

# COMPUTATIONAL SHAPE UNDERSTANDING for 3D RECONSTRUCTION and MODELING

THIS IS A TEMPORARY TITLE PAGE  
It will be replaced for the final print by a version  
provided by the service academique.

Thèse n. xxxx 2014  
présenté le 06 Mai 2014  
à la Faculté des Informatique et Communications  
laboratoire d'informatique graphique et géométrique  
programme doctoral en Informatique, communications et in-  
formation  
École Polytechnique Fédérale de Lausanne

pour l'obtention du grade de Docteur ès Sciences  
par

Duygu Ceylan

acceptée sur proposition du jury:

Prof Michael C. Gastpar, président du jury

Prof Mark Pauly, directeur de thèse

Prof Sabine Susstrunk, rapporteur

Prof Niloy J. Mitra, rapporteur

Dr Wilmot Li, rapporteur

Lausanne, EPFL, 2014



ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE





Two roads diverged in a wood, and I-  
I took the one less traveled by,  
And that has made all the difference.  
— Robert Frost

To my father, my mother, and my sister. . .



# Acknowledgements

I would like to give my sincere gratitude to my advisor Mark Pauly. The last five years have been very intense but full of joy of conducting outstanding research. I would like to thank for all the support and the opportunities you have provided which have enabled me to achieve things I wouldn't be able to imagine when I first joined your lab. Your guidance has helped me to mature as a researcher.

I would like to give my special thanks to Niloy Mitra, who not only has been a great collaborator but has always been a source of inspiration. Working with you has been a wonderful experience.

I would like to thank Wilmot Li for the unforgettable internship at Adobe Research, it was a dream come true... Without our encouraging discussions, it would not be possible to explore new fields with success.

Thank you so much Sabine Süssstrunk, Niloy Mitra, Wilmot Li, and Michael Gastpar for being a part of the thesis committee. I appreciate your feedbacks and suggestions for improving this dissertation.

I would like to acknowledge all of my collaborators, this thesis would not have been possible without you. Many thanks go to Hao Li for providing invaluable guidance during the first years of my PhD. Thank you Minh Dang and Boris Neubert, a deadline has never been so joyful as it has been with you guys! I would like to extend my gratitude to Maneesh Agrawala, Thibaut Weise, Youyi Zheng, and Michael Wand for their excellent collaborative efforts. I want to acknowledge Zohreh Sasanian for helping me assemble mechanical prototypes, you brought a lot of fun to the stressful deadline days. I want to thank Sawsan AlHalawani for proofreading some of my papers, Minh Dang for performing his professional kendo moves for our results, Tina J. Smith, Charlotte Emma Rakhit, and Victor Ma for video voice-overs. I highly appreciate the help of Boris Neubert, Madeleine Robert, and Baris Kasikci for helping me to translate the abstract of this thesis to German and French.

I would like to thank all my lab mates at LGG, especially my previous and current office mates Mario Deuss, Stefan Lienhard, Romain Testuz, Minh Dang, and Boris Neubert for the fun times. Special thanks go to Madeleine Robert for tirelessly handling all the administrative

## Acknowledgements

---

work and for her support during my stressful times scheduling my thesis exam. I would like to thank Vladimir Kim for being a great neighbor during my internship at Adobe Research and Sema Berkiten for making my summer more fun. I am also grateful for the companionship of Nobuyuki Umetani and Yu-Shiang Wong during my visits to UCL. I also appreciate Nobuyuki's help in answering all my questions about mechanism design.

I would like to take this opportunity to thank my close friends and neighbors Pinar Tozun and Kerem Kapucu. It was comforting to know that you are only a few minutes away. Kerem, thanks for your patience in watching all of the videos of my mechanical prototypes and listening to my endless questions. Pinar, this would not been such a special journey without your friendship and sincerity. Our Friday movie nights, *Gillmore Girls* discussions, and of course the *to-be-banned lists* has made this a colorful journey. I am grateful to the remaining members of the Turkish gang; Cansu Kaynak, Baris Kasikci, and Onur Kocberber, for all the fun times. I want to thank Danica Probic for being Pinar's office mate, and thus accommodating our large magnet collections.

Finally, I'm grateful to my father, my mother, my sister, and Taner who has recently become a part of our family. Your love has been the most meaningful support during this long PhD journey, especially during the stressful deadline times. It has been a big comfort to know that you will always respect my decisions.

My research has been supported by the funding from the European Research Council under the European Union's Seventh Framework Programme (FP 2007-2013) ERC Grant Agreement 257453, ERC Starting Grant COSYM.

*Lausanne, 02 May 2014*

# Abstract

The physical and the digital world are becoming tightly connected as we see an increase in the variety of 2D and 3D acquisition devices, e.g., smartphones, digital camera, scanners, commercial depth sensors. The recent advances in the acquisition technologies facilitate the data capture process and make it accessible for casual users. This tremendous increase in the digital content comes with many application opportunities including medical applications, industrial simulations, documentation of cultural artifacts, visual effects etc.

