally, a robot-assisted surgical system combined with a deformable object tracking has been presented, giving a visual feedback of the deformation state of non-rigid object when a force is applied to it. This is made by relying on an accurate and fast enough biomechanical model of the object combined with a visual tracking approach designed to properly simulate the model.

Experiments have been performed using two deformable objects (porcine *Kidney* and calf *Brain*). The obtained results have been compared with the theoretical outcome obtained from Abaqus, in order to assess the method, in terms of accuracy and computational cost. The deformation values obtained using the online FEM are smaller than those obtained by Abaqus software. This is due to the fact that while the input of the Abaqus simulation is the displacement recorded by the robot, the input of the FEM is the displacement acquired by the cameras. However, these cameras do not get the exact displacements because of the occlusion of the tool or the material itself. These occlusions can also affect camera tracking (as with non-textured tracking systems). In the same way, it has been developed a comparison between the results of FEM with the scanner ground truth. It has been observed that deformations simulated with Abaqus are larger than those obtained by the scanner. This is because the scanner is not able to acquire the real deformation due to the occlusion of the tool. That is, the scanner acquires more accurate deformation than the cameras, but the deformation is less accurate than the one obtained with the simulation performed with Abaqus. To improve the accuracy of the results higher density meshes could be employed. However, this will increase the computation time. Depending on the concrete application where the method is applied the correct balance between precision and computational cost needs to be found, as it is the case in most real-time applications. However, as it can be seen in Table 5.9, the average errors obtained in the experiments with the current implementation are low enough to be considered valid to serve as a basis for visual feedback in surgery. The method returns a deformation of the tested objects that matches the theoretical and experimental results obtained, with a precision that enables the development of assistance surgery applications.

Part III

Conclusions

Conclusions and future work

*Benvolgut, ho deixo aquí,
que sé que ets un home ocupat.
Suposo que és moment d'acomiar-me esperant
no haver-te empenyat massa,
no haver semblat un boig,
que la força ens acompanyi, adéu, fins sempre, sort!*

MANEL

This chapter presents the main conclusions reached during the course of this thesis, as well as the future research lines in which the present work can be complemented.

6.1 Conclusions

This thesis provides an overview of the different optical tracking techniques that exist to date. More precisely, this work is focused on improving and developing tracking systems towards to 3D deformable shapes. It follows a progressive analysis from the tracking of rigid planar surfaces such as markers until handling deformations of non-rigid non-planar 3D objects. Furthermore, it ensures a real-time performance, since it is considered a determining factor for developing *Augmented Reality (AR)* applications in order to give a correct visual feedback in a smooth manner.

Marker-based tracking systems have proved to be efficient and low-cost techniques. However, they continue having problems when dealing with

occlusions. Therefore, a solution has been developed in order to keep the tracking in case occlusion occurs.

With regard to the management of deformations in real time, two main methods have been presented. The first solves the detection and tracking of non-rigid planar surfaces as well as the deformations that are applied to the surface. The second, in turn, infers the deformations that a body suffers when a force is being applied. Furthermore, it uses a physical method to obtain a correct physical behaviour. Moreover, since *Computer Vision (CV)* and AR are recent technologies and are constantly progressing, the last contribution of this thesis uses a more specific technology such as RGB-D camera instead of using conventional web cameras as with first approaches.

The main contributions of each of the three phases that compose the thesis are described below.

1. *Rigid surface marker tracking*

Marker-based tracking systems, such as ARToolKitPlus library, are very reliable, robust and fast methods. However, the main drawback is the tracking failure when a minimal occlusion occurs. In this sense, a new marker design for the treatment of occlusions has been described.

It is based on customizable textures that are placed along the frame of the marker (unused area), which provide extra information during the occlusion. Furthermore, these textures are customizable, that is, they could have their own commercial design for marketing purposes. In this particular case, the design is focused on ARToolKitPlus markers, but it can be also adapted to any marker that does not codify information on its frame, making it highly adaptable.

In addition, the system is backward compatible since the proposed tracking method complements the ARToolKitPlus pipeline. Therefore, the main functionality of ARToolKitPlus is not altered but the new tracking system helps when it fails. This proposal offers an incremental tracking that runs in real time and combines two methods: one is based on appearance while the other is based on temporal coherence. The first method, *Tracking-by-Detection (TbD)*, creates a database of the marker (based on the *Scale Invariant Feature Transform (SIFT)* descriptor) in a preprocess stage in order to detect it on the online execution. This method serves as both initialization and also as recovery in case of failure. The second method, *Frame-to-Frame tracking (FtF)*, consists in measuring the

2D displacements of the visual cues using Lucas-Kanade optical flow procedure. Accordingly, the ARToolKitPlus pipeline will operate until an occlusion occurs and consequently the system fails. At this time, the FtF from the proposed system will start. If a failure occurs with the latter, the TbD automatically attempts to detect the marker in the image.

Finally, two user interaction interfaces for marketing opportunities have been introduced to show the new possibilities that can manage during the marker occlusion.

2. *Non-rigid 3D surface tracking*

An alternative method has been introduced in order to simultaneously calculate the 3D structure and the camera pose of a surface in an input image. The provided solution consists in creating two main different databases. One database, called *appearance database*, is created with the descriptors of the texture of the surface. For this purpose, the SIFT descriptor is used. It serves to detect the texture of the surface in runtime execution as if it were a rigid problem (*Initialization* step). The second database, *deformation database*, contains a set of deformation templates without any prior knowledge of the physical properties of the object. It is based on the curling of a sheet of paper and the most common deformations are stored in this database. The idea is to find the most similar template in the database in the online execution (FtF step).

The novelty of this system relies on a *Particle Filter (PF)* that obtains real-time performance through an efficient search along the deformation database to solve both problems simultaneously. Additionally, a *Design of Experiments (DoE)* has been presented in order to achieve the optimal parametrization of the PF.

Finally, the achieved accuracy level is good enough to give a correct visual feedback with a real-time performance.

3. *Deformable 3D object tracking*

A solution to compute simultaneously both the camera pose and the new 3D deformable structure of an object in real time has been proposed.

The registration and recovery of 3D deformable objects combine two main phases. The first is focused on solving the visual part of the problem while the second simulates the deformation of the object.

For the image processing, a more sophisticated image capturing device such as Kinect has been used. This camera information which is incomplete

and contains much noise is adjusted to the 3D model through a fast and intelligent search that establishes a set of correspondences between the 3D model and the input point cloud.

The deformation step in turn, feeds the input of a physical simulation system with the computed displacements. More specifically, two different types of physical formulations have been studied; one based on a *Mass-Spring Model (MSM)* system, and the other based on *Finite Element Methods (FEM)*. The two formulations have also been compared in terms of computation time and error accuracy. It has been verified that the performance of the MSM is better than the FEM. However, the accuracy level is higher in the latter. Even though, the visual feedback obtained by MSM is correct enough to give a realistic physical behaviour and, it runs in real time.

Finally, the accuracy level achieved with the FEM formulation has enabled us to develop the basis of a new AR prototype for surgical interventions. This has allowed us to introduce the exposed theoretical solution in the field of medicine, thus giving a surgical application context. It combines a surgical robotic arm with the proposed vision system to obtain a visual feedback of the stress state of the body tissue under surgical loads. Thus, it is possible to know in a visual way if the bodies are being damaged.

6.2 Future Research Lines

The contributions obtained in this thesis suggest several future lines of research.

1. *Rigid surface marker tracking*

The fact that AR is an emerging technology and with considerable growth potential in mobile platforms, it would be recommendable to adjust this work for these gadgets. Therefore, it is necessary to study the optimizations that are required to achieve robustness and real time in these devices, which have limited capabilities.

Finally, for reducing the computational cost of this solution, the TbD step should be improved.

2. *Non-rigid 3D surface tracking*

In terms of the database generation, it would be convenient to use this type of formulation in conjunction with a dimensional reduction technique such as *Principal Component Analysis (PCA)* to retain the most significant modes of deformation and thus improve the performance.

It would also be interesting to experiment not only with smooth movements but with strong ones where problems like the loss of information (that involve outliers) are dealt with. For that purpose, a new robust tracking combined with a sophisticated search in the database could be an interesting solution.

To improve accuracy, a possible improvement could be to combine this method with existing ones that offer more robustness but the computational cost is higher. Hence, considering the results of the system as an initial approximation of the deformed structure and its corresponding pose, a more robust system can be fed with this information in order to achieve high levels of accuracy in real time.

On the other hand, it would be appropriate to extend the proposed *Initialization* step in order to estimate both the pose and the deformed 3D structure without the need of treating the surface as a rigid object when calculating the first camera pose. Consequently, it can be provided a TbD method.

Finally, since the method is highly adaptable, the 3D reconstruction can be resolved not only with the reprojection error but also by adding other features such as colour recognition.

3. *Deformable 3D object tracking*

Establishing correspondences between the keypoints and the input point cloud is the bottleneck of the physics registration step. Therefore, a methodology that ensures correct matching and reduces the computational cost is an issue that will be addressed by the authors in the future.

Through the proposed surgical system it would be possible to automatize the movements of the robotic arm. The aim for the surgeon would be to select a point of the 3D model through the visualization software and the robot would automatically move to that point and apply an incision. All this leads to moving the arm robot and to obtaining visual feedback in the same application. Furthermore, this information should be shown instantly. Thus, since the FEM physical formulation uses a parallelizable technique, facilitates future improvements such as a GPU implementation in order to guarantee real-time performance.

Finally, it has to be emphasized that the flexibility of the presented framework allows incorporating new tracking methods that fit the properties of the algorithm. This helps to reduce the effects of noise in the data and the computational cost.

Part IV

Appendices

Homography

The following sections describe the concept of planar homography, as well as the algorithm which will be useful to estimate such matrix of transformations given various correspondence points. A more detailed explanation of this procedure can be found in (Hartley and Zisserman, 2003).

A.1 Definition

A *homography* defines a global movement or mapping that arises between two planes (see Figure A.1). In the particular case of vision systems, it consists in relating two images in perspective where the points or lines of a plane of one image in the scene are matched against points or lines in another image. This matching is valid only when the scene is plane or the displacement between the two images is small. This concept is also known as *2D projective transformation* since it is a linear transformation on homogeneous 3-vectors represented by non-singular 3x3 matrix.

Thus, the association between a world and an image plane is performed by a 2D-to-2D projective warp. In other words, for all points X in the world plane represented as $X = (X, Y, 1)$ in a homogeneous coordinate system can be related to all the points $x = (x, y, 1)$ in the image space, defining the correspondences $X \leftrightarrow x$ by:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = H \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \sim \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} * \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}, \quad (\text{A.1})$$

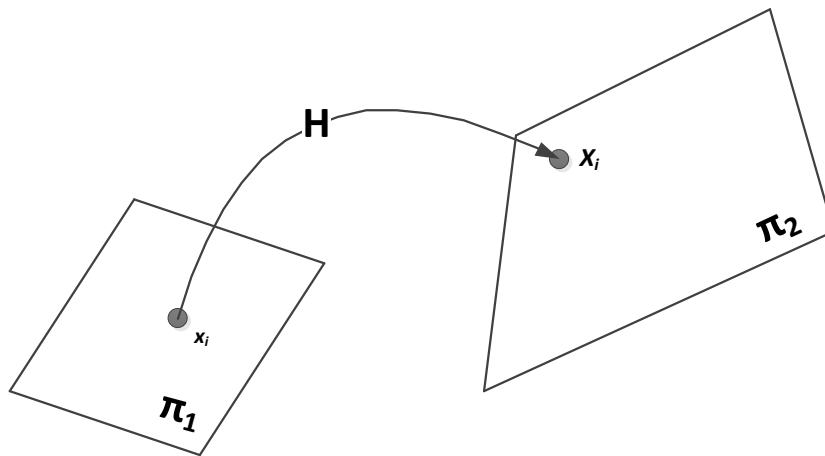


Figure A.1: The mapping of points from π_1 plane to π_2 plane.

where the 3×3 matrix H is the planar homography.

A.2 The Direct Linear Transformation (DLT)

The basic *Direct Linear Transformation (DLT)* algorithm is a simple linear algorithm that given a set of n correspondences ($n > 4$) $X \leftrightarrow x$, determines H (homography) where $x_i = HX_i$. This way, H transformation between two planes can be computed from four point correspondences, as each point correspondence provides two constraints. Additionally, these correspondences should not form triplets of collinear points in order to avoid a degenerate transformation and obtain a unique solution of H .

By cross-multiplying the third components of each side from Equation A.1, a linear system of two equations can be obtained from the elements of H :

$$\begin{aligned} x(h_{31}X + h_{32}Y + h_{33}) &= h_{11}X + h_{12}Y + h_{13} \\ y(h_{31}X + h_{32}Y + h_{33}) &= h_{21}X + h_{22}Y + h_{23} \end{aligned} \quad (\text{A.2})$$

Thus, given at least four such correspondences, a system of eight linear equations can be obtained in order to solve the elements of H :

$$\begin{pmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -x_3Y_3 & -x_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -y_3X_3 & -y_3Y_3 & -y_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -x_4Y_4 & -x_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -y_4X_4 & -y_4Y_4 & -y_4 \end{pmatrix} h = 0, \quad (\text{A.3})$$

where h is a 9-element vector containing the h_{ij} elements in the form $h = [h_{11}h_{12}h_{13}h_{21}h_{22}h_{23}h_{31}h_{32}h_{33}]^T$

Since this matrix equation is in the form of $Ah = 0$, the solution is the null space of A and can thus be computed using known methods, such as *Singular Value Decomposition (SVD)* (Flaquer et al., 2004). More precisely, $A_{8 \times 9} = U_{8 \times 9}D_{9 \times 9}V_{9 \times 9}^T$, where $D = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_9)$ is the diagonal matrix of singular values arranged in descending order down the diagonal and the matrices U and V are orthonormal. The columns of V are the eigenvectors of $A^T A$ and the required solution (\vec{h}) is the column of V corresponding the smallest singular value (σ_9 , last column).

DLT minimizes the algebraic error (minimizes the norm $\|Ah\|$), which is not geometrically or statistically meaningful. However, it has a low computational cost and offers a linear (and consequently unique) solution, so it is used as a starting point for a non-linear minimization of a geometric or statistical cost function.

A.3 Pose Estimation from a 3D Plane

In the particular case of obtaining the transformation matrix R_t of a plane, $Z = 0$ is considered in order to solve the planar homography in a simple way. Thus, given the known camera intrinsic parameters (K) and the image projection of a 3D planar structure, the camera extrinsic parameters (R_t) can be determined. The relation between the coordinates of points that lie on a 3D plane ($X_i = (X, Y, 0)^T$) and its image projection ($x_i = K R_t X_i$) can be represented using homogeneous coordinates as:

$$x_i = K \begin{bmatrix} \vec{r}^1 & \vec{r}^2 & \vec{r}^3 & \vec{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K \begin{bmatrix} \vec{r}^1 & \vec{r}^2 & \vec{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \quad (\text{A.4})$$

where \vec{r}^1 , \vec{r}^2 and \vec{r}^3 are the first, second and third column of the rotation matrix (R) respectively, t is the translation vector and H is a 3x3 homogeneous matrix, called as *homography matrix*.

The matrix H can be estimated from four correspondences $\vec{m}_i \leftrightarrow \vec{M}_i$ using the DLT algorithm (see Section A.2). Furthermore, since K is known, the camera pose can be recovered from the product $K^{-1}H$, where the last column \vec{r}^3 is computed as the cross product of \vec{r}^1 and \vec{r}^2 ($\vec{r}^3 = \vec{r}^1 \times \vec{r}^2$) to satisfy the orthonormality constraint of the rotation matrix R . Indeed, the orthonormality conditions are never perfectly met, so a renormalization step is applied (Simon and Berger, 2002): $\vec{r}^2 = \vec{r}^2 / \|\vec{r}^2\|$, $\vec{r}^3 = \vec{r}^1 \times \vec{r}^2 / \|\vec{r}^1 \times \vec{r}^2\|$, $\vec{r}^1 = \vec{r}^2 \times \vec{r}^3$.