The success of these digital applications depends on two fundamental tasks. On the one hand, our goal is to obtain an accurate and high-quality digital representation of the physical world. On the other hand, performing high-level shape analysis, e.g. structure discovery in the underlying content, is crucial. Both of these tasks are extremely challenging due to the large amount of available digital content and the varying data quality of this content including noisy and partial data measurements. Nonetheless, there exists a tight coupling between these two tasks: accurate low-level data measurement makes it easier to perform shape analysis, whereas use of suitable semantic priors provides opportunities to increase the accuracy of the digital data.

In this dissertation, we investigate the benefits of tackling the *low-level data measurement* and *high-level shape analysis* tasks in a coupled manner for 3D reconstruction and modeling purposes. We specifically focus on image-based reconstruction of urban areas where we exploit the abundance of symmetry as the principal shape analysis tool. Use of symmetry and repetitions are reinforced in architecture due to economic, functional, and aesthetic considerations. We utilize these priors to *simultaneously* provide non-local coupling between geometric computations and extract semantic information in urban data sets.

Concurrent to the advances in 3D geometry acquisition and analysis, we are experiencing a revolution in digital manufacturing. With the advent of accessible 3D fabrication methods such as 3D printing and laser cutting, we see a cyclic pipeline linking the physical and the digital worlds. While we strive to create accurate digital replicas of real-world objects on one hand, there is a growing user-base in demand of manufacturing the existing content on the other hand. Thus, in the last part of this dissertation, we extend our shape understanding tools to the problem of designing and fabricating functional models. Each manufacturing device comes with technology-specific limitations and thus imposes various constraints on the digital models that can be fabricated. We demonstrate that, a good level of shape understanding is necessary to

## **Abstract**

---

optimize the digital content for fabrication.

Key words: shape analysis, 3D reconstruction, symmetry detection, computational design, fabrication constraints

# Zusammenfassung

Die Grenze zwischen digitaler und physikalischer Welt verschwimmt zusehends, verursacht durch die zunehmende Verbreitung von digitalen Aufnahmegeräten, die 2D (beispielsweise Smartphone und Digitalkameras) sowie 3D (z.B. Laserscanner oder kommerzielle Tiefensensoren) Daten verarbeiten können. Die jüngsten Fortschritte dieser Technologie vereinfachen den Datenerhebungsprozess und ermöglichen deren Einsatz auch für unbedarfte Benutzer. Die Zunahme von digitalen Inhalten führt zu unzähligen Anwendungsmöglichkeiten, beispielsweise in der Medizin, bei industriellen Simulationen, der Dokumentation von Kulturgütern oder für visuelle Effekte.

Der Erfolg dieser digitalen Anwendungen ist von zwei grundlegenden Problemstellungen abhängig. Einerseits ist unser Ziel eine exakte und qualitative hochwertige digitale Repräsentation der physikalischen Welt. Andererseits ist es notwendig digitale Inhalte, ähnlich dem menschlichen Verstand, auf einem abstrakten und allgemeinen Level zu verstehen. Diese beiden Aufgaben stellen, unter anderem wegen der Vielzahl digitaler Inhalte als auch wegen der variierenden Datenqualität dieser Inhalte, verursacht durch Rauschen oder partiell fehlende Daten, eine große Herausforderung dar. Nichtsdestoweniger existiert eine enge Verbindung zwischen diesen beiden Problemstellungen: eine exakte und hochwertige Datengrundlage erleichtert und begünstigt das abstrakte Verstehen und Begreifen der digitalen Inhalte, wobei die Verwendung passender semantischer a-priori Informationen (Priors) im Gegenzug die Genauigkeit der digitalen Daten erhöht.

In dieser Dissertation untersuchen wir die Vorteile einer gemeinsamen Betrachtung von Datenerhebung und abstraktem Verstehen des Inhaltes im Bereich der 3D Rekonstruktion und Modellierung. Im Besonderen sind wir an bildbasierter Rekonstruktionsverfahren städtischer Umgebungen unter Verwendung von Symmetrieinformationen als allgemeines Analysewerkzeug interessiert. Die Verwendung von Symmetrie und Repetition bietet sich innerhalb der Architektur aus Gründen der Wirtschaftlichkeit sowie aus funktionalen und ästhetischen Gründen an. Wir verwenden diese Priors um gleichzeitig übergreifende Verknüpfung zwischen geometrischer Berechnung und Extraktion semantischer Informationen in städtischen Datensätzen zu erreichen.

Neben den Fortschritten innerhalb der 3D Geometrieerzeugung und -analyse erleben wir momentan eine Revolution im Bereich der digitalen Fertigung. So ermöglichen 3D Druck und Lasercutting eine cyclische Verbindung zwischen physikalischer und digitaler Repräsentation. Während wir auf der einen Seite danach streben möglichst exakte Repliken von realen Objekten

## **Zusammenfassung**

---

zu schaffen, steht auf der anderen Seite der Bedarf nach einer physikalischen Repräsentation von existierenden Inhalten. Daher erweitern wir das Anwendungsgebiet der entwickelten Werkzeuge auf Design- und Herstellungsprobleme funktionaler Modelle. Jedes Herstellungsverfahren ist mit verfahrensspezifischen Limitierungen verbunden und schränkt damit die Art der Modelle, die tatsächlich physikalisch erzeugt werden können, ein. Wir zeigen, dass ein genaues Verständnis für die Art und Ausprägung eines Objektes notwendig ist, um digitale Inhalte für den Herstellungsprozess zu optimieren.