Resolution of Kinect depth data

The following sections provide the resolution of the depth data from a Kinect device as well as a comparison between different depth sensors that can be found in the market.

B.1 Depth Resolution

Knowing the relative geometry between the IR projector and the IR camera (defined as baseline), the measurement of depth is determined by a triangulation process. Depths of objects in the scene are calculated by comparing the received pattern against the emitted reference pattern. This reference pattern is previously defined (memorized at a known distance from the sensor) and stored in the memory of the sensor. For each projected dot, if the position is different from that on the reference plane, the position of this dot in the infrared image will be shifted in the direction of the baseline. All these shifts form a disparity image using a simple image correlation procedure.

As Figure B.1 depicts, the distance of the dot projected on the point k of the object is smaller than that of the reference plane point o . By similar triangles this displacement D yields in a disparity d . Therefore, for each pixel, the distance to the sensor can then be retrieved from the corresponding disparity.

$$\frac{D}{b} = \frac{Z_o - Z_k}{Z_o},$$

$$\frac{d}{f} = \frac{D}{Z_k}, \tag{B.1}$$

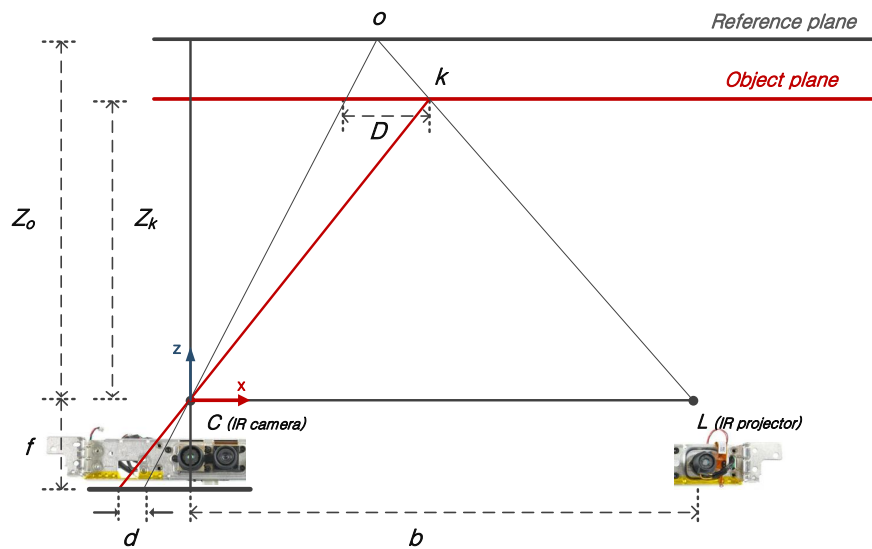


Figure B.1: IR geometric model. Relation between relative depth and measured disparity. Original source (Khoshelham and Elberink, 2012).

where Z_o is the distance from the reference plane to the sensor, Z_k denotes the depth of the point k in object space and f is the focal length of the IR camera.

Substituting D in Equation B.1:

$$Z_k = \frac{Z_o}{1 + \frac{Z_o}{f}d} \quad (\text{B.2})$$

Figure B.2 shows the depth map produced by the Kinect sensor. The depth values are encoded with grey values. The closer are the distances from the sensor the darker are the pixel values, and viceversa. On the other hand, black pixels indicates that there are not depth values available because they are out of the depth range, sensitive to external infrared sources (sun light) or crystalline or highly reflective objects. It may be also possible to obtain the raw data represented as a point cloud. This point cloud has a lot of noise with large holes, as Figure B.2 down depicts.

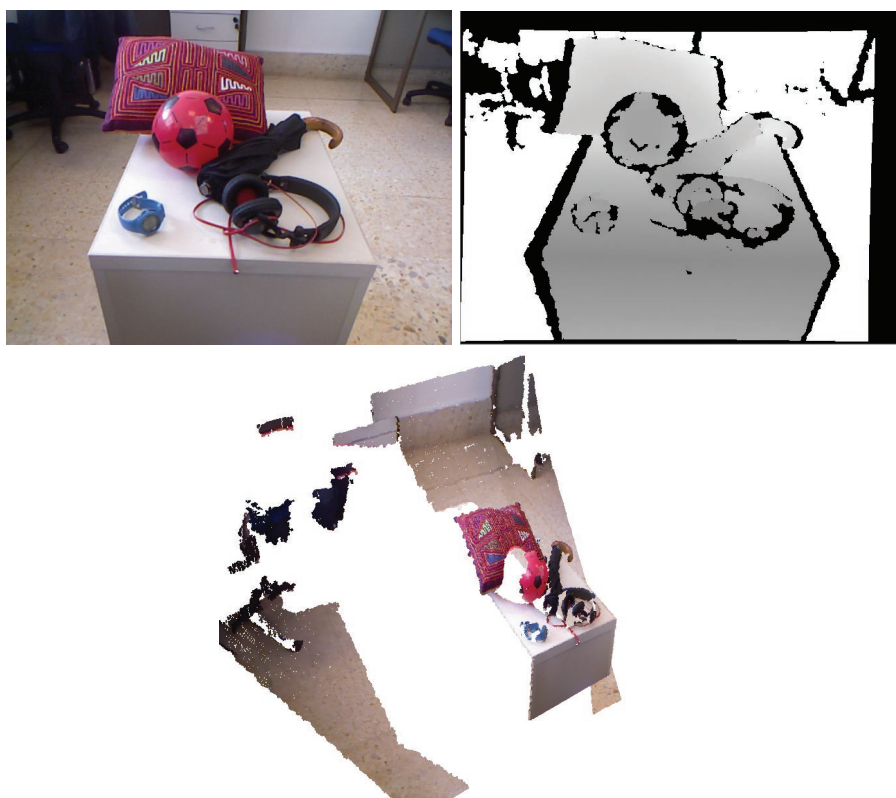










Figure B.2: Depth image (right-top) for a given scene (left-top) and its corresponding point cloud (down).

B.2 Depth Sensor Comparison

Several depth sensor devices that can be found in the market are presented.

Table B.1: Comparison between different depth sensors.

								
Operation Range (m)	0.8-3.5	0.35-1.4	0.8-4 / 0.5-9.7	0.5-4.5	0.35-3	0.8 - 3.5	0.8 - 3.5	0.8 - 3.5
FoV (H/V/D)	57.5,45,69	57.5,45,69	57,43,70	70,60	45,57,5,69	58,45,70	58,45,70	58,45,70
Spatial. x/y Resolution	@2m - 3,4 mm	@0.5m - 0,9 mm	@2m - 3mm	n/a	@0.5 - 0,9mm	n/a	n/a	n/a
Depth Resolution	@2m - 1,2 cm	@0.5m - 0,1 cm	@2m - 1 cm	n/a	@0.5 - 1mm	n/a	n/a	n/a
Max fps rate	60	60	30	30	30	30	30	30
Colour Image	640x480	640x480	640x480	1920x1080	240x320	n/a	n/a	1280x1024
Depth Camera	640x480	640x480	640x480	512x424	240x320	640x480	640x480	640x480
Dimensions (cm)	18x3,5x2,5	18x3,5x2,5	27,94 x 7,62x 7,62	24,9x6,6x6,7	17,8x12,9x3,3	18x3,5x5	18x3,5x5	18x3,5x5
Type of measurement	Structured light	Structured light	Structured light	Time of flight	Structured light	Structured light	Structured light	Structured light
Developer	PrimeSense	PrimeSense	Microsoft	Microsoft	Cubify	Asus	Asus	Asus

Design of Experiments

This appendix describes the methodology and each of the phases of the implementation of a *Design of Experiments (DoE)*, as well as the Case Study concerning Chapter 4.

C.1 Description of the Methodology

By *experiment* is understood the performance of a test under a controlled process with a predetermined objective. The *Design of Experiments (DoE)* is defined by the methodology of applying statistics to the experimentation process (LM., 2005). It consists in planning and developing a set of experiments inducing deliberating changes to the input variables of the process in order to identify possible changes in the output variable, also called *response*.

The model of Figure C.1 shows a general framework for the process of experimentation. The main idea is to reduce the impact of the identified problem by finding an optimal parametrization through the DoE. A description of each step of the methodology is presented below.

Define

This phase consists in defining the response and establishing the goals and objectives to be achieved.

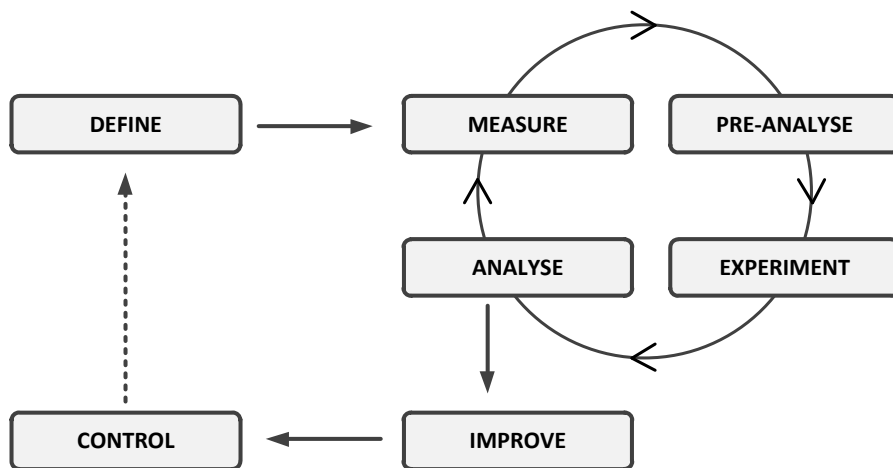


Figure C.1: Model of DoE methodology. (Courtesy of (Tanco et al., 2009)).

Measure

It identifies and classifies the variables that may influence the response. These variables are known as *factors*. Moreover, the *primary factors* are also known as those factors whose influence in the response is unknown and are wished to study. Furthermore, those factors that are always controllable are called *control-factors*, i.e., the *ranges* and *levels* (different values of the factor which are experimented) can be set for each one.

As described in (Castillo, 2007), depending on some carried out pilot tests and in the experience, three different ranges for factors can be distinguished

1. *Region of Operation*. This is the range where theoretically the experiments can be conducted.
2. *Region of Interest*. It is the range that is wished to be explored. Included in the *Region of Operation*.
3. *Region of Experimentation*. Included in the *Region of Interest*. This is the range of each primary factor, once the experiment has been designed (*Pre-Analyse* stage).

Pre-Analyse

This is the phase where the type of design is selected to be set up. More concretely, the total number, the conditions and the order in which the experiment will be conducted is determined. The most frequently used experimental designs are the following ones:

- *Full Factorial*. It includes all possible combinations of the levels of every factor. The design is defined as N^k , where N is all the possible combinations of the defined levels of the k factors involved in the experiments. Therefore, the number of experiments is the product of the number of levels of each factor. Experiments with factors at two levels (2^k , where k =factors) are very efficient and allow estimation of all effects.
- *Fractional Factorial*. It reduces the number of runs.
- *Plackett-Burman*. They are used for screening experiments, i.e., there is no idea of the process and there are many factors.
- *Central Composite design*. It lets the estimation of quadratic models. It is used for optimization processes, or when the response is not lineal.
- *Box-Behnken*. They are used for three level factors as well as to fit a quadratic model.

Experiment

This stage executes all the experiments to determine which factors are statistically significant and recollects all the information which is concluded with *Analyse* step.

Analyse

This phase uses statistical methods to analyse the data collected. The use of statistical software for the construction and the analysis of the designs, presents major advantages and allows to save time and not to deepen the theoretical aspects required to construct a design. The Minitab is the priority due to its graphics supports that allows taking wise decisions in a simple and effective way.

1. Determine active primary factors.

In a first step, the effect of each factor as well as the interactions between them is calculated. The former expresses the variation of the response caused by changes of levels of the input factors, while the latter corresponds to the impact whereby the apparent influence of a factor in the response depends on one or more factors. This information suggests which factors are statistically *significant*. *Significant* are those factors with a low probability of error that influence the response. An *Analysis of Variance (ANOVA)* is used to determine, at a given level of confidence, which effects influence the response, that is, to determine active primary factors. ANOVA is a formal and an accuracy statistical method which consists in assigning the total variance to the factors in order to perform statistical tests (*t*-test and *F*-test) and to know with a certain level of confidence, the effects that significantly influence the response. In this regard, the *p*-value of each factor is the index which shows the level of confidence. Thus, it is reported that an effect is considered as significant if its *p*-value is lower than 0.05.

2. Interpret results.

The former two steps (*Experiments* and *Analysis*) are executed sequentially and in an iterative manner. Based on the range of interest that has been previously assigned to each factor, local ranges of experimentations are selected. Then, in view of the effects achieved of the factors through the *p*-values and related interactions between them, a new range is selected (either higher or lower depending on the response) for the significant factor. In addition, all the non-significant factors are excluded from the following analysis, setting for them the last assigned value.

Besides the ANOVA, the presentation of the factors in a *Pareto Chart* (see Figures C.2, C.3) as well as the *Normal Probability Plot* (see Figures C.4, C.5) also allows to confirm the information in a graphic manner. The *Pareto chart* represents the magnitude of all the effects through the indicative line that specifies which are the significant ones. On the other hand, the second plot visualizes the significant effects, which are those that are not distributed along the straight line. Moreover, apart from the factors shown in the previous tables (see Figures C.2, C.3 and Figures C.4, C.5), it should be also mentioned that these plots show the interactions between factors that are significant.

3. Optimization.

Finally, it is possible and often necessary to select more than one response. In this regard, an optimization is introduced to get the parametrization for multiple responses and finish with the *Analysis* stage. It searches for a combination of the factors that jointly optimize the two responses by satisfying the requirements of each one. This can be done by acquiring some knowledge of the process and the ranges of the responses. In this sense, Minitab offers a simple method called *Response Optimizer* tool that calculates an optimal setting for the responses. It searches for a combination of the factors that jointly optimize the responses by satisfying the requirements of each one.

Moreover, multiple responses are combined into a single response using desirability functions. After calculating the desirability (d) (individual response), Minitab combines them to get a measure of the composite (D), which depends on the weight that has been set to each response.

The plot for the optimization in the *Case Study* (see Figure C.6) shows the effect of each factor on the two responses. The red vertical line determines the current desirable optimization settings according to the top displayed numbers. The horizontal blue lines and the corresponding numbers in the first column represent the value of the response for the current factor level. Finally, in each response the *individual desirability* (d) is shown, while the *optimal desirability* (D) is shown in the top. If these values (d -error, d -execution, D -composite) are close to 1, then the results that are wanted to achieve would be better.

The last two phases of the model, *Improve* and *Control*, are used to maximize or minimize the goal and thus, monitor and verify stability. These last two stages have not been taken into account in this thesis.

C.2 Case Study

Define

The error (inversely proportional to the weight of each particle) and the execution time of the *Particle Filter (PF)* were selected as responses, i.e., variables that are wished to measure as the result of the evaluated process. The target was global parametrization that minimizes both responses. Furthermore, two textures (*Stones* and *Guernica*) were used (see Table 4.1 top and middle) with a different number of feature points in order to attain general parametrization.

Measure

The primary factors observed in Table C.1 were identified. *Number of particles (A)* corresponds to the total number of particles to process in the PF (see Section 4.4.2.2), *DefSame (B)* is the degree of deformation to search for in the same cluster (see Section 4.4.2.2), *DefAll (C)* is the degree of deformation to search for in the remaining clusters (see Section 4.4.2.2), *GroupDef (D)* is the threshold for grouping particles in order to calculate the pose (see Section 4.4.2.2) and finally, *Granularity (E)* corresponds to the number of templates stored in each type of deformation or cluster deformation (see Section 4.4.2.2).

Table C.1: Classification of the primary factors.

FACTOR	IDENTIFIER
Number of particles	<i>NumParticles (A)</i>
Deformation same cluster	<i>DefSame (B)</i>
Deformation all clusters	<i>DefAll (C)</i>
Group deformation	<i>GroupDef (D)</i>
Granularity-storage database	<i>Granularity (E)</i>

1. Region of operation (see Table C.2).
2. Region of interest (see Table C.3).
3. Region of experimentation (see Table C.4). Unlike the previous regions, here it was decided to delete, basis on the pilot testing, those values that were so high and whose results were not significant at first glance.

Table C.2: Region of operation of each factor.

IDENTIFIER	REGION OF OPERATION
<i>NumParticles</i> (A)	> 0
<i>DefSame</i> (B)	> 0
<i>DefAll</i> (C)	> 0
<i>GroupDef</i> (D)	> 0
<i>Granularity</i> (E)	> 28

Table C.3: Region of interest of each factor.

IDENTIFIER	REGION OF INTEREST
<i>NumParticles</i> (A)	[500 - 1000 - 1500 - 2000 - 2500 - 3000 - 3500 - 4000 - 4500 - 5000 - 6000 - 7500 - 10000]
<i>DefSame</i> (B)	[5 - 10 - 15 - 30 - 50]
<i>DefAll</i> (C)	[5 - 10 - 15 - 30 - 50]
<i>GroupDef</i> (D)	[1 - 5 - 10 - 15 - 20]
<i>Granularity</i> (E)	[1400 (50) - 2800 (100) - 5600(200) - 8400(300) -11200(400) - 14000 (500)]

Pre-Analysis

It was decided to use a *full factorial design*. Thus, $N^k = 2^5 = 32$ experiments.

Analysis

Table C.5 depicts the p -values extracted from the execution of the experiment for both error and time cases for each texture. The ANOVA emphasizes that the most dominant factors were *GroupDef* and *Granularity*. *NumParticles* was also important, although the p -value associated with the error of the first texture was not fully reliable.

Optimization

In this case, both responses had the same weight.

In the time case, there was evidence that 20 *ms* is the largest value that can achieve in order to have real-time performances. Conversely, the error, the closer the value of this index was to zero, the higher level of robustness encountered.

Table C.4: Region of experimentation of each factor.

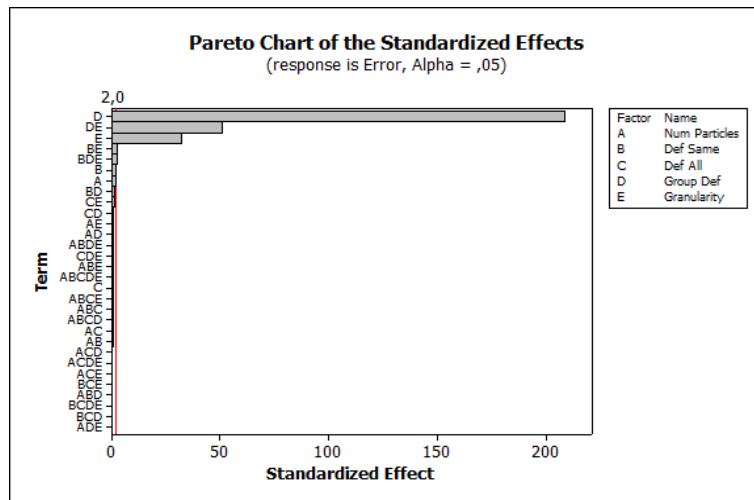
IDENTIFIER	REGION OF EXPERIMENTATION
<i>NumParticles</i> (A)	[500 - 1000 - 1500 - 2000 - 2500 - 3000 - 3500 - 4000 - 4500 - 5000]
<i>DefSame</i> (B)	[5 - 10 - 15 - 30]
<i>DefAll</i> (C)	[5 - 10 - 15 - 30]
<i>GroupDef</i> (D)	[1 - 5 - 10]
<i>Granularity</i> (E)	[1400 (50) - 2800 (100) - 5600(200) - 8400(300) -11200(400) - 14000 (500)]

Table C.5: ANOVA for *error* and *time*.

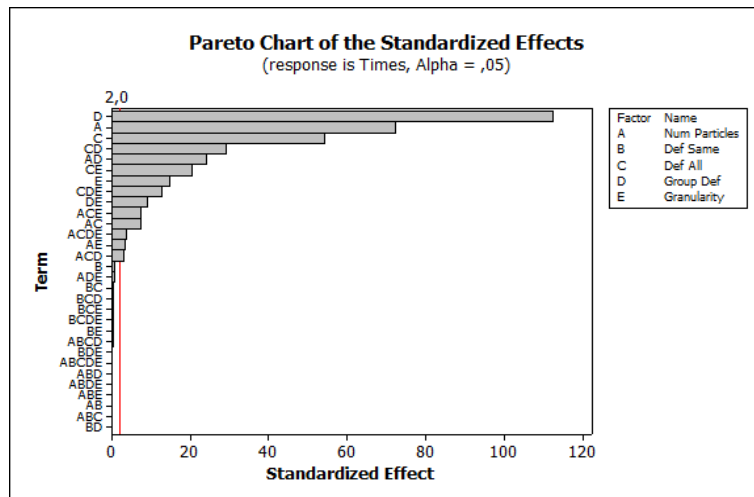
Texture	FACTOR	<i>p</i> -value error	<i>p</i> -value time
Stones	<i>NumParticles</i> (A)	0.060	0.000
	<i>DefSame</i> (B)	0.037	0.431
	<i>DefAll</i> (C)	0.678	0.000
	<i>GroupDef</i> (D)	0.000	0.000
	<i>Granularity</i> (E)	0.000	0.000
Guernica	<i>NumParticles</i> (A)	0.000	0.000
	<i>DefSame</i> (B)	0.598	0.941
	<i>DefAll</i> (C)	0.621	0.000
	<i>GroupDef</i> (D)	0.000	0.000
	<i>Granularity</i> (E)	0.000	0.000

In this case, the goal was to *minimize*, so the target value was determined for execution time in 20 (close to real time if it is considered additional cost associated with image processing) and 0 for error. With regard to the upper bound, a high value for both responses was fixed. The reason of selecting 20 in the time case instead of 0, was because of the prioritization of the error until this value and moreover, the smaller values were not taken into account. So any response value below the target the response desirability will be one and above the upper bound zero.

Analysing the results, was deduced that the best parametrization is 3500 for *NumParticles*, 5 for *DefSame*, 5 for *DefAll*, 1 for *DefGroup* and 14000 for *Granularity*.

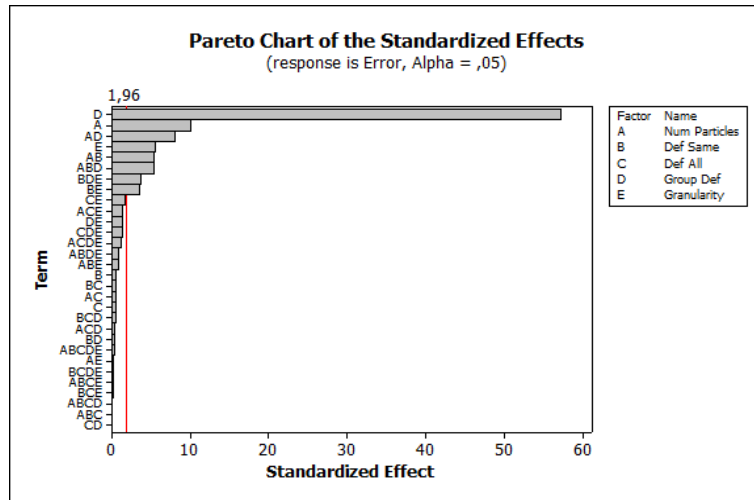


(a)

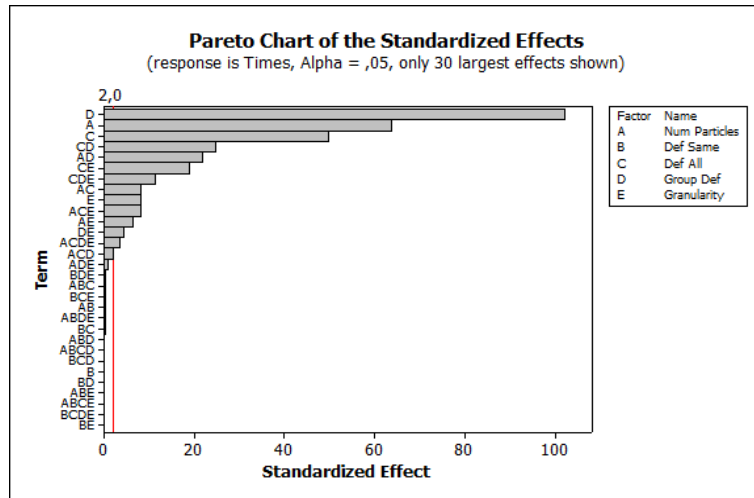


(b)

Figure C.2: Pareto charts of the *Stones* texture. All the effects through the indicative line that provides to know which are the significant ones. The chart (a) corresponds to the error response and the (b) for the time response.

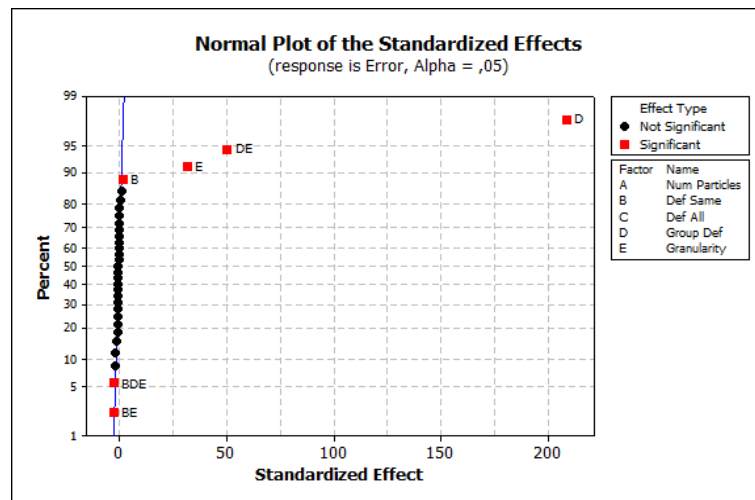


(a)

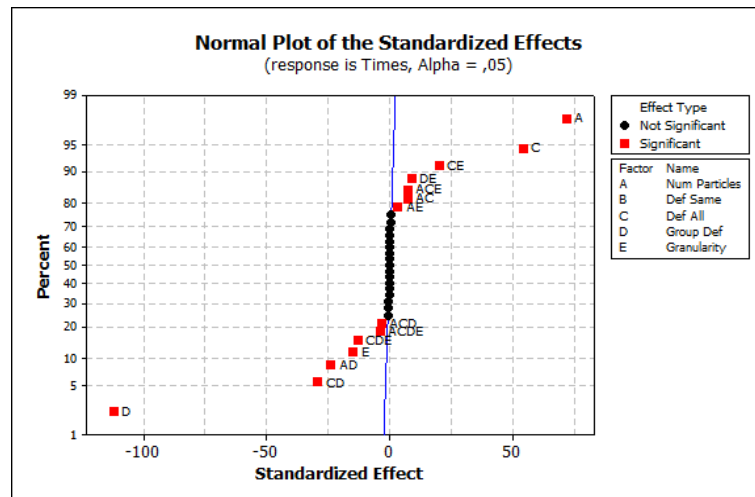


(b)

Figure C.3: Pareto charts of the *Guernica* texture. All the effects through the indicative line that provides to know which are the significant ones. The chart (a) correspond to the error response and the (b) for the time response.

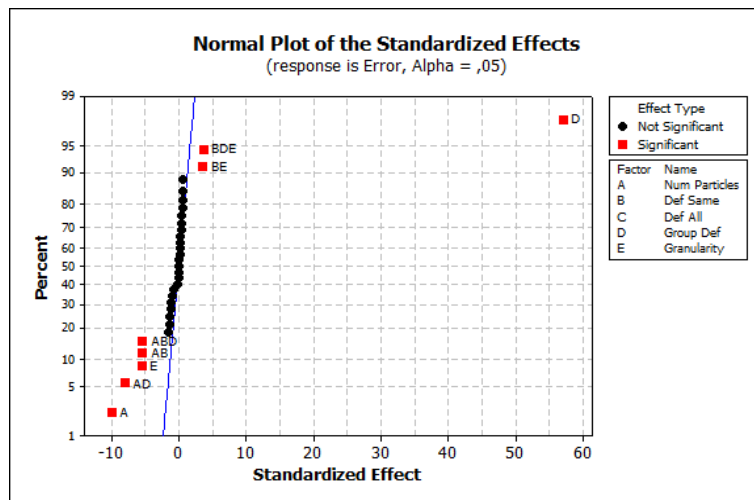


(a)

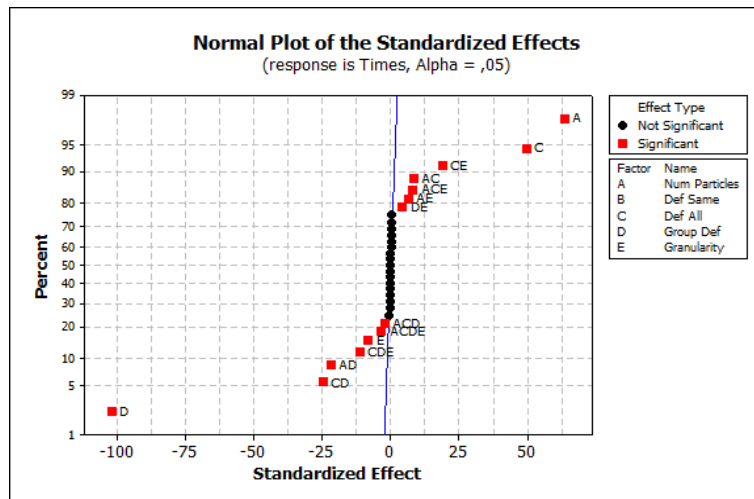


(b)

Figure C.4: Normal plots of the *Stones* texture. The significant effects are not distributed along the straight line. The chart (a) corresponds to the error response and the (b) for the time response.