Stichwörter: Formanalyse, 3D Rekonstruktion, Symmetrie Extraktion, Fabrikation



# Résumé

Les mondes physique et numérique deviennent étroitement liés depuis que nous observons une augmentation de la variété des dispositifs d'acquisition de 2D et 3D (smartphones, appareils photo numériques, scanners, capteurs de profondeur commerciaux). Les avancées récentes relatives aux technologies d'acquisition facilitent le processus de saisie des données et les rendent accessibles aux utilisateurs occasionnels. Cet accroissement considérable des contenus numériques ouvre maintes opportunités incluant les applications médicales, les simulations industrielles, la documentation sur les artefacts culturels, les effets spéciaux, etc.

Le succès de ces applications numériques dépend de deux fonctions fondamentales. D'une part, notre objectif est d'obtenir une représentation précise et de haute qualité du monde physique. D'autre part, produire une compréhension de haut niveau du contenu numérique proche de celle l'être humain est crucial. Ces deux tâches sont extrêmement difficiles en raison de la large quantité de contenus numériques disponibles et de la qualité variable de ces données à cause notamment de les mesures bruyantes et partielles. Il existe néanmoins un couplage étroit de ces deux tâches : la mesure précise des données de bas niveau permet de produire une compréhension de haut niveau du contenu numérique quand l'utilisation d'antécédents sémantiques adéquats fournit des opportunités d'accroître l'exactitude des données numériques.

Dans cette thèse, nous étudions les avantages de la mesures de données de bas niveau et la compréhension de haut niveau des formes d'une manière couplée pour la reconstruction 3D et la modélisation. Nous nous concentrons en particulier sur la reconstruction de zones urbaines à partir d'images où nous exploitons l'abondance de symétrie comme outil principal d'analyse de formes. L'utilisation de la symétrie et des répétitions est renforcée en architecture en raison de considérations économiques, fonctionnelles et esthétiques. Nous utilisons ces antécédents pour fournir simultanément des couplages non-locaux entre les calculs géométriques et les extractions des informations sémantiques des ensembles de données urbaines.

Parallèlement aux progrès dans l'acquisition et l'analyse de la géométrie 3D, nous vivons une révolution dans la fabrication numérique. Avec l'avènement des méthodes accessibles de fabrication 3D telles que l'impression 3D et le découpage laser, nous assistons à un cycle périodique liant les mondes physique et numérique. Alors que nous nous efforçons d'un côté de créer des répliques numériques précises d'objets réels, il y a d'un autre côté une demande croissante pour la fabrication de contenus existants. Ainsi, dans la dernière partie de cette thèse, nous étendons nos

## Résumé

---

outils de compréhension de formes au problème de la conception et la fabrication des modèles fonctionnels. Chaque appareil de fabrication comporte des limitations technologiques spécifiques et impose ainsi différentes contraintes sur les modèles numériques qui peuvent être fabriqués. C'est ce que nous démontrons, un bon niveau de compréhension de formes est nécessaire à l'optimisation du contenu digital pour la fabrication.

Mots clefs : analyse de formes, reconstruction 3D, détection de symétrie, les contraintes de fabrication

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract (English/Français/Deutsch)</b>	<b>vii</b>
<b>List of figures</b>	<b>xiv</b>
<b>List of tables</b>	<b>xx</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Objectives and Challenges . . . . .	3
1.2 Contributions . . . . .	7
1.3 Organization . . . . .	7
<b>2 3D Urban Modeling Revisited</b>	<b>9</b>
2.1 Overview of Approaches . . . . .	9
2.2 Image-based Modeling . . . . .	12
2.2.1 Camera Model . . . . .	13
2.2.2 Epipolar Geometry . . . . .	15
2.2.3 Recovering Camera Parameters . . . . .	16
2.2.4 Triangulation . . . . .	20
2.2.5 Structure-from-Motion . . . . .	20
2.2.6 Multi-view Stereo . . . . .	22
2.2.7 Challenges . . . . .	23
<b>3 Factored Acquisition of Buildings</b>	<b>27</b>
3.1 Overview . . . . .	28
3.2 Algorithm Details . . . . .	29
3.2.1 3D Line Reconstruction . . . . .	29
3.2.2 Piecewise-Planar Model Generation . . . . .	31
3.2.3 Discrete Symmetry Extraction . . . . .	34
3.2.4 Procedural Depth Refinement . . . . .	38
3.3 Evaluation . . . . .	38
3.4 Closing Remarks . . . . .	43