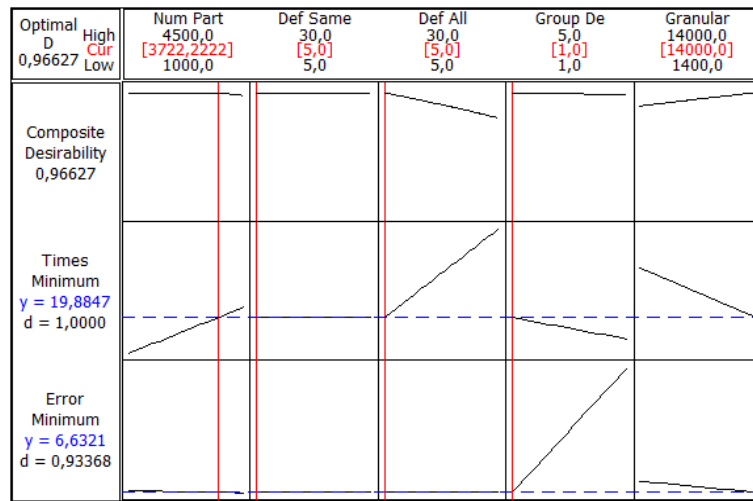


(a)

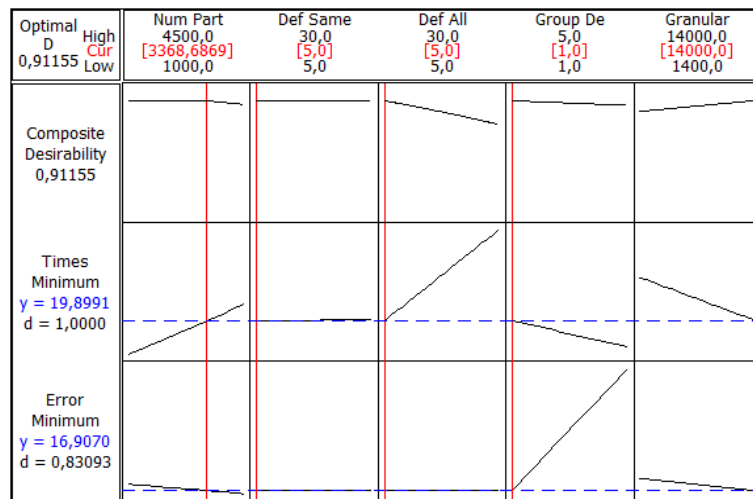


(b)

Figure C.5: Normal plots of the *Guernica* texture. The significant effects are not distributed along the straight line. The chart (a) corresponds to the error response and the (b) for the time response.



(a)



(b)

Figure C.6: Desirability plots depicting the optimal values for factors in an optimal combination with the desired targeted responses for *Stones* (a) and *Guernica* textures (b).

AR Framework links

The following provides a list of *Augmented Reality (AR)* SDKs.

Table D.1: AR SDKs.

SDK	Website
ALVAR	http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/index.html
ARTag	http://www.artag.net/
ARLab	http://www.arlab.com/
ARmedia	http://www.inglobetechnologies.com/
ARPA	http://arpa-solutions.net/en
ARToolKit	http://www.hitl.washington.edu/artoolkit/
ARToolKitPlus	http://studierstube.icg.tugraz.at/handheld_ar/artoolkitplus.php/
ArUco	http://www.uco.es/investiga/grupos/ava/node/26
BazAR	http://cvlab.epfl.ch/software/bazar/index.php
BeyondAR	http://beyondar.com/platform
Catchoom	http://catchoom.com/
D'Fusion	http://www.t-immersion.com/
FLARToolkit	http://www.libspark.org/wiki/saqoosha/FLARToolKit/en

Layar	https://www.layar.com/
Metaio	http://www.metaio.com/sdk/
NyARToolkit	http://nyatla.jp/nyartoolkit/wp/
OpenCV	http://opencv.org/
osgART	http://osgart.org/index.php/Main_Page
PCL	http://pointclouds.org/
PTAM	http://www.robots.ox.ac.uk/~gk/PTAM/
Qualcomm Vuforia	https://developer.vuforia.com/
Qualcomm FastCV	https://developer.qualcomm.com/mobile-development/add-advanced-features/computer-vision-fastcv/
Studierstube	http://www.icg.tugraz.at/project/studierstube
Total Immersion	http://www.t-immersion.com/ar-key-words/augmented-reality-sdk
Wikitude	http://www.wikitude.com/products/wikitude-sdk/

Generated Publications

This appendix includes the front page of the articles that have already been published by the author of this thesis in scientific journals and international conferences. Submitted articles that are currently under review are also included.

Journals

Álvarez, H., Leizea, I., and Borro, D. "A new marker design for a robust marker tracking system against occlusions". *Computer Animation and Virtual Worlds*, Vol. 23, N. 5, pp. 503–518. 2012.

Leizea, I., Álvarez, H., and Borro, D. "Real time non-rigid 3d surface tracking using particle filter". *Computer Vision and Image Understanding*, Vol. 133, N. 0, pp. 51–65. April, 2105.

Leizea, I., Mendizabal, A., Álvarez, H., Aguinaga, I., Sánchez, E., and Borro, D. "Tracking of deformable objects based on a visual approach and physics simulation for robot-assisted surgery". *Submitted to Computer Graphics and Applications, IEEE*, 2015.

Conferences

Álvarez, H., Leizea, I., and Borro, D. "A survey on optical tracking for augmented reality". In *Proceedings of the XXII Conferencia Española de Computación Gráfica (CEIG)*, pp. 31–40. Jaén, Spain. September, 2012.

Leizea, I., Álvarez, H., Aguinaga, I., and Borro, D. "Real-time deformation, registration and tracking of solids based on physical simulation". In *Proceedings of the 13th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 165–170. Munich, Bavaria, Germany. September, 2014.

RESEARCH ARTICLE

A new marker design for a robust marker tracking system against occlusions

Hugo Álvarez*, Ibai Leizea and Diego Borro

Ceit and Tecnun (University of Navarra), Manuel de Lardizábal 15, 20018 San Sebastián, Spain

ABSTRACT

Marker systems are a widely used optical tracking method that does not support occlusions. Thus, this paper proposes a new marker design to overcome the problem of marker occlusions. It is highly adaptable, because it can be used by any marker tracking system that uses its central area to codify the digital identification. Our proposal takes advantage of an untapped frame to place some textures that will be tracked during marker occlusion. In addition, these textures are customizable, which lets users make their own designs. Two tracking methods are combined to offer a robust tracking, updating the six degrees of freedom of the camera in real time. The first one is a fast technique based on temporal coherence, whereas the second one is a robust technique based on appearance, which is used as a recovery mode. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS

augmented reality; marker; tracking; occlusion

*Correspondence

Hugo Álvarez, Ceit and Tecnun (University of Navarra), Manuel de Lardizábal 15, 20018 San Sebastián, Spain.

E-mail: halvarez@ceit.es

1. INTRODUCTION

The aim of augmented reality is to add virtual objects to the real world, giving the impression that both worlds coexist. To achieve this goal, much effort has been made to obtain an accurate and robust tracking system based on computer vision. Markerless solutions use natural features [1,2] or a 3D model [3,4] to calculate the camera pose. These alternatives do not need environment adaptation, because they rely on natural features that are in the scene or lie on the surface of the model to be tracked. Thus, in some cases, a scene rich in texture is required, whereas in other cases, the 3D model must be known, which is not always easy to obtain. Marker tracking systems are an alternative to those problems, because they add special patterns to the scene [5–7]. Furthermore, they are faster, more accurate, and more robust than markerless tracking solutions. The main inconvenience is the environment adaptation, although in some cases, this adaptation is feasible and does not involve any obstacle at all. Besides the environment adaptation, marker occlusion is another shortcoming, as the system fails even if the marker is only slightly occluded (Figure 1, left). This produces an undesirable effect on users who lose the sense of realism. Because of this limitation, instead of using multiple markers so that there is always one marker visible [8], we have developed a new

marker design that deals with occlusions and updates the six degrees of freedom of the camera (Figure 1, right). Our proposal offers more robustness and does not need more environment modification.

We have chosen ARToolkitPlus [9] to introduce our ideas because it is a widely used noncommercial marker tracking system that uses markers that do not take advantage of their frame to codify information (Figure 2, left). Notice, however, that our method can be adapted to any marker that does not codify information on its frame. We have placed textured patches along the frame of the marker to have more visible features (with known 3D coordinates) during marker occlusion. Moreover, the content of these textures is customizable, which lets users design their own markers or expand marketing opportunities. In addition, the different occlusion states provide us with additional information that can be used to implement some human–machine interfaces.

The rest of the paper is organized as follows. First, some works related to marker occlusion will be enumerated. Afterwards, the choice of the new marker design will be justified. Then, all steps involved in the tracking will be described. Additionally, some experiments will demonstrate the performance of the proposed method. Finally, conclusions and future works will be cited.

Computer Vision and Image Understanding 133 (2015) 51–65



Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviuReal time non-rigid 3D surface tracking using particle filter[☆]Ibai Leizea^{*}, Hugo Álvarez¹, Diego Borro¹^a CEIT and Tecnun, University of Navarra, Paseo Manuel Lardizábal, 15, 20018 Donostia-San Sebastián, Spain

ARTICLE INFO

Article history:
Available online 17 December 2014

Keywords:
Real time tracking
Deformable tracking
Particle filter
Shape recovery

ABSTRACT

Recovering a deformable 3D surface from a single image is an ill-posed problem because of the depth ambiguities. The resolution to this ambiguity normally requires prior knowledge about the most probable deformations that the surface can support. Many methods that address this problem have been proposed in the literature. Some of them rely on physical properties, while others learn the principal deformations of the object or are based on a reference textured image. However, they present some limitations such as high computational cost or the lack of the possibility of recovering the 3D shape. As an alternative to existing solutions, this paper provides a novel approach that simultaneously recovers the non-rigid 3D shape and the camera pose in real time from a single image. This proposal relies on an efficient particle filter that performs an intelligent search of a database of deformations. We present an exhaustive Design of Experiments to obtain the optimal parametrization of the particle filter, as well as a set of results to demonstrate the visual quality and the performance of our approach.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Recovering the 3D shape of a non-rigid surface from a monocular video sequence is a highly ambiguous problem because many 3D surfaces could have the same projection (see Fig. 1). Even when we have the intrinsic camera parameters and a well-textured surface, it is difficult to select the best mesh between all possible configurations of a deformable surface in order to solve the depth ambiguities. Along with the standard approaches that try to solve the shape recovery problem, we find approaches that establish prior knowledge [60], have a reference image [40], or are even based on a set of images of the target object [6].

Some methods rely on modelling the physical properties of a surface to achieve an approximation of the physical behaviour [14]. Although they obtain accurate results, the challenge is to determine the physical properties that govern the surface behaviour. Others adjust the behaviour of a surface to the movements registered in a database [43], which contains the most representative deformations of the surface. There are other approaches [29] that reconstruct the deformation of a non-rigid surface by using 2D–3D correspondences between an input image and a reference image in which the 3D shape of the surface and the intrinsic

parameters of the camera are known. Likewise, some methods [4] that do not require prior knowledge have been proposed. They extract 2D–3D correspondences for each frame by tracking points over the video sequence.

The central idea of our method is to recover simultaneously and in real time the camera pose and the non-rigid 3D surface through the use of an efficient particle filter [19,37]. The particle filter performs an intelligent search in a database where a range of deformation templates and appearance descriptors are stored. Furthermore, two tracking methods are combined to offer robust tracking. The first one is based on appearance, which serves to do both the initialization and reset from a failure, while the second one is based on temporal coherence.

The rest of the paper is organized in 5 sections as follows. First, we provide an overview of some works related to recovering a deformable 3D surface and the justification for choosing our method is presented. In Section 3, we address all the steps that have been carried out to solve the problem of recovering a non-rigid 3D surface. Afterwards, in Section 4 a Design of Experiments is presented to determine the optimal values of the parameters of the Particle Filter, as well as, a test suite that demonstrates the good performance of our tracking method. Finally, conclusions and future works are enumerated in Section 5.

2. Related work

The non-rigid surface recovery problem has been addressed in several ways. Here we present a categorization of four main

[☆] This paper has been recommended for acceptance by Vincent Lepetit.

^{*} Corresponding author. Fax: +34 943 213 076.

E-mail addresses: ileizea@ceit.es (I. Leizea), hvalvarez@ceit.es (H. Álvarez), dborro@ceit.es (D. Borro).

¹ Fax: +34 943 213 076.

Tracking of deformable objects based on a visual approach and physics simulation for robot-assisted surgery

Ibai Leizea, Ainhitze Mendizabal, Hugo Álvarez, Iker Aguinaga, Emilio Sánchez and Diego Borro

Abstract—This paper presents a visual tracking system for deformable objects used with a robot-assisted surgical system. The system is capable of capturing in real time the deformation exerted by the robot using Computer Vision, and despite the occlusions produced by the robotic system itself. The information captured by this system enables new assistance approaches to assist surgeons during procedures, such as Augmented Reality. The tracking algorithm is helped by a fast and accurate physically-based biomechanical model of the object, together with a visual tracking approach designed to properly simulate the model.

This method has been tested in a realistic experimental setup, to evaluate the level of accuracy and computation requirements of the proposed system. The deformations obtained from the experiments have been compared with the theoretical results obtained by finite element analysis obtained using the Abaqus software package. The results obtained provide an accurate visual representation of the deformed solid.

Index Terms—Visual tracking, robotics, finite element method, material characterization.

1 INTRODUCTION

The use of robotic devices in medicine and more specifically in the surgical field entails a remarkable improvement in the delivery of healthcare services. Nowadays, robotic devices and computer-assisted technologies are included as a means of guidance, diagnosis, verification and general assistance during the performance of a surgical intervention [1].

However, the use of robotic devices in surgery creates new challenges that must be overcome for their successful deployment. A robot acting on a tissue exerts a force and produces deformations over which the surgeon has no direct feedback. The tracking of the deformation of the organs is required for assistive technologies such as Augmented Reality, for example to support the surgeon by a visual feedback of the deformations that are taking place on the tissues, or to support the surgeon by locating the position of a malignancy to be removed from an organ. In most cases, the inclusion of sensors to track the position of the organs of the patient is not usually feasible. Thus, the use of alternative approaches such as Computer Vision techniques has become a real need to solve this kind of problem.

Nevertheless, the very presence of the robot produces occlusions that limit how much information can be obtained using only Computer Vision. The combination of Computer Vision with a suitable physical simulation can be used to reduce the missing information. In this case, the accuracy required for the surgical simulations remains important. In contrast to deformable models used in video games and animations, the purpose of soft tissue models in medical simulation is to model realistically the behaviour of biological tissues. Consequently, the simulation of the deformation of the tissue should be controlled using real

material parameters. Thus, these parameters should be obtained from biomechanics experiments instead of intuitively adjusted parameters. However, in robotic guidance and surgery assistance systems not only accuracy is essential, but also computational performance must be taken into account, in order to provide a fast enough visual feedback to the surgeon. Therefore, in this application it is critical to obtain a compromise between accuracy and computational cost. Being able to provide surgical realism at interactive rates of simulation is one of the most challenging problem in robot surgical assistance systems [2].

This paper proposes a novel visual tracking method based on physical simulation. This procedure is capable of obtaining the deformation produced by a robotic system on patient tissues, and therefore, it obtains the basic information to provide assistance and guidance to surgeons using visual or haptic feedback. More concretely, this method has been integrated within the framework of robotic surgery system through a surgical module developed in [3].

The approach followed in this paper is divided into two main phases. The first deals with handling robotic surgical arm movements, while the second computes the deformations applied from the first step through a vision system. This vision system is composed of a RGB-D camera which returns both the colour and depth information. However, this data is incomplete (due to occlusions and areas where the sensor has not been able to capture data) and contains too much noise. Accordingly, we rely on a physical simulation module to reconstruct the object deformation. More concretely, we feed the input into a non-linear Finite Element Method (FEM) formulation in order to simulate the physical behaviour of the tissue under deformation. The mechanical properties of the deformable objects and tissues used in the experiments have been experimentally obtained

CEIG - Spanish Computer Graphics Conference (2012)
Isabel Navazo and Gustavo Patow (Editors)

A Survey on Optical Tracking for Augmented Reality

H. Álvarez and I. Leizea and D. Borro

Ceit and Tecnun (University of Navarra), San Sebastián, Spain

Abstract

Good tracking creates the sensation that virtual objects belong to the real world, offering a perfect alignment between virtual and real objects., i.e., it increases the effectiveness of augmented reality. Moreover, optical tracking is the most popular solution for addressing the tracking problem as it does not require that bulky machines be added to the scene or force the user to carry heavy devices, and instead a simple camera captures images of the scene. Additionally, it is also a low cost alternative. Due to this reasoning, optical tracking has been extensively studied by several authors, resulting in many different optical tracking methods. However, it is not a solved problem and it still remains as an important research topic because it is often difficult to provide robustness, accuracy and low computational cost at the same time. This article reviews the methods that have been designed to overcome the problem of tracking through using images from the camera as the sole source of information.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking

1. Introduction

Augmented Reality (AR) is a technology that enriches the way in which users experience the real world with additional virtual information (Figure 1). In contrast to Virtual Reality (VR), where the user is completely immersed in a synthetic world, AR consist on adding virtual objects to the real world.