## Contents

---

<b>4</b>	<b>Symmetry and Structure-from-Motion</b>	<b>45</b>
4.1	Overview . . . . .	47
4.2	Algorithm Details . . . . .	48
4.2.1	Initial Grid Estimation . . . . .	48
4.2.2	Symmetry-based Image Registration . . . . .	49
4.2.3	Symmetry-based Structure-from-Motion . . . . .	53
4.2.4	Vanishing Line Refinement . . . . .	55
4.2.5	Extension to Multiple Grids . . . . .	55
4.3	Evaluation . . . . .	57
4.4	Applications . . . . .	62
4.5	Closing Remarks . . . . .	65
<b>5</b>	<b>Understanding Structured Variations</b>	<b>67</b>
5.1	Overview . . . . .	69
5.2	Algorithm Details . . . . .	70
5.2.1	Template Models . . . . .	70
5.2.2	Template Fitting . . . . .	72
5.2.3	Coupled Template Matching and Deformation . . . . .	74
5.3	Evaluation . . . . .	80
5.4	Closing Remarks . . . . .	87
<b>6</b>	<b>Designing Functional Models</b>	<b>89</b>
6.1	Overview . . . . .	91
6.2	Algorithm Details . . . . .	93
6.2.1	Mechanical Design . . . . .	93
6.2.2	Motion and Layout Optimization . . . . .	98
6.2.3	Unified Layout Design . . . . .	104
6.3	Evaluation . . . . .	105
6.4	Closing Remarks . . . . .	109
<b>7</b>	<b>Conclusion and Future Work</b>	<b>111</b>
7.1	Summary and Take-Home Messages . . . . .	111
7.2	Open Questions and Future Work . . . . .	114
<b>A</b>	<b>SfM Comparisons</b>	<b>117</b>
<b>B</b>	<b>Template Database</b>	<b>125</b>
<b>C</b>	<b>Chapter 5 Detailed Results</b>	<b>127</b>
	<b>Bibliography</b>	<b>151</b>
	<b>Curriculum Vitae</b>	<b>153</b>

# List of Figures

1.1	This dissertation investigates the coupling between the low-level digitization and high-level shape analysis tasks for reconstructing the physical world. The acquired information is beneficial for a variety of applications. . . . .	2
1.2	Use of symmetry and repetitions are reinforced in architecture at multiple scales due to economic, functional, and aesthetic considerations. . . . .	3
1.3	Our goal is to integrate structure-discovery into the 3D reconstruction process to yield high-quality 3D models. . . . .	4
1.4	By injecting symmetry information early into the reconstruction process, our goal is to compute the camera parameters of a set of input images and detect 2D/3D symmetry patterns simultaneously. The symmetry information provides a novel link between the input images and the 3D output to enable interesting interaction possibilities. . . . .	5
1.5	Elements that exhibit variations of a base geometry are common in ornate historic buildings. . . . .	5
1.6	Commercial motion sensing input devices such as the Microsoft Kinect enable direct capturing of human motions. . . . .	6
2.1	Urban reconstruction has a wide spread application domain including urban design, emergency planning and entertainment industry. . . . .	10
2.2	The modeling of urban spaces has been performed using a variety of different approaches. (Images courtesy of corresponding authors.) . . . . .	11
2.3	<i>Pinhole camera geometry</i> . The camera center, $\mathbf{C}$ , is placed at the origin and the image plane is placed in front of the camera center. The principal point, $\mathbf{p}$ , is the point where the principal axis meets the image plane. . . . .	13
2.4	The transformation between the camera and the world coordinate frames. . . . .	14
2.5	The geometric relations of point correspondences between a pair of images. . . . .	15
2.6	The four possible solutions for the camera position and orientations recovered from their essential matrix. Only in the first configuration, the reconstructed point is in front of both cameras. . . . .	19
2.7	In the ideal case, the rays projected through corresponding points intersect resulting in a 3D point (shown in blue). In practice, these rays do not necessarily intersect (shown in red). . . . .	20

## List of Figures

---

2.8	<i>Photo Tourism</i> system developed by Snavely et al. [103] is a seminal work in using large image collections from the Internet in a SfM pipeline. (Image courtesy of Snavely et al.) . . . . .	21
2.9	Several factors such as variation in illumination across images, reflective or textureless surfaces, large occlusions, and degrading resolution for upper floors of buildings make the correspondence problem very challenging. . . . .	23
2.10	Even though MVS methods such as PMVS [33] output impressive results, they still suffer from high noise levels, specially along the direction of projective rays. . . . .	24
2.11	Dominant repetitions in urban data sets and provide important structural priors to augment the reconstruction process. . . . .	24
3.1	In a typical image-based 3D reconstruction pipeline, the structure-from-motion step is followed by a dense reconstruction of the captured scene. . . . .	27
3.2	We propose a reconstruction pipeline where we explore structural priors at two levels: while line and plane features exploit continuous symmetries, repetitive elements model discrete symmetries. . . . .	28
3.3	Given an edge $e_k$ in image $I_i$ , its matching edge in image $I_j$ falls inside the region defined by the epipolar lines (shown in orange) corresponding to the endpoints of $e_k$ . For each potential match, a similarity score is computed by comparing small patches (shown in gray) along the matching sample points on $e_k$ and $e_m$ . . . . .	30
3.4	Edges detected directly on images often contain outliers from reflections, shadows, occluding elements etc. (middle). We employ matching and triangulation at the edge level to prune out such outliers (right). . . . .	31
3.5	We use an MRF-based plane labeling method to segment the input images. The segmentation is cleaned using the intersection lines of neighboring planes. User input resolves regions of insufficient feature lines. . . . .	34
3.6	Symmetry refinement is performed on the initially detected repetitions to initialize the line and the transformation parameters for grid fitting. After the missing elements are detected using the refined symmetry information, the optimization procedure is repeated to get the final grid alignment. This procedure consists of an <i>inner</i> loop of successive iterations of line and transformation optimization and an <i>outer</i> loop of updating the template strokes and reselecting the close edges. . . . .	37
3.7	User-guided depth refinement based on the extracted symmetric elements helps to recover shallow depth features. We show recovered geometry with and without texture. . . . .	38
3.8	For each example, we show the optimized repetition patterns with different colors indicating separate structures. The red planes shown in the final reconstructions have been added with user assistance due to lack of stable line features. <i>Building 4</i> and <i>table</i> examples have highly reflective surfaces. . . . .	39
3.9	We perform synthetic evaluations to measure the accuracy of our approach. . . . .	40