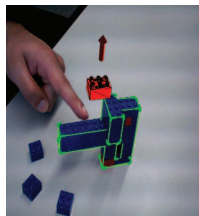


Figure 1: *Virtual instruction that illustrates how to disassemble a real object.*

The main challenge of an augmented reality system is to obtain robust and accurate *registration*. The registration problem is based on finding a perfect alignment between real and virtual objects, since it is essential to create the

illusion that virtual and real worlds coexist. To solve this problem, the position and orientation of the observer has to be determined. Using this information a virtual camera can be configured, which indicates exactly where the virtual objects should be drawn in the image. Likewise, the problem of finding the parameters that define the camera is known as *tracking*, which is the process that extracts the position and orientation (jointly called *pose*) of the camera relative to a global coordinate system (usually cited as world coordinate system). It requires the extraction of the 6 degree of freedom (DOF) that represent the user's viewpoint: 3-DOF for orientation and 3-DOF for translation. There are many alternatives to addressing this problem, and they differ in the type of sensors they use [RBG01]: inertial sensors combine accelerometers and gyroscopes to estimate the translation and rotations respectively, ultrasound sensors rely on the delay times of ultrasonic pulses to infer position and orientation, GPS receivers use the signals emitted by a set of satellites to triangulate its pose, magnetic sensors measure the magnetic fields to deduce the viewpoint parameters, and optical sensors process the image of the scene captured by a camera to obtain its corresponding 6-DOF.

Tracking based on optical sensors (also called *visual tracking*) is the most popular solution because it is inexpensive and does not require a significant adaptation of the en-

Real-time Deformation, Registration and Tracking of Solids Based on Physical Simulation

Ibai Leizea* Hugo Álvarez† Iker Aguinaga‡ Diego Borro§

CEIT and Tecnun (University of Navarra), Spain

ABSTRACT

This paper proposes a novel approach to registering deformations of 3D non-rigid objects for Augmented Reality applications. Our prototype is able to handle different types of objects in real-time regardless of their geometry and appearance (with and without texture) with the support of an RGB-D camera. During an automatic offline stage, the model is processed in order to extract the data that serves as input for a physics-based simulation. Using its output, the deformations of the model are estimated by considering the simulated behaviour as a constraint. Furthermore, our framework incorporates a tracking method based on templates in order to detect the object in the scene and continuously update the camera pose without any user intervention. Therefore, it is a complete solution that extends from tracking to deformation formulation for either textured or untextured objects regardless of their geometrical shape. Our proposal focuses on providing a correct visual with a low computational cost. Experiments with real and synthetic data demonstrate the visual accuracy and the performance of our approach.

Index Terms:

1.4.8 [IMAGE PROCESSING AND COMPUTER VISION]: Scene Analysis—; 1.6.8 [SIMULATION AND MODELING]: Types of Simulation— [1.2.10]: ARTIFICIAL INTELLIGENCE— Vision and Scene Understanding

1 INTRODUCTION

Recovering the 3D shape of a non-rigid object is a highly ambiguous problem because many deformable objects (with completely different shapes) could have the same projection. In Computer Vision, there are few solutions that solve the registration of a non-rigid 3D object. Moreover, the number of approaches decreases when a real-time constraint is set. This is mainly caused by the complexity of the recognition and non-rigid registration steps.

This paper proposes a real-time non-rigid 3D object registration method that is able to track and register 3D models regardless of their geometric shape. Thus, our system can be used, for example, to provide visual feedback for applications such as assembly of flexible components in industry or medical surgery. Additionally, detection is not based on features, and therefore, we can operate with textured or untextured objects. This makes the approach robust against sudden illumination changes. The proposed method not only obtains the deformation, but also the camera pose for each frame. The detection and tracking system is based on templates (LINEMOD method [9]). Unlike other solutions, our approach does not require a static camera. In fact, the method handles large camera movements.

*e-mail: ileizea@ceit.es

†e-mail: halvarez@ceit.es

‡e-mail: iaguinaga@ceit.es

§e-mail: dborro@ceit.es

IEEE International Symposium on Mixed and Augmented Reality 2014
Science and Technology Proceedings
10 - 12 September 2014, Munich, Germany
978-1-4799-6184-9/13/\$31.00 ©2014 IEEE

The process is divided into two steps. The first one takes advantage of an RGB-D camera to obtain both the colour and depth information. The second step, meanwhile, fits this information into a physical model. We rely on being able to model the mechanical properties of the solid to attain an approximation of the physical behaviour. More concretely, a Mass-Spring Model (MSM) is used to adjust the mesh and recover the new 3D shape due to the deformation. Finally, compared to existing solutions [7], our method offers accurate visual feedback, versatility (the ability to work with different types of objects) and real-time performance over accuracy.

The rest of the paper is organized in 5 sections as follows. In Section 2 we provide an overview of some works related to recovering a deformable 3D shape and the justification for our contribution. Then, in Section 3 we address all the steps that have been carried out to solve the problem of recovering a non-rigid 3D object and deducing the camera pose that best fits the projection. Section 4 shows a test suite that demonstrates the good performance of our method. Finally, Section 5 presents the main conclusions drawn from this work.

2 RELATED WORK

Amongst the majority of approaches that recover the 3D structure of a non-rigid object, the problem is divided into two steps: image registration and shape inference. As far as image registration is concerned, the issue can be tackled by using two main approaches: feature-based methods or direct methods. Feature-based methods [1] focus on detecting points (also called features) that are distinguishable from their neighbourhood, and direct methods [4], in turn, use the intensity differences between two images to calculate the correspondences.

The shape inference process adjusts the acquired visual information to the shape. Physics-based systems adjust the shape according to its physical properties. Others [3], however, integrate geometrical or temporal constraints to the system and solve it through optimization techniques such as Second Order Cone Programming (SOCP). Similarly, there are approaches that learn the principal deformations of the objects to tackle the problem.

Image registration as well as shape inference solutions are combined in several ways in many of the approaches presented below.

In terms of feature-based vision methods, we can find examples in [13] and [14]. These works perform the reconstruction of non-rigid surfaces by learning the deformation-modes of the shape. They use Principal Component Analysis (PCA) to compute modes of deformation and generate a database of feasible shapes [2]. In [14], the 3D deformation modes are performed by varying angles between facets, while in [13], the mesh is subdivided into a set of patches that are combined linearly. In contrast, there are solutions that do not obtain the deformation-modes of the shape but still solve the visual part by using a feature strategy [17]. In this sense, local rigidity constraints [18] or unconstrained quadratic optimization [22] surface reconstruction can be carried out. However, the primary focus of many of these approaches [14] is on planar surfaces. Moreover, most of them have a high computational cost.

Following the idea of using visual cues, there are real-time approaches like [12], even using RGB-D cameras [8] and [19], which

References

- Agudo, A., Calvo, B., and Montiel, J. "3d reconstruction of non-rigid surfaces in real-time using wedge elements". In *Computer Vision–ECCV, Workshops and Demonstrations*, pp. 113–122. 2012.
- Agudo, A., Calvo, B., and Montiel, J. "Finite element based sequential bayesian non-rigid structure from motion". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 1418–1425. 2012.
- Ahmed, N., Theobalt, C., Rossl, C., Thrun, S., and Seidel, H.-P. "Dense correspondence finding for parametrization-free animation reconstruction from video". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 1–8. 2008.
- Akenine-Möller, T. "Fast 3d triangle-box overlap testing". In *ACM SIGGRAPH Courses*, p. 8. 2005.
- Alahi, A., Ortiz, R., and Vandergheynst, P. "Freak: Fast retina keypoint". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 510–517. 2012.
- Aldoma, A., Marton, Z.-C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., Rusu, R., Gedikli, S., and Vincze, M. "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation". *Robotics Automation Magazine, IEEE*, Vol. 19, N. 3, pp. 80–91. Sept, 2012.
- Allen, B., Curless, B., and Popović, Z. "The space of human body shapes: reconstruction and parameterization from range scans". In *ACM Transactions on Graphics (TOG)*, volume 22, pp. 587–594. 2003.
- ALVAR. "Library for virtual and augmented reality". 2011.

- Álvarez, H., Aguinaga, I., and Borro, D. "Providing guidance for maintenance operations using automatic markerless augmented reality system". In *Mixed and Augmented Reality (ISMAR), 10th IEEE International Symposium on*, pp. 181–190. Basel, Switzerland. October, 2011.
- Álvarez, H. and Borro, D. "A novel approach to achieve robustness against marker occlusion". In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pp. 478–483. Lisboa, Portugal. February, 2009.
- Álvarez, H., Leizea, I., and Borro, D. "A new marker design for a robust marker tracking system against occlusions". *Computer Animation and Virtual Worlds*, Vol. 23, N. 5, pp. 503–518. 2012.
- Álvarez, H., Leizea, I., and Borro, D. "A survey on optical tracking for augmented reality". In *Proceedings of the XXII Conferencia Española de Computación Gráfica (CEIG)*, pp. 31–40. Jaén, Spain. September, 2012.
- Álvarez-Ponga, H. *Study of augmented reality methods for real time recognition and tracking of untextured 3d models in monocular images*. PhD thesis, Tecnun, University of Navarra. 2012.
- Andersen, M., Jensen, T., Lisouski, P., Mortensen, A., Hansen, M., Gregersen, T., and Ahrendt, P. "Kinect depth sensor evaluation for computer vision applications". Technical Report ECE-TR-6, AARHUS University. Department of Engineering. February, 2012.
- Armstrong, M. and Zisserman, A. "Robust object tracking". In *Asian Conference on Computer Vision (ACCV)*, volume I, pp. 58–61. Singapore. December, 1995.
- Arulampalam, M. S., Maskell, S., and Gordon, N. "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking". *IEEE Transactions on Signal Processing*, Vol. 50, pp. 174–188. 2002.
- Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking". *Signal Processing, IEEE Transactions on*, Vol. 50, N. 2, pp. 174–188. 2002.
- Azuma, R. T. et al. "A survey of augmented reality". *Presence*, Vol. 6, N. 4, pp. 355–385. 1997.
- Baker, S. "Real-time non-rigid driver head tracking for driver mental state estimation". 2004.

- Baker, S. and Matthews, I. "Lucas-kanade 20 years on: A unifying framework". *International Journal of Computer Vision*, Vol. 56, N. 3, pp. 221–255. feb, 2004.
- Barandiarán, J., Álvarez, H., and Borro, D. "Edge-based markerless 3d tracking of rigid objects". In *International Conference on Artificial Reality and Telexistence (ICAT)*, pp. 282–283. Esbjerg, Denmark. November, 2007.
- Bartoli, A. and Olsen, S. I. "A batch algorithm for implicit non-rigid shape and motion recovery". In *Dynamical Vision*, pp. 257–269. 2007.
- Bartoli, A. and Zisserman, A. "Direct estimation of non-rigid registrations". In *British Machine Vision Conference (BMVC)*, pp. 899–908. 2004.
- Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. "Speeded-up robust features (surf)". *Computer Vision and Image Understanding*, Vol. 110, pp. 346–359. 2008.
- Belongie, S., Malik, J., and Puzicha, J. "Shape matching and object recognition using shape contexts". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, N. 4, pp. 509–522. apr, 2002.
- Bergamasco, F., Albarelli, A., Rodola, E., and Torsello, A. "Rune-tag: A high accuracy fiducial marker with strong occlusion resilience". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 113–120. 2011.
- Bergamasco, F., Albarelli, A., and Torsello, A. "Pi-tag: a fast image-space marker design based on projective invariants". *Machine vision and applications*, Vol. 24, N. 6, pp. 1295–1310. 2013.
- Bhat, K. S., Twigg, C. D., Hodgins, J. K., Khosla, P. K., Zoran, P., and Seitz, S. M. "Estimating cloth simulation parameters from video". In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pp. 37–51. Aire-la-Ville, Switzerland. 2003.
- Bickel, B., Bäcker, M., Otaduy, M. A., Matusik, W., Pfister, H., and Gross, M. "Capture and modeling of non-linear heterogeneous soft tissue". *ACM Transactions on Graphics (TOG)*, Vol. 28, N. 3, p. 89. 2009.
- Bishop, G. and Welch, G. "An introduction to the kalman filter". *Proc of SIGGRAPH, Course*, Vol. 8, pp. 27599–3175. 2001.

- Blanz, V., Basso, C., Poggio, T., and Vetter, T. "Reanimating faces in images and video". In *Computer Graphics Forum*, volume 22, pp. 641–650. 2003.
- Blanz, V. and Vetter, T. "A morphable model for the synthesis of 3d faces". 1999.
- Bleser, G., Pastarmov, Y., and Stricker, D. "Real-time 3d camera tracking for industrial augmented reality applications". In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pp. 47–54. Plzen-Bory, Czech Republic. January, 2005.
- Bookstein, F. L. "Principal warps: Thin-plate splines and the decomposition of deformations". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, N. 6, pp. 567–585. 1989.
- Bouquet, J.-Y. "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm". 2001.
- Boyd, S. and Vandenberghe, L. *Convex optimization* (ISBN: 0521833787). Cambridge University Press, New York, NY, USA. 2004.
- Brand, M. "A direct method for 3d factorization of nonrigid motion observed in 2d". In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pp. 122–128. Washington, DC, USA. 2005.
- Bregler, C., Hertzmann, A., and Biermann, H. "Recovering non-rigid 3d shape from image streams". In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pp. 690–696. 2000.
- Brown, D. C. "Decentering Distortion of Lenses". *Photometric Engineering*, Vol. 32, N. 3, pp. 444–462. 1966.
- Brown, M. Z., Burschka, D., and Hager, G. D. "Advances in computational stereo". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 25, N. 8, pp. 993–1008. 2003.
- Brunet, F., Hartley, R., Bartoli, A., Navab, N., and Malgouyres, R. "Monocular template-based reconstruction of smooth and inextensible surfaces". In *Proceedings of the 10th Asian conference on Computer vision (AACV)*, volume 3, pp. 52–66. Berlin, Heidelberg. 2011.
- Cai, Q., Gallup, D., Zhang, C., and Zhang, Z. "3d deformable face tracking with a commodity depth camera". In *Computer Vision–ECCV*, pp. 229–242. Springer. 2010.