3.10	The comparison with the patch-based MVS method [33] illustrates that symmetry priors and non-local consolidation are essential for objects with complex materials and repetition patterns. The method of Wu et al. [119] fails to recover the depth of the repeating elements if the depth change with respect to the main plane is too small. We provide depth assignments computed by different weighting terms ((a) no repetition term, (b) repetition and smoothness terms weighted equally, (c) smoothness term weighted more). We refer the reader to the paper for details. . . . .	42
4.1	Given a set of images, we perform symmetry detection in each image based on a user marked template (in orange). We use this symmetry information (shown in yellow) to solve for a consistent global repetition pattern using a graph-based optimization. We then use a symmetry-based SfM method to simultaneously calibrate the cameras and extract a 3D reconstruction. . . . .	47
4.2	Given a pair of rectified images (left) with a repeating element $T$ marked in one image (top-left), we use SIFT features to estimate the scale factor $s$ relating the image pair (middle). We then create a scaled template $sT$ suitable for the other image. Thus, we detect image-level repetitions (in yellow) across each individual rectified image (right). . . . .	49
4.3	Given a pair of images with their detected grids (left), for each candidate alignment we detect the overlapping grid regions (shown in green). We compute feature matches outside the overlapping regions and count the number of matches supporting the candidate alignment. Top-right shows the correct alignment with highest support (51 matches) and the bottom-right is a wrong candidate alignment (25 supporting matches). . . . .	50
4.4	Starting from all candidate pairwise matches, we introduce an optimization that iteratively improves the quality of the alignments (wrong alignments are shown in red). The minimum spanning tree of the final graph (shown as solid edges) provides the final image alignments. . . . .	52
4.5	For the images $I_1, I_2$ , and $I_3$ (in top), the wrong candidate alignment between $(I_1, I_2)$ (in red) is replaced by the correct accumulated alignment along the shortest path $I_1 \rightarrow I_3 \rightarrow I_2$ during the iterative grid optimization. For each alignment, the overlapped images are shown together with the mapped grid regions (yellow and green). . . . .	53
4.6	Once we obtain the initial 3D representation, we refine the rectification of the input images and repeat the symmetry-based SfM step. In the initial image matching step, the windows in red have been grouped together with the windows in yellow (resulting in a 2-by-4 grid) but have been discarded due to high projection error (resulting in a 2-by-3 grid). . . . .	56
4.7	For each example, we provide a sample input image, the user marked template in a single image (orange), the extracted repetition pattern and the calibrated cameras. This information is used for a range of image manipulations. . . . .	58

## List of Figures

---

4.8	Our approach handles buildings with multiple facades while preserving the orientation of the individual facades both for orthogonal ( <i>Building 8</i> ) and non-orthogonal ( <i>Building 7</i> ) relations. We provide a satellite imagery of <i>Building 7</i> for reference. . . . .	59
4.9	For the <i>Building 1</i> ) data set, the method of Jiang et al. [48] registers 21 out of 26 images. For the <i>Building 7</i> example, our method produces significantly higher-quality output especially for the right facade of the building highlighted in orange. . . . .	59
4.10	For the <i>Building 1</i> data set, the method of Jiang et al. [48] registers 21 out of 26 images. For the <i>Building 7</i> example, our method produces significantly higher-quality output especially for the right facade of the building highlighted in orange. . . . .	61
4.11	For the <i>Building 9</i> data set, due to lack of sufficient discriminating feature matches our method fails to resolve the ambiguities. The dense reconstructions computed with the camera parameters obtained from our method and Bundler are shown. The method of Zach et al. [122] does not produce any camera parameters. . . . .	62
4.12	Occlusion mask marked in one image (top-left) is propagated to the other images. Occlusion removal using propagated information from other images provides significantly improved results compared to the single-image based state-of-the-art <i>PatchMatch</i> [8] method. . . . .	64
4.13	Extracted facade grid patterns (left) are changed and then composited with the foreground (e.g. lamp post). These changes are propagated to the other images. . . . .	65
5.1	For many buildings, similar elements can be arranged by varying transformations, located at different facades of a building, and exhibit certain structured variations. . . . .	67
5.2	(a) In case of perfect input data, elements with the same geometry are mapped to a single point in the 2-dimensional deformation space of a rectangular template $T$ . (b) The presence of noise and missing data, however, makes it challenging to observe clear clusters in the deformation space. Similarity matrices computed using the pairwise element distances in the deformation space reveal this behavior. . . . .	69
5.3	The feature lines of the given template model are shown in red. Each feature line consists of individual atomic wires characterized by a set of parameters. For example, circular atomic wires are characterized by the center ( $\mathbf{C}$ ), radius ( $r$ ), mid-angle ( $\phi$ ), and the opening angle ( $\alpha$ ) of the arc. We detect orthogonality, equal-length, and reflection constraints among the feature wires of the model (right). . . . .	71
5.4	For the templates $T_0$ and $T_1$ , we illustrate various instances ( $T_0^0, \dots, T_0^2, T_1^0, \dots, T_1^2$ ) with different parameters of the detected feature wires (shown in red). Each instance is represented as a point in the deformation space of the corresponding template based on these parameters. A multi-dimensional scaling projection of the deformation space of the templates is shown. . . . .	72



5.5	Given a MVS reconstruction of a building, we utilize a set of templates to match its elements, i.e. windows. We combine observations from template deformations via a subspace analysis to extract element relations. Using these relations as constraints, we label each element with a deformed template instance (same instances are denoted in same color). We repeat template deformation by consolidating data across elements matched to similar template instances. This analysis reveals elements that are identical (represented as red blocks in smoothness weight matrices) or share partial similarities (highlighted in green on the matching templates). . . . .	74
5.6	We propose an iterative approach to simultaneously find the matching template instances of a given set of elements and compute similarities among them. Intuitively, at each iteration we improve template fitting by consolidating data across multiple elements via the detected similarity relations. Thus, potential element clusters become clearer as observed by the formation of red element blocks in the smoothness matrices. . . . .	79
5.7	Due to the coupling introduced by the smoothness term, the amount of variation among the elements affects the final selection of template instances. For different set of templates (with feature wires shown in red) and elements, we show the selected template instances once based on data term only and in the second row additionally considering the smoothness term. . . . .	81
5.8	We show the selected template instances for a synthetic house model (consisting of 38 narrow triangle-top, 4 wide triangle-top, 23 long arch, and 23 short arch windows) when using different number of templates. For each case, we also show the color-coded smoothness matrices and the partial similarities detected between the elements (highlighted in green). Note that the removal of the triangle-top template selected in (a) results in a selection of another triangle-top window in (b). 82	82
5.9	We evaluate our algorithm on MVS reconstructions obtained from rendered images of a synthetic model. Due to loss of fine details, we cannot recover the subtle variation in width of the triangle-top windows in blue (a) and the occlusion by a large tree results in wrong template assignments for some elements (b). . . .	83
5.10	(a) Individual template fitting for a set of elements results in the selection of 5 different templates whereas our algorithm assigns the elements to 5 different instances of the same template. (b) Given a template selection, we visualize each element in the low-dimensional deformation space of the template (elements that are exact replicas are shown in same color) using the deformation parameters obtained by individual fitting vs. our algorithm. The clusters generated by the k-means (k=5) algorithm are indicated by different symbols. . . . .	84
5.11	Color bars at the sides of the final smoothness matrix denote the color of the fitting template instances of the corresponding block of identical elements. We show the partial similarities detected across such element blocks in the accompanying graph by indicating the identical parts of the matching template instances in green. 85	85

## List of Figures

---

5.12	For each example, we show the smoothness matrices in the first (top left) and final (bottom left) iterations of our algorithm. Color bars at the sides of the matrices denote the color of the matching template instances of the corresponding block of identical elements. Partial similarities detected across element blocks are shown on the corresponding templates in green. We denote the elements matched to wrong template instances with dotted circles. . . . .	86
5.13	Due to the lack of sufficient 3D lines resulting from large occlusions, our algorithm fails to match the indicated windows (in orange) to the correct template instances. It is possible to augment our analysis with additional priors to resolve such failures. We show the element smoothness matrices with and without use of such priors. . . . .	87
6.1	When performing a periodic motion such as walking, each bone of a humanoid figure undergoes an oscillation, possibly with different phase and frequency. . . .	92
6.2	The basic components of our mechanical designs include gears, pulleys, and four-bar linkages. . . . .	92
6.3	We drive the rigid links of the automaton with oscillation modules that consist of gears, pulleys, and four-bar linkages. These components are stacked onto axles that are connected to each link. We show the stacking order of the parts starting from the bottom layer (a), as well as an exploded view (b). . . . .	94
6.4	The output of an oscillation module is defined by a set of motion parameters including the lengths and the initial orientation of the linkage bars (a). The motion of a one module in a chain of modules depends on the motion of its parent link (b). . . . .	95
6.5	We apply a two-step optimization procedure to optimize for the motion parameters of the oscillation modules used in our designs. . . . .	99
6.6	The main support structure of the mechanical automaton, the torso, is oriented based on the motion planes of the moving limbs. Both sides of this support is used to accommodate the pulleys propagating the input crank rotation to the limbs. In special cases, the shoulders are used as support structures as well. . . .	104
6.7	For each of the example automata generate by our system, snapshots of the original motion sequence and the corresponding simulation result of the automata are given for various time frames. . . . .	107
6.8	Physical prototypes of the <i>walking</i> (top) and the <i>dancing</i> examples show several snapshots of the input figure performing the desired motion and the corresponding mechanical automata. . . . .	108