- Calonder, M., Lepetit, V., Strecha, C., and Fua, P. "Brief: Binary robust independent elementary features". In *Computer Vision–ECCV*, pp. 778–792. Springer. 2010.
- Caponio, A., Hincapié, M., and Mendivil, E. G. "lmar: highly parallel architecture for markerless augmented reality in aircraft maintenance". In *Virtual and Mixed Reality–New Trends*, pp. 20–29. Springer. 2011.
- Castillo, E. *Process optimization: A statistical approach*. International Series in Operations Research and Management Science. Springer. Berlin. 2007.
- Caudell, T. P. and Mizell, D. W. "Augmented reality: An application of heads-up display technology to manual manufacturing processes". In *System Sciences, Proceedings of the Twenty-Fifth Hawaii International Conference on*, volume 2, pp. 659–669. 1992.
- Cawood, S. and Fiala, M. *Augmented reality: A practical guide*. Pragmatic Bookshelf. 2008.
- Chen, L., Fu, H., Li, A., Ho, W., and Taj, C.-L. *Scalable maps of random dots for middle-scale locative mobile games*. IEEE. 2013.
- Cheng, S., Clarke, E. C., and Bilston, L. E. "Rheological properties of the tissues of the central nervous system: a review." *Medical engineering & physics*, Vol. 30, N. 10, pp. 1318–1337. December, 2008.
- Chui, H. and Rangarajan, A. "A new point matching algorithm for non-rigid registration". *Computer Vision and Image Understanding*, Vol. 89, N. 2-3, pp. 114–141. Feb, 2003.
- Chum, O. and Matas, J. "Matching with prosac–progressive sample consensus". In *Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Conference on*, volume 1, pp. 220–226. 2005.
- Clatz, O., Delingette, H., Talos, I.-f., Golby, A. J., Kikinis, R., Jolesz, F. A., Ayache, N., and Warfield, S. K. "Robust Non-Rigid Registration to Capture Brain Shift from Intra-Operative MRI". *IEEE Trans Med Imaging*, Vol. 24, N. 11, pp. 1417–1427. 2007.
- Cohen, L. D. and Cohen, I. "Finite element methods for active contour models and balloons for 2d and 3d images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, pp. 1131–1147. 1991.

- Cohen, L. D. and Cohen, I. "Deformable models for 3-d medical images using finite elements and balloons". In *Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Conference on*, pp. 592–598. 1992.
- Collins, T. and Bartoli, A. "Locally affine and planar deformable surface reconstruction from video". In *International Workshop on Vision, Modeling and Visualization*, pp. 339–346. 2010.
- Cootes, T. F., Edwards, G. J., and Taylor, C. J. "Active appearance models". In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 484–498. 1998.
- Costa, M. S. and Shapiro, L. G. "3d object recognition and pose with relational indexing". *Computer Vision and Image Understanding*, Vol. 79, pp. 364–407. 2000.
- Cotin, S., Delingette, H., and Ayache, N. "A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation". *The Visual Computer Journal*, Vol. 16, pp. 437–452. 2000.
- Cover, T. and Hart, P. "Nearest neighbor pattern classification". *Information Theory, IEEE Transactions on*, Vol. 13, N. 1, pp. 21–27. 1967.
- Davison, A. "Real-time simultaneous localisation and mapping with a single camera". In *Proceedings of 9th IEEE International Conference on Computer Vision (ICCV)*, pp. 1403–1410 vol.2. Nice, France. October, 2003.
- DeCarlo, D. and Metaxas, D. "Deformable model-based shape and motion analysis from images using motion residual error". In *Proceedings of the Sixth International Conference on Computer Vision (ICCV)*, p. 113. Washington, DC, USA. 1998.
- Decarlo, D. and Metaxas, D. "Optical flow constraints on deformable models with applications to face tracking". *International Journal of Computer Vision*, Vol. 38, N. 2, pp. 99–127. jul, 2000.
- Del Bue, A. and Agapito, L. "Non-rigid stereo factorization". *International Journal of Computer Vision*, Vol. 66, N. 2, pp. 193–207. 2006.
- Del Bue, A., Smeraldi, F., and Agapito, L. "Non-rigid structure from motion using ranklet-based tracking and non-linear optimization". *Image and Vision Computing*, Vol. 25, N. 3, pp. 297–310. 2007.

- Delingette, H., Hebert, M., and Ikeuchi, K. "Deformable surfaces: A free-form shape representation". In *Proc. SPIE Vol 1570: Geometric Methods in Computer Vision*, pp. 21–30. 1991.
- Donato, G. and Belongie, S. "Approximation methods for thin plate spline mappings and principal warps". *October*, Vol. 94063, pp. 1–13. 2002.
- Drost, B., Ulrich, M., Navab, N., and Ilic, S. "Model globally, match locally: Efficient and robust 3d object recognition". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 998–1005. San Francisco, USA. June, 2010.
- Drummond, T. and Cipolla, R. "Real-time visual tracking of complex structures". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, pp. 932–946. 2002.
- Ecker, A., Jepson, A. D., and Kutulakos, K. N. "Semidefinite programming heuristics for surface reconstruction ambiguities". In *Computer Vision–ECCV*, pp. 127–140. 2008.
- Faugeras, O. D., Luong, Q.-T., and Maybank, S. J. "Camera self-calibration: Theory and experiments". In *Computer Vision–ECCV*, pp. 321–334. 1992.
- Fayad, J., Agapito, L., and Del Bue, A. "Piecewise quadratic reconstruction of non-rigid surfaces from monocular sequences". In *Computer Vision–ECCV*, pp. 297–310. Springer. 2010.
- Feiner, S., Macintyre, B., and Seligmann, D. "Knowledge-based augmented reality". *Communications of the ACM*, Vol. 36, N. 7, pp. 53–62. 1993.
- Ferrant, M., Warfield, S. K., Nabavi, A., Jolesz, F. A., and Kikinis, R. "Registration of 3D IntraoperativeMR Images of the Brain Using a Finite Element Biomechanical Model". *Medical Imaging, IEEE Transactions on*, Vol. 20, N. 12, pp. 1384–97. 2001.
- Fiala, M. "Artag, a fiducial marker system using digital techniques". *Proceedings of the IEEE Computer Society Conference on Computer Vision Pattern Recognition (CVPR)*, pp. 590–596. 2005.
- Fiala, M. "Comparing artag and artoolkit plus fiducial marker systems". In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications (HAVE)*. 2005.

- Figl, M., Rueckert, D., Hawkes, D., Casula, R., Hu, M., Pedro, O., Zhang, D. P., Penney, G., Bello, F., and Edwards, P. "Image guidance for robotic minimally invasive coronary artery bypass". *Computerized Medical Imaging and Graphics*, Vol. 34, N. 1, pp. 61–68. 2010.
- Fischler, M. A. and Bolles, R. C. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*, Vol. 24, N. 6, pp. 381–395. 1981.
- Flaquer, J., Olaizola, J., and Olaizola, J. *Curso de álgebra lineal*. Eunsa. 2004.
- Fleet, D. J., Black, M. J., Yacoob, Y., and Jepson, A. D. "Design and use of linear models for image motion analysis". *International Journal of Computer Vision*, Vol. 36, N. 3, pp. 171–193. 2000.
- Franken, T., Dellepiane, M., Ganovelli, F., Cignoni, P., Montani, C., and Scopigno, R. "Minimizing user intervention in registering 2d images to 3d models". *The Visual Computer*, Vol. 21, N. 8–10, pp. 619–628. 2005.
- Friedman, J. H., Bentley, J. L., and Finkel, R. A. "An algorithm for finding best matches in logarithmic expected time". *ACM Transactions on Mathematical Software (TOMS)*, Vol. 3, N. 3, pp. 209–226. 1977.
- Fua, P. and Leclerc, Y. G. "Object-centered surface reconstruction: Combining multi-image stereo and shading". *International Journal of Computer Vision*, Vol. 16, pp. 35–56. 1995.
- Fugl, A. R., Jordt, A., Petersen, H. G., Willatzen, M., and Koch, R. "Simultaneous estimation of material properties and pose for deformable objects from depth and color images". In *Pattern Recognition*, pp. 165–174. Springer. 2012.
- Gall, J., Rosenhahn, B., and Seidel, H. P. "Robust pose estimation with 3d textured models". In *Pacific-Rim Symposium on Image and Video Technology (PSIVT)*, pp. 84–95. 2006.
- Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. "Automatic generation and detection of highly reliable fiducial markers under occlusion". *Pattern Recognition*, Vol. 47, N. 6, pp. 2280–2292. 2014.
- Gauglitz, S., Höllerer, T., and Turk, M. "Evaluation of interest point detectors and feature descriptors for visual tracking". *International Journal of Computer Vision*, Vol. 94, pp. 335–360. 2011.

- Gay-Bellile, V., Bartoli, A., and Sayd, P. "Direct estimation of non-rigid registrations with image-based self-occlusion reasoning". In *Computer Vision (ICCV), IEEE 11th International Conference*, p. 1. oct., 2007.
- Georgel, P. F., Benhimane, S., and Navab, N. "A unified approach combining photometric and geometric information for pose estimation." In *BMVC*, pp. 1–10. 2008.
- Gladilin, E., Zachow, S., Deuffhard, P., and Hege, H.-C. "A biomechanical model for soft tissue simulation in craniofacial surgery". *Proceedings International Workshop on Medical Imaging and Augmented Reality*, Vol. i, pp. 137–141. 2001.
- Gumerov, N., Zandifar, A., Duraiswami, R., and Davis, L. S. "Structure of applicable surfaces from single views". In *Computer Vision-ECCV*, pp. 482–496. Springer. 2004.
- Haehnel, D., Thrun, S., and Burgard, W. "An extension of the icp algorithm for modeling nonrigid objects with mobile robots". In *IJCAI*, pp. 915–920. 2003.
- Hakkarainen, M., Woodward, C., and Billingham, M. "Augmented assembly using a mobile phone". In *Mixed and Augmented Reality (ISMAR), 7th IEEE/ACM International Symposium on*, pp. 167–168. 2008.
- Hamarneh, G., Amir-Khalili, A., Nosrati, M. S., Figueroa, I., Kawahara, J., Al-Alao, O., Peyrat, J.-M., Abi-Nahed, J., Al-Ansari, A., and Abugharbieh, R. "Towards multi-modal image-guided tumour identification in robot-assisted partial nephrectomy". In *Biomedical Engineering (MECBME), Middle East Conference on*, pp. 159–162. February, 2014.
- Haouchine, N., Dequidt, J., Berger, M.-O., and Cotin, S. "Single view augmentation of 3d elastic objects". In *Mixed and Augmented Reality (ISMAR), IEEE International Symposium on*, pp. 229–236. 2014.
- Haouchine, N., Dequidt, J., Berger, M.-O., Cotin, S., et al. "Deformation-based augmented reality for hepatic surgery". In *Medicine Meets Virtual Reality (MMVR)*. 2013.
- Haouchine, N., Dequidt, J., Kerrien, E., Berger, M.-O., Cotin, S., et al. "Physics-based augmented reality for 3d deformable object". In *VRIPHYS-Virtual Reality Interaction and Physical Simulation*. 2012.

- Haouchine, N., Dequidt, J., Peterlik, I., Kerrien, E., Berger, M.-O., and Cotin, S. "Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery". In *Mixed and Augmented Reality (ISMAR), IEEE International Symposium on*, pp. 199–208. 2013.
- Haouchine, N., Dequidt, J., Peterlik, I., Kerrien, E., Berger, M.-O., Cotin, S., et al. "Towards an accurate tracking of liver tumors for augmented reality in robotic assisted surgery". In *International Conference on Robotics and Automation (ICRA)*. 2014.
- Hartley, R. and Zisserman, A. *Multiple view geometry in computer vision*. Cambridge university press. 2003.
- Hartley, R. I. "Euclidean reconstruction from uncalibrated views". In *Proceedings of the Second Joint European - US Workshop on Applications of Invariance in Computer Vision*, pp. 237–256. Ponta Delgada, Azores, Portugal. October, 1994.
- Hartley, R. I. "Kruppa's equations derived from the fundamental matrix". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, N. 2, pp. 133–135. 1997.
- Hawkins, D. M. *Identification of outliers*, volume 11. Springer. 1980.
- Hayashi, T., de Sorbier, F., and Saito, H. "Texture overlay onto non-rigid surface using commodity depth camera." *VISAPP (2)*, pp. 66–71. 2012.
- Hemayed, E. "A survey of camera self-calibration". In *Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 351 – 357. Miami, FL, USA. July, 2003.
- Henderson, S. and Feiner, S. "Exploring the benefits of augmented reality documentation for maintenance and repair". *Visualization and Computer Graphics, IEEE Transactions on*, Vol. 17, N. 10, pp. 1355–1368. 2011.
- Henderson, S. J. and Feiner, S. "Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret". In *Mixed and Augmented Reality (ISMAR), 8th IEEE International Symposium on*, pp. 135–144. 2009.
- Henderson, S. J. and Feiner, S. K. "Augmented reality in the psychomotor phase of a procedural task". In *Mixed and Augmented Reality (ISMAR), 10th IEEE International Symposium on*, pp. 191–200. 2011.

- Hilsmann, A. and Eisert, P. "Tracking and retexturing cloth for real-time virtual clothing applications". In *Proceedings of the 4th International Conference on Computer Vision/Computer Graphics Collaboration Techniques*, MIRAGE, pp. 94–105. Berlin, Heidelberg. 2009.
- Hilsmann, A., Schneider, D. C., and Eisert, P. "Realistic cloth augmentation in single view video under occlusions". *Computers & Graphics*, Vol. 34, N. 5, pp. 567–574. 2010.
- Hinterstoisser, S., Benhimane, S., and Navab, N. "N3m: Natural 3d markers for real-time object detection and pose estimation". In *IEEE 11th International Conference on Computer Vision (ICCV)*, pp. 1–7. Rio de Janeiro, Brazil. October, 2007.
- Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., and Lepetit, V. "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes". In *Computer Vision (ICCV), IEEE International Conference on*, pp. 858–865. Nov, 2011.
- Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., and Navab, N. "Dominant orientation templates for real-time detection of texture-less objects". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2257–2264. San Francisco, USA. June, 2010.
- Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes". In *Computer Vision—ACCV*, pp. 548–562. Springer. 2013.
- Holzer, S., Hinterstoisser, S., Ilic, S., and Navab, N. "Distance transform templates for object detection and pose estimation". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1177–1184. Miami, FL, USA. June, 2009.
- Hong, L., C. S. K. and Chen, J. "Deforming pages of 3d electronic books". 2004.
- Horgan, C. O. and Murphy, J. G. "Simple shearing of soft biological tissues". *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 467, N. 2127, pp. 760–777. September, 2010.
- Indyk, P. and Motwani, R. "Approximate nearest neighbors: towards removing the curse of dimensionality". In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613. 1998.

- Irving, G., Teran, J., and Fedkiw, R. "Tetrahedral and hexahedral invertible finite elements". *Graphical Models*, Vol. 68, N. 2, pp. 66–89. 2004.
- Irving, G., Teran, J., and Fedkiw, R. "Invertible finite elements for robust simulation of large deformation". In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA, pp. 131–140. Aire-la-Ville, Switzerland, Switzerland. 2004.
- Johnson, H. J. and Christensen, G. E. "Consistent landmark and intensity-based image registration". *Medical Imaging, IEEE Transactions on*, Vol. 21, N. 5, pp. 450–461. 2002.
- Jolliffe, I. T. *Principal component analysis*. Springer Verlag. 1986.
- Jordt, A. and Koch, R. "Fast tracking of deformable objects in depth and colour video." In *BMVC*, pp. 1–11. 2011.
- Jordt, A. and Koch, R. "Direct model-based tracking of 3d object deformations in depth and color video". *International Journal of Computer Vision*, pp. 1–17. 2013.
- Jordt, A. and Koch, R. "Reconstruction of deformation from depth and color video with explicit noise models". *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, pp. 128–146. 2013.
- Kähkönen, K., Hyväkkä, J., Porkka, J., Siltanen, S., and Woodward, C. "Integrating building product models with live video stream". 2007.
- Kahl, F. "Multiple view geometry and the l-infinity norm". In *IEEE International Conference on Computer Vision (ICCV)*. Beijing. October, 2005.
- Karlsson, N., Di Bernardo, E., Ostrowski, J., Goncalves, L., Pirjanian, P., and Munich, M. E. "The vslam algorithm for robust localization and mapping". In *Robotics and Automation (ICRA), Proceedings of the 2005 IEEE International Conference on*, pp. 24–29. 2005.
- Kass, M., Witkin, A., and Terzopoulos, D. "Snakes: Active contour models". *International Journal of Computer Vision*, Vol. 1, N. 4, pp. 321–331. 1988.
- Kato, H. and Billinghurst, M. "Marker tracking and hmd calibration for a video-based augmented reality conferencing system". In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR)*, pp. 85–94. Washington, DC, USA. October, 1999.

- Ke, Q. and Kanade, T. "Quasiconvex optimization for robust geometric reconstruction". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, N. 10, pp. 1834–1847. oct, 2007.
- Kerdok, A. E. *Characterizing the Nonlinear Mechanical Response of Liver to Surgical Manipulation*. PhD thesis, Harvard University. 2006.
- Khoshelham, K. "Accuracy analysis of kinect depth data". In *ISPRS workshop laser scanning*, volume 38, p. W12. 2011.
- Khoshelham, K. and Elberink, S. O. "Accuracy and resolution of kinect depth data for indoor mapping applications". *Sensors*, Vol. 12, N. 2, pp. 1437–1454. 2012.
- Kim, J.-H., Bartoli, A., Collins, T., and Hartley, R. "Tracking by detection for interactive image augmentation in laparoscopy". In *Biomedical Image Registration*, pp. 246–255. Springer. 2012.
- Klein, G. and Murray, D. "Parallel tracking and mapping for small ar workspaces". In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 1–10. Nara, Japan. November, 2007.
- Koch, R. "Dynamic 3-d scene analysis through synthesis feedback control". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 15, N. 6, pp. 556–568. 1993.
- Lamdan, Y. and Wolfson, H. "Geometric hashing: A general and efficient model-based recognition scheme". In *Second International Conference on Computer Vision (CV)*, pp. 238–249. Tampa, USA. December, 1988.
- Landeira, M. A. *Development of a General Purpose for Performance of Surgical Procedures*. PhD thesis, Tecnun, University of Navarra. 2014.
- Landeira, M. A., Sánchez, E., Tejada, S., and Díez, R. "Desarrollo e implementación de una estrategia de gestión de singularidades para un sistema robótico redundante cooperativo destinado a la asistencia en intervenciones quirúrgicas". *Revista Iberoamericana de Automatica e Informatica industrial*, Vol. 00, pp. 1–12. 2014.
- LaViola, J. J. "Double exponential smoothing: an alternative to kalman filter-based predictive tracking". In *Proceedings of the Workshop on Virtual Environments (EGVE)*, pp. 199–206. Zurich, Switzerland. May, 2003.

- Lee, G., Billingham, M., and Kim, G. "Occlusion based interaction methods for tangible augmented reality environments". In *Proceedings of ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry (VRCAI)*, pp. 419–426. 2004.
- Leizea, I., Álvarez, H., Aguinaga, I., and Borro, D. "Real-time deformation, registration and tracking of solids based on physical simulation". In *Proceedings of the 13th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 165–170. Munich, Bavaria, Germany. September, 2014.
- Leizea, I., Álvarez, H., and Borro, D. "Real time non-rigid 3d surface tracking using particle filter". *Computer Vision and Image Understanding*, Vol. 133, N. 0, pp. 51–65. April, 2105.
- Leizea, I., Mendizabal, A., Álvarez, H., Aguinaga, I., Sánchez, E., and Borro, D. "Tracking of deformable objects based on a visual approach and physics simulation for robot-assisted surgery". *Submitted to Computer Graphics and Applications, IEEE*, 2015.
- Lepetit, V. and Fua, P. "Monocular model-based 3d tracking of rigid objects: a survey". *Foundations and trends in computer graphics and vision*, Vol. 1, pp. 1–89. 2005.
- Lepetit, V. and Fua, P. "Keypoint recognition using randomized trees". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, N. 9, pp. 1465–1479. sep, 2006.
- Lepetit, V., Moreno-Noguer, F., and Fua, P. "Epnnp: An accurate o(n) solution to the pnp problem". *International Journal of Computer Vision*, Vol. 81, pp. 155–166. 2009.
- Leutenegger, S., Chli, M., and Siegwart, R. Y. "Brisk: Binary robust invariant scalable keypoints". In *Computer Vision (ICCV), IEEE International Conference on*, pp. 2548–2555. 2011.
- Liang, J., DeMenthon, D., and Doermann, D. "Flattening curved documents in images". In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pp. 338–345. Washington, DC, USA. 2005.
- Liesaputra, V. "Turning the page of an electronic book". In *5th NZ Computer Science Research Student Conference (NZCSRSC)*, pp. 23–30. 2007.

- LM., L. "Tools and toys for teaching design of experiments methodology". In *33rd Annual General Conference of the Canadian Society for Civil Engineering. June, Toronto, Ontario, Canada.* 2005.
- Longuet-Higgins, H. C. "A computer algorithm for reconstructing a scene from two projections". *Nature*, Vol. 293, pp. 133–135. 1981.
- Lowe, D. G. "Object recognition from local scale-invariant features". In *Computer Vision, The proceedings of the seventh IEEE international conference on*, volume 2, pp. 1150–1157. 1999.
- Lowe, D. G. "Distinctive image features from scale-invariant keypoints". *International Journal of Computer Vision*, Vol. 60, N. 2, pp. 91–110. 2004.
- Lucas, B. D. and Kanade, T. "An iterative image registration technique with an application to stereo vision". In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 674–679. 1981.
- Lv, Q., Josephson, W., Wang, Z., Charikar, M., and Li, K. "Multi-probe lsh: efficient indexing for high-dimensional similarity search". In *Proceedings of the 33rd international conference on Very large data bases*, pp. 950–961. 2007.
- Madsen, K., Nielsen, H. B., and Tingleff, O. "Methods for non-linear least squares problems". 1999.
- Maidi, M., Ababsa, F., and Mallem, M. "Handling occlusions for robust augmented reality systems". *Journal on Image and Video Processing*, Vol. 2010, pp. 1–12. January, 2010.
- Malik, S., Roth, G., and McDonald, C. "Robust 2d tracking for real-time augmented reality". In *Proceedings of International Conference on Vision Interface (VI)*, pp. 399–406. Calgary, Canada. May, 2002.
- Marimon, D., Maret, Y., Abdeljaoued, Y., and Ebrahimi, T. "Particle filter-based camera tracker fusing marker- and feature point-based cues". In *Proc. of the IS&T/SPIE Electronic Imaging Conf. on Visual Communications and Image Processing*, number LTS-CONF-2006-060. 2007.
- Matthews, I. and Baker, S. "Active appearance models revisited". *International Journal of Computer Vision*, Vol. 60, N. 2, pp. 135–164. nov, 2004.
- McDonald, C. and Roth, G. "Replacing a mouse with hand gesture in a plane-based augmented reality system". In *Proceedings of Vision Interface (VI)*. 2003.

- McInerney, T. and Terzopoulos, D. "A finite element model for 3d shape reconstruction and nonrigid motion tracking". In *Computer Vision, Proceedings, Fourth International Conference on*, pp. 518–523. 1993.
- McInerney, T. and Terzopoulos, D. "A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4d image analysis". *Computerized Medical Imaging and Graphics*, Vol. 19, pp. 69–83. 1995.
- Meden, B., Knödel, S., and Bourgeois, S. "Markerless augmented reality solution for industrial manufacturing". In *Proceedings of the 13th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Munich, Germany. September, 2014.
- Metaxas, D. and Terzopoulos, D. "Constrained deformable superquadrics and nonrigid motion tracking". In *Computer Vision and Pattern Recognition (CVPR), Proceedings IEEE Computer Society Conference on*, pp. 337–343. 1991.
- Mikolajczyk, K. and Schmid, C. "Scale & affine invariant interest point detectors". *International Journal of Computer Vision*, Vol. 60, pp. 63–86. 2004.
- Mikolajczyk, K. and Schmid, C. "A performance evaluation of local descriptors". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, pp. 1615–1630. 2005.
- Milgram, P. and Kishino, F. "A taxonomy of mixed reality visual displays". *IEICE TRANSACTIONS on Information and Systems*, Vol. 77, N. 12, pp. 1321–1329. 1994.
- Miller, K., Wittek, A., and Joldes, G. "Biomechanics of the brain for computer-integrated surgery". *Acta of Bioengineering and Biomechanics*, Vol. 12, N. 2, 2010.
- Misra, S., Ramesh, K., and Okamura, A. M. "Modelling of non-linear elastic tissues for surgical simulation". *Computer methods in biomechanics and biomedical engineering*, Vol. 13, N. 6, pp. 811–818. 2010.
- Moreno-Noguer, F. and Fua, P. "Stochastic exploration of ambiguities for non-rigid shape recovery". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 99, N. PrePrints, 2012.

- Moreno-Noguer, F. and Porta, J. M. "Probabilistic simultaneous pose and non-rigid shape recovery". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 1289–1296. 2011.
- Moreno-Noguer, F., Porta, J. M., and Fua, P. "Exploring ambiguities for monocular non-rigid shape estimation". In *Proceedings of the 11th European conference on computer vision conference on Computer vision: Part III, ECCV*, pp. 370–383. Berlin, Heidelberg. 2010.
- Moreno-Noguer, F., Salzmann, M., Lepetit, V., and Fua, P. "Capturing 3d stretchable surfaces from single images in closed form". In *Computer Vision and Pattern Recognition*, pp. 1842–1849. 2009.
- Munoz, E., Buenaposada, J. M., and Baumela, L. "A direct approach for efficiently tracking with 3d morphable models". In *Computer Vision, IEEE 12th International Conference on*, pp. 1615–1622. 2009.
- Murphy-Chutorian, E. and Trivedi, M. M. "Head pose estimation in computer vision: A survey". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 31, N. 4, pp. 607–626. 2009.
- Naimark, L. and Foxlin, E. "Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker". In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality (ISMAR)*, p. 27. 2002.
- Nakai, T., Kise, K., and Iwamura, M. "Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval". In *Lecture Notes in Computer Science (7th International Workshop)*, pp. 541–552. 2006.
- Nastar, C. and Ayache, N. "Frequency-based nonrigid motion analysis: application to four dimensional medical images". *Pattern Analysis and Machine Intelligence, IEEE Transactions*, Vol. 18, N. 11, p. 1067. nov, 1996.
- Neubert, J., Pretlove, J., and Drummond, T. "Rapidly constructed appearance models for tracking in augmented reality applications". *Machine Vision and Applications*, Vol. 23, N. 5, pp. 843–856. 2012.
- Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. "Dtam: Dense tracking and mapping in real-time". In *Computer Vision (ICCV), IEEE International Conference on*, pp. 2320–2327. 2011.

- Nicolau, S., Soler, L., Mutter, D., and Marescaux, J. "Augmented reality in laparoscopic surgical oncology". *Surgical oncology*, Vol. 20, N. 3, pp. 189–201. 2011.
- Ohbuchi, R., Bajura, M., and Fuchs, H. "Case study: Observing a volume rendered fetus within a pregnant patient". In *IEEE Conference on Visualization*, volume 5, p. 364. 1998.
- Olsen, S. I. and Bartoli, A. "Implicit non-rigid structure-from-motion with priors". *Journal of Mathematical Imaging and Vision*, Vol. 31, N. 2–3, pp. 233–244. 2008.
- Pentland, A. and Sclaroff, S. "Closed-form solutions for physically based shape modeling and recognition". *Pattern Analysis and Machine Intelligence, IEEE Transactions*, Vol. 13, N. 7, p. 715. jul, 1991.
- Perriollat, M., Hartley, R., and Bartoli, A. "Monocular template-based reconstruction of inextensible surfaces". *International Journal of Computer Vision*, Vol. 95, N. 2, pp. 124–137. November, 2011.
- Petersen, N. and Stricker, D. "Learning task structure from video examples for workflow tracking and authoring". In *Mixed and Augmented Reality (ISMAR), IEEE International Symposium on*, pp. 237–246. 2012.
- Picinbono, G., Delingette, H., and Ayache, N. "Real-time large displacement elasticity for surgery simulation: Non-linear tensor-mass model". In *Proceedings of the Third International Conference on Medical Robotics, Imaging and Computer Assisted Surgery (MICCAI)*, pp. 643–652. 2000.
- Pilet, J., Lepetit, V., and Fua, P. "Fast non-rigid surface detection, registration and realistic augmentation". *International Journal of Computer Vision*, Vol. 76, N. 2, pp. 109–122. feb, 2008.
- Pilet, J., Lepetit, V., and Pascal.Fua. "Real-time non-rigid surface detection". In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pp. 822–828. Washington, DC, USA. 2005.
- Pilet, J. and Saito, H. "Virtually augmenting hundreds of real pictures: An approach based on learning, retrieval, and tracking". In *Virtual Reality Conference (VR), IEEE*, pp. 71–78. 2010.

- Pizarro, D. and Bartoli, A. "Feature-based deformable surface detection with self-occlusion reasoning". *International Journal of Computer Vision*, Vol. 97, N. 1, pp. 54–70. Mar, 2012.
- Platonov, J., Heibel, H., Meier, P., and Grollmann, B. "A mobile markerless ar system for maintenance and repair". In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 105–108. Santa Barbara, USA. October, 2006.
- Platt, J. *Fast training of support vector machines using sequential minimal optimization*. MIT Press. 1998.
- Pratt, P., Stoyanov, D., Visentini-Scarzanella, M., and Yang, G.-Z. "Dynamic guidance for robotic surgery using image-constrained biomechanical models". In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 77–85. Springer. 2010.
- Pupilli, M. and Calway, A. "Real-time camera tracking using a particle filter". In *British Machine Vision Conference (BMVC)*, pp. 519–528. Oxford, U.K. September, 2005.
- Rabaud, V. and Belongie, S. "Linear embeddings in non-rigid structure from motion". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 2427–2434. 2009.
- Rao, Q., Tropper, T., Grünler, C., Hammori, M., and Chakraborty, S. "Ar-ivi: Implementation of in-vehicle augmented reality". In *Proceedings of the 13th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Munich, Germany. September, 2014.
- Rolland, J. P., Davis, L., and Baillet, Y. "A survey of tracking technology for virtual environments". *Fundamentals of wearable computers and augmented reality*, Vol. 1, pp. 67–112. 2001.
- Rosenhahn, B., Kersting, U., Powell, K., Klette, R., Klette, G., and Seidel, H.-P. "A system for articulated tracking incorporating a clothing model". *Machine Vision and Applications*, Vol. 18, N. 1, pp. 25–40. 2007.
- Rosten, E. and Drummond, T. "Machine learning for high-speed corner detection". In *Computer Vision—ECCV 2006*, pp. 430–443. Springer. 2006.

- Rosten, E., Porter, R., and Drummond, T. "Faster and better: A machine learning approach to corner detection". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 32, N. 1, pp. 105–119. 2010.
- Rothganger, F., Lazebnik, S., Schmid, C., and Ponce, J. "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints". *International Journal of Computer Vision*, Vol. 66, N. 3, pp. 231–259. 2006.
- Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. "Orb: an efficient alternative to sift or surf". In *Computer Vision (ICCV), IEEE International Conference on*, pp. 2564–2571. 2011.
- Russell, C., Fayad, J., and Agapito, L. "Energy based multiple model fitting for non-rigid structure from motion". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 3009–3016. 2011.
- Sääski, J., Salonen, T., Hakkarainen, M., Siltanen, S., Woodward, C., and Lempiäinen, J. "Integration of design and assembly using augmented reality". In *Micro-Assembly Technologies and Applications*, pp. 395–404. Springer. 2008.
- Salih, Y. and Malik, A. S. "Comparison of stochastic filtering methods for 3d tracking". *Pattern Recognition*, Vol. 44, pp. 2711–2737. 2011.
- Salonen, T., Sääski, J., Hakkarainen, M., Kannetis, T., Perakakis, M., Siltanen, S., Potamianos, A., Korkalo, O., and Woodward, C. "Demonstration of assembly work using augmented reality". In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pp. 120–123. 2007.
- Salonen, T., Sääski, J., Woodward, C., Korkalo, O., Marstio, I., and Rainio, K. "Data pipeline from cad to ar based assembly instructions". In *ASME-AFM World Conference on Innovative Virtual Reality*, pp. 165–168. 2009.
- Salzmann, M. and Fua, P. "Reconstructing sharply folding surfaces: A convex formulation". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 1054–1061. 2009.
- Salzmann, M. and Fua, P. "Linear local models for monocular reconstruction of deformable surfaces". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 33, N. 5, pp. 931–944. 2011.