# List of Tables

- 3.1 The table shows the number of input images ( $N_i$ ), the resolution of the images in megapixels (res), the average number of 2D edges detected ( $N_e$ ), the number of 3D lines reconstructed ( $N_l$ ), the number of automatically fitted planes ( $N_p$ ), the number of manually selected planes ( $N'_p$ ), the number of elements marked by the user ( $N'_r$ ), and the total number of repeating elements detected ( $N_r$ ) for each data set. The computation times for 3D line reconstruction ( $T_l$ ) and plane-based image segmentation ( $T_p$ ) are given in minutes measured on a 3.33 MHz 24-core machine. . . . . 40
- 4.1 The table shows the number of input images ( $N_i$ ), the resolution of the images in megapixels (res), and the total number of repeating elements detected ( $N_r$ ) for each data set. We also report how our method, Bundler, and the method of Zach et al. perform: a correct reconstruction is produced (*yes*), the output is poor in quality (*poor*), there is a confusion in the number of repeated elements (*conf.*), or reconstruction contains multiple misaligned components (*mult.*). The computation times for image-based symmetry detection ( $T_s$ ) and a single iteration of symmetry-based SfM ( $T_o$ ) are given in minutes measured on a 2.8 GHz 4-core machine. . . . . 57
- 5.1 The table shows the number of input images ( $N_i$ ), the number of user selected elements ( $N_s$ ), the total number of detected elements ( $N_e$ ), the numbers of templates selected by the independent analysis ( $T_d$ ) and with the coupled analysis ( $T_c$ ), and the total number of template instances discovered ( $T_i$ ). . . . . 85



# 1 Introduction

The physical world is becoming more and more interconnected with the digital world as we witness a tremendous increase in the variety of 2D and 3D acquisition devices, e.g., smartphones, point-and-shoot cameras, scanners, and commercial depth sensors. The recent advances show a clear improvement in the accuracy and the resolution of these sensing devices while making it more convenient for casual users to acquire digital content. This ease of acquisition brings up many application opportunities in scientific and commercial fields that rely on collection of large and complex data sets. Common applications include mechanical prototyping, documentation of cultural artifacts, design of medical implants, robotics, urban modeling, etc.

Often the success of these applications that center around 3D reconstruction depends on two fundamental tasks: **accurate low-level data measurement** and **high-level shape analysis**. Imagine an urban planing scenario such as the renovation of an architecture site or design of an urban area. Typically, users with different backgrounds like stakeholders, architects, and city planners need to exchange ideas. Very often, these users do not share a common background and are not familiar with each other's conventions. In such cases, it is critical to present and discuss ideas using an effective representation such as the digital replica of the physical site. On the one hand, an accurate and high-quality 3D reconstruction of the area is necessary for careful planning, prototyping, or running environmental simulations. On the other hand, automatic extraction of semantic knowledge from the acquired digital content similar to a human being is crucial. For example, explicit encoding of the elements of a building enable direct edits such as modification or replacement of these elements while patterns detected in arrangements of the elements help to easily propagate these edits.

The increase in the size of the available digital content places additional demands on the *low-level digitization* and *high-level shape analysis* tasks to handle the resulting complexity. These tasks become particularly challenging since large amounts of digital data often come with varying quality including a significant amount of noisy and partial data measurements. Nonetheless, accurate low-level data measurement makes it easier to provide a high-level analysis of the acquired content, whereas use of suitable semantic priors provides opportunities to increase the

## Chapter 1. Introduction

---

reconstruction accuracy. Therefore, instead of tackling these tasks individually, exploiting the tight coupling between them enables to overcome most of these challenges.

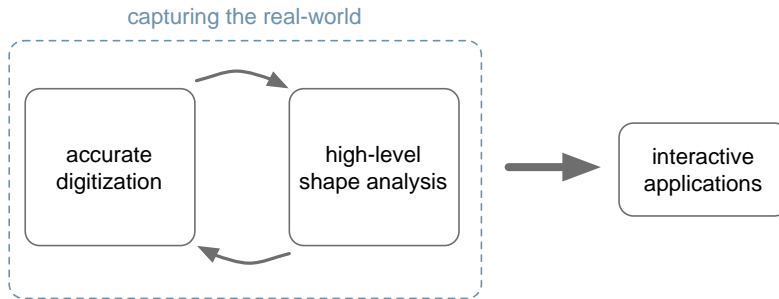


Figure 1.1: This dissertation investigates the coupling between the low-level digitization and high-level shape analysis tasks for reconstructing the physical world. The acquired information is beneficial for a variety of applications.

This dissertation investigates the benefits of tackling the tasks of *digitization* and *shape analysis* in a coupled manner in the specific problem domain of urban reconstruction (see Figure 1.1). We consider 3D modeling of urban spaces an important and challenging problem whose outcome is beneficial for various applications centered around digital 3D cities including urban planning and design, mapping and navigation, and content creation for entertainment. These applications heavily depend on accurate 3D building models to enable tasks that require interaction with street-level buildings and facades.

An important principle in our investigations is to explore the notion of *symmetry*. Generally speaking, symmetry preserves certain properties of an object under some operation applied to it [112]. In the context of geometry, we consider the geometric transformations, e.g. rotations, translations, reflections, as the symmetry operations. In the physical world, we observe geometric symmetries at various scales including the reflectional symmetry of the human body, rotational symmetry of a starfish, or the regular pattern of an insect eye. This abundance of symmetry in nature has inspired humans to incorporate symmetry in various fields including visual arts, music, and architecture [72]. In architectural designs, symmetry has been utilized due to its impact on economical and functional considerations and aesthetic concerns (see Figure 1.2). Thus, detection of symmetry in such data sets has been an important problem in geometry processing.