- Salzmann, M., Hartley, R., and Fua, P. "Convex optimization for deformable surface 3-d tracking". In *Computer Vision (ICCV), IEEE 11th International Conference on*, pp. 1–8. 2007.
- Salzmann, M., Ilic, S., and Fua, P. "Parameterizing deformable surfaces for monocular 3-d tracking". 2005.
- Salzmann, M., Ilic, S., and Fua, P. "Physically valid shape parameterization for monocular 3-d deformable surface tracking". In *British Machine Vision Conference (BMVC)*. 2005.
- Salzmann, M., Lepetit, V., and Fua, P. "Deformable surface tracking ambiguities". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 1–8. 2007.
- Salzmann, M., Moreno-Noguer, F., Lepetit, V., and Fua, P. "Closed-form solution to non-rigid 3d surface registration". 2008.
- Salzmann, M., Pilet, J., Ilic, S., and Fua, P. "Surface deformation models for nonrigid 3d shape recovery". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 29, N. 8, pp. 1481–1487. 2007.
- Salzmann, M., Urtasun, R., and Fua, P. "Local deformation models for monocular 3d shape recovery". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 1–8. 2008.
- San-Vicente Otamendi, G., Celigüeta Lizarza, J. T., and Aguinaga Hoyos, I. "Designing deformable models of soft tissue for virtual surgery planning and simulation using the mass-spring model". 2011.
- Sánchez, J. R., Álvarez, H., and Borro, D. *Gft: Gpu fast triangulation of 3d points* (ISBN: 3-642-15909-5), volume 6374 of *Lecture Notes in Computer Science, Computer Vision and Graphics*, pp. 235–242. Springer-Verlag Berlin Heidelberg. 2010.
- Sánchez, J. R., Álvarez, H., and Borro, D. "Towards real time 3d tracking and reconstruction on a gpu using monte carlo simulations". In *International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 185–192. Seoul, Korea. October, 2010.
- Sánchez-Riera, J., Ostlund, J., Fua, P., and Moreno-Noguer, F. "Simultaneous pose, correspondence and non-rigid shape". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 1189–1196. 2010.

- Savioja, P., Järvinen, P., Karhela, T., Siltanen, P., and Woodward, C. "Developing a mobile, service-based augmented reality tool for modern maintenance work". In *Virtual Reality*, pp. 554–563. Springer. 2007.
- Schattel, D., Tönnis, M., Klinker, G., Schubert, G., and Petzold, F. "On-site augmented collaborative architecture visualization". In *Proceedings of the 13th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Munich, Germany. September, 2014.
- Schmalstieg, D. and Wagner, D. "Experiences with handheld augmented reality". In *Mixed and Augmented Reality (ISMAR), 6th IEEE and ACM International Symposium on*, pp. 3–18. Nara, Japan. November, 2007.
- Schulman, J., Lee, A., Ho, J., and Abbeel, P. "Tracking deformable objects with point clouds". In *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 1130–1137. 2013.
- Schwald, B., Figue, J., Chauvineau, E., Vu-hong, F., Raully, F. D. D. E., Anez, F. G., Baldo, O., and Santos, J. M. "Starmate : Using augmented reality technology for computer guided maintenance of complex mechanical elements". In *eBusiness and eWork Conference. Venice*. 2001.
- Sclaroff, S. and Isidoro, J. "Active blobs: region-based, deformable appearance models". *Computer Vision and Image Understanding*, Vol. 89, N. 2–3, pp. 197–225. feb, 2003.
- Sehgal, A. "3D object recognition using bayesian geometric hashing and pose clustering". *Pattern Recognition*, Vol. 36, N. 3, pp. 765–780. 2003.
- Shaji, A., Varol, A., Torresani, L., and Fua, P. "Simultaneous point matching and 3d deformable surface reconstruction". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference*, p. 1221. june, 2010.
- Sharf, A., Lewiner, T., Shamir, A., Kobbelt, L., and Cohen-or, D. "Competing fronts for coarse-to-fine surface reconstruction". In *Computer Graphics Forum*, pp. 389–398. 2006.
- Shen, S., Shi, W., and Liu, Y. "Monocular 3-d tracking of inextensible deformable surfaces under l2-norm". *Image Processing, IEEE Transactions on*, Vol. 19, N. 2, pp. 512–521. 2010.

- Shen, S., Zheng, Y., and Liu, Y. "Deformable surface stereo tracking-by-detection using second order cone programming." In *Pattern Recognition (ICPR), 19th International Conference on*, pp. 1–4. 2008.
- Shimizu, N., Yoshida, T., Hayashi, T., de Sorbier, F., and Saito, H. "Non-rigid surface tracking for virtual fitting system". In *VISAPP (2)*, pp. 12–18. 2013.
- Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. "Real-time human pose recognition in parts from single depth images". *Communications of the ACM*, Vol. 56, N. 1, pp. 116–124. 2013.
- Siltanen, S., Hakkarainen, M., Korkalo, O., Salonen, T., Saaski, J., Woodward, C., Kannetis, T., Perakakis, M., and Potamianos, A. "Multimodal user interface for augmented assembly". In *Multimedia Signal Processing (MMSP), IEEE 9th Workshop on*, pp. 78–81. 2007.
- Siltanen, S., Saraspää, H., and Karvonen, J. "A complete interior design solution with diminished reality". In *Proceedings of the 13th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Munich, Germany. September, 2014.
- Silverman, B. W. and Jones, M. C. "E. fix and jl hodges (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on fix and hodges (1951)". *International Statistical Review/Revue Internationale de Statistique*, pp. 233–238. 1989.
- Simon, G. and Berger, M.-O. "Reconstructing while registering: a novel approach for markerless augmented reality". In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 285–293. Darmstadt, Germany. October, 2002.
- Skrinjar, O., Nabavi, A., and Duncan, J. "Model-driven brain shift compensation". *Medical image analysis*, Vol. 6, N. 4, pp. 361–73. 2002.
- Smisek, J., Jancosek, M., and Pajdla, T. "3d with kinect". In *Consumer Depth Cameras for Computer Vision*, pp. 3–25. Springer. 2013.
- Speidel, S., Roehl, S., Suwelack, S., Dillmann, R., Kenngott, H., and Mueller-Stich, B. "Intraoperative surface reconstruction and biomechanical modeling for soft tissue registration". In *Proc. Joint Workshop on New Technologies for Computer/Robot Assisted Surgery*. 2011.

- Stark, M., Goesele, M., and Schiele, B. "Back to the future: Learning shape models from 3d cad data". In *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 106.1–106.11. Aberystwyth, UK. August, 2010.
- Sutherland, I. E. "A head-mounted three dimensional display". In *Proceedings of the December 9-11, fall joint computer conference, part I*, pp. 757–764. 1968.
- Suzuki, A., Manabe, Y., and Yata, N. "Design of an ar marker for cylindrical surface". In *Mixed and Augmented Reality (ISMAR), IEEE International Symposium on*, pp. 293–294. 2013.
- Szentandrasei, I., Zacharias, M., Havel, J., Herout, A., Dubska, M., and Kajan, R. "Uniform marker fields: Camera localization by orientable de bruijn tori". In *Mixed and Augmented Reality (ISMAR), IEEE International Symposium on*, pp. 319–320. 2012.
- Tanco, M., Viles, E., Ilzarbe, L., and Alvarez, M. J. "Implementation of design of experiments projects in industry". *Appl. Stoch. Model. Bus. Ind.*, Vol. 25, N. 4, pp. 478–505. July, 2009.
- Tateno, K. "A nested marker for augmented reality". In *IEEE Virtual Reality Conference (VR)*, pp. 259–262. 2007.
- Taylor, J., Jepson, A. D., and Kutulakos, K. N. "Non-rigid structure from locally-rigid motion". In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 2761–2768. 2010.
- Teichrieb, V., Lima, M., Lourenc, E., Bueno, S., Kelner, J., and Santos, I. H. F. "A survey of online monocular markerless augmented reality". *International Journal of Modeling and Simulation for the Petroleum Industry*, Vol. 1, N. 1, pp. 1–7. 2007.
- Terzopoulos, D. and Metaxas, D. "Dynamic 3d models with local and global deformations: Deformable superquadrics". *IEEE Trans.Pattern Anal.Mach.Intell.*, Vol. 13, N. 7, pp. 703–714. jul, 1991.
- Terzopoulos, D., Witkin, A., and Kass, M. "Symmetry-seeking models and 3d object reconstruction". *International Journal of Computer Vision*, Vol. 1, pp. 211–221. 1987.

- Titsias, M. K. and Lawrence, N. D. "Gaussian process latent variable models for visualisation of high dimensional data". In *Adv. in Neural Inf. Proc. Sys.* 2004.
- Tomasi, C. and Kanade, T. "Shape and motion from image streams under orthography: a factorization method". *International Journal of Computer Vision*, Vol. 9, N. 2, pp. 137–154. 1992.
- Torr, P. H. and Zisserman, A. "Feature based methods for structure and motion estimation". In *Vision Algorithms: Theory and Practice*, pp. 278–294. Springer. 2000.
- Torresani, L., Hertzmann, A., and Bregler, C. "Learning non-rigid 3d shape from 2d motion." In *NIPS*, volume 1, p. 2. 2003.
- Torresani, L., Yang, D. B., Alexander, E. J., and Bregler, C. "Tracking and modeling non-rigid objects with rank constraints". In *Computer Vision and Pattern Recognition (CVPR), Proceedings of the IEEE Computer Society Conference on*, volume 1, pp. 1–493. 2001.
- Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. "Bundle adjustment - a modern synthesis". In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice (ICCV)*, pp. 298–372. Corfu, Greece. September, 2000.
- Tsai, R. Y. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses". *Robotics and Automation, IEEE Journal of*, Vol. 3, N. 4, pp. 323–344. 1987.
- Tsap, L. V., Goldgof, D. B., and Sarkar, S. "Nonrigid motion analysis based on dynamic refinement of finite element models". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, pp. 526–543. 1998.
- Tsap, L. V., Goldgof, D. B., Sarkar, S., and Huang, W.-C. "Efficient nonlinear finite element modeling of nonrigid objects via optimization of mesh models". *Computer Vision and Image Understanding*, Vol. 69, N. 3, pp. 330–350. mar, 1998.
- Tuytelaars, T. and Mikolajczyk, K. "Local invariant feature detectors: a survey". *Foundations and Trends® in Computer Graphics and Vision*, Vol. 3, N. 3, pp. 177–280. July, 2008.

- Uchiyama, H. and Marchand, E. "Deformable random dot markers". In *Mixed and Augmented Reality (ISMAR), 10th IEEE International Symposium on*, pp. 237–238. 2011.
- Ulrich, M., Steger, C., and Baumgartner, A. "Real-time object recognition using a modified generalized hough transform". *Pattern Recognition*, Vol. 36, N. 11, pp. 2557–2570. 2003.
- Ulrich, M., Wiedemann, C., and Steger, C. "Cad-based recognition of 3d objects in monocular images". In *International Conference on Robotics and Automation (ICRA)*, pp. 1191–1198. Kobe, Japan. May, 2009.
- Umeyama, S. "Least-squares estimation of transformation parameters between two point patterns". *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 13, N. 4, pp. 376–380. 1991.
- Urtasun, R., Fleet, D. J., Hertzmann, A., and Fua, P. "Priors for people tracking from small training sets". In *Proceedings of the Tenth IEEE International Conference on Computer*, volume 1, pp. 403–410. Washington, DC, USA. 2005.
- Vacchetti, L., Lepetit, V., and Fua, P. "Stable real-time 3d tracking using online and offline information". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, pp. 1385–1391. 2004.
- Varol, A., Salzmann, M., Tola, E., and Fua, P. "Template-free monocular reconstruction of deformable surfaces". In *Computer Vision, IEEE 12th International Conference on*, pp. 1811–1818. 2009.
- Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. "Pose tracking from natural features on mobile phones". In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 125–134. Cambridge, UK. September, 2008.
- Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. "Real-time detection and tracking for augmented reality on mobile phones". *Visualization and Computer Graphics, IEEE Transactions on*, Vol. 16, N. 3, pp. 355–368. 2010.
- Wagner, D. and Schmalstieg, D. *Artoolkitplus for pose tracking on mobile devices*. na. 2007.

- Warfield, S. K., Talos, F., Tei, A., Bharatha, A., Nabavi, A., Ferrant, M., McL. Black, P., Jolesz, F. a., and Kikinis, R. "Real-time registration of volumetric brain MRI by biomechanical simulation of deformation during image guided neurosurgery". *Computing and Visualization in Science*, Vol. 5, N. 1, pp. 3–11. January, 2001.
- White, R. and Forsyth, D. A. "Combining cues: Shape from shading and texture". In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 2, pp. 1809–1816. 2006.
- Wilson, E. L. "Isoparametric elements". In *Three dimensional static and dynamic analysis of structures: a physical approach with emphasis on earthquake engineering*. Computers and Structures Inc. 1998.
- Wittek, A., Hawkins, T., and Miller, K. "On the unimportance of constitutive models in computing brain deformation for image-guided surgery". *Biomechanics and modeling in mechanobiology*, Vol. 8, N. 1, pp. 77–84. 2009.
- Wohlgemuth, W. and Triebfürst, G. "Arvika: augmented reality for development, production and service". In *Proceedings of Designing augmented reality environments (DARE)*, pp. 151–152. 2000.
- Xiao, J. and Kanade, T. "Uncalibrated perspective reconstruction of deformable structures". In *Computer Vision (ICCV), Tenth IEEE International Conference on*, volume 2, pp. 1075–1082. 2005.
- Xiao, J., xiang Chai, J., and Kanade, T. "A closed-form solution to non-rigid shape and motion recovery". In *European Conference on Computer Vision*, pp. 573–587. 2004.
- Xu, K., Chia, K. W., and Cheok, A. D. "Real-time camera tracking for marker-less and unprepared augmented reality environments". *Image and Vision Computing*, Vol. 26, N. 5, pp. 673–689. 2008.
- Zhang, Z. "A Flexible New Technique for Camera Calibration". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, N. 11, pp. 1330–1334. 2000.
- Zhu, J., Hoi, S. C., Xu, Z., and Lyu, M. R. "An effective approach to 3d deformable surface tracking". In *Proceedings of the 10th European Conference on Computer Vision: Part III*, pp. 766–779. Berlin, Heidelberg. 2008.

- Zhu, J., Hoi, S. C. H., and Lyu, M. R. "Real-time non-rigid shape recovery via active appearance models for augmented reality". In *Proceedings of the 9th European conference on Computer Vision*, volume 1, pp. 186–197. Berlin, Heidelberg. 2006.
- Zhu, J. and Lyu, M. R. "Progressive finite newton approach to real-time nonrigid surface detection". In *Proc. Conf. Computer Vision and Pattern Recognition*. 2007.
- Zhu, J., Lyu, M. R., and Huang, T. S. "A fast 2d shape recovery approach by fusing features and appearance". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, pp. 1210–1224. 2009.
- Zhu, Z., Branzoi, V., Wolverton, M., Murray, G., Vitovitch, N., and Yarnall, L. "Ar-mentor: Augmented reality based mentoring system". In *Proceedings of the 13th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Munich, Germany. September, 2014.