In our analysis, we consider symmetry as a means to *provide non-local coupling between geometric computations* and to *extract and create semantic information* in urban data sets. By consolidating information across multiple observations of the underlying geometry, we aim to provide high-quality 3D models of urban spaces. In addition, we explore the use of symmetry as a guiding principle to facilitate the development of novel interaction metaphors with the captured real-world scenes.

While we focus on 3D reconstruction of architectural data sets, we believe that our results provide useful insights for other disciplines that are concerned with the digitization and understanding



Figure 1.2: Use of symmetry and repetitions are reinforced in architecture at multiple scales due to economic, functional, and aesthetic considerations.

of large-scale geometric data sets. Concurrent to the efforts in 3D geometry acquisition and analysis, we are experiencing a revolution in digital manufacturing. Recent advances in rapid prototyping technologies, including 3D printers, laser cutters, and CNC machines, have created a growing userbase in demand of tools that enable them to create digital content to be manufactured by such devices. Such manufacturing technologies, however, often come with device-specific limitations that impose additional requirements on the digital models that can be fabricated. Thus, specialized digital geometry processing algorithms are needed to close this gap between acquisition and production [11].

We present one of the early efforts in automating the process of designing and fabricating functional models, specifically mechanical humanoid figures performing everyday actions like walking and dancing. We demonstrate that a good *geometric understanding* of the digital content is crucial to fulfill the requirements of such an automated system to create properly functioning physical prototypes.

### 1.1 Objectives and Challenges

This dissertation investigates the fundamental question of how to accurately digitize the physical world while providing a high-level shape analysis of the acquired digital data. In particular, we are interested in exploring the coupling between the digitization and shape analysis tasks in the context of 3D urban modeling. The shape analysis tasks we perform centers around structure discovery, i.e. detection of symmetric and repeating building elements.

Having a variety of applications centered around 3D digital cities, reconstruction of urban spaces has attracted a lot of attention from the research community. Chapter 2 provides an overview of the different methods proposed in this domain. While different 3D acquisition possibilities

## Chapter 1. Introduction

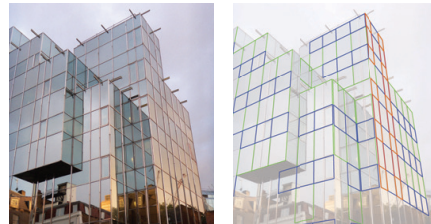
---

exist, image-based modeling methods have become one of the most popular due to the simplicity and economical advantages of the image acquisition process. This simplicity, however, comes with stronger demands on the processing algorithms. Fundamentally, any algorithm that uses *triangulation* to infer 3D information from images has to address the difficult and often ambiguous *correspondence problem*, i.e. identify the point-pairs that represent the same world space location between any image pair.

Advances in the camera technology and the processing algorithms have led to significant improvements in the quality of 3D reconstructed models [102]. Despite this success, many challenges arising from lighting variations, insufficient textures, or occlusions remain unsolved in establishing robust correspondences across input images. Furthermore, most traditional methods use local feature matching in combination with local smoothness priors [33] to produce 3D samples. Such local processing can lead to high noise levels and a significant amount of outliers.

Our goal is to exploit the concept of symmetry, which is regularly used as an organizing principle in urban planning and design, to overcome these shortcomings. Use of repeated structures is often reinforced to ease the construction process and such repetitions provide multiple observations of the same geometry.

Our objective is to combine these observations to obtain clean, precise, and high-quality 3D building models. In addition, we intent to use symmetry as a means to extract semantic information since repeating structures are often composed of elements such as windows on a facade (see Figure 1.3). This information is particularly useful for applications that focus on post-processing and editing of the acquired geometry.



While repetitions provide the means to consolidate information about the underlying geometry, they also come with the inherent ambiguity issue. The correspondence problem becomes particularly challenging in presence of repeated elements that give rise to multiple and ambiguous correspondences. Traditional image-based methods that do not explicitly take symmetries into account suffer from large-scale ambiguities and exhibit one of the following artifacts: (i) they produce suboptimal reconstructions that are sparse and noisy, or (ii) they generate apparently reasonable 3D output, but with an incorrect number of repeated elements. We observe a cyclic dependency in the problem of 3D reconstruction with repeating structures: stable symmetry detection requires reliable 3D information, while accurate reconstruction requires stable symmetry detection to resolve ambiguities. Our goal is to break this

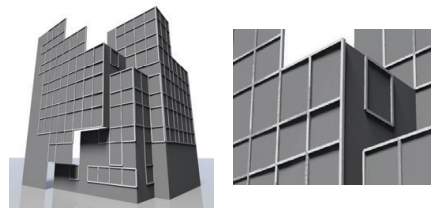


Figure 1.3: Our goal is to integrate structure-discovery into the 3D reconstruction process to yield high-quality 3D models.



dependency by injecting symmetry information early into the reconstruction process.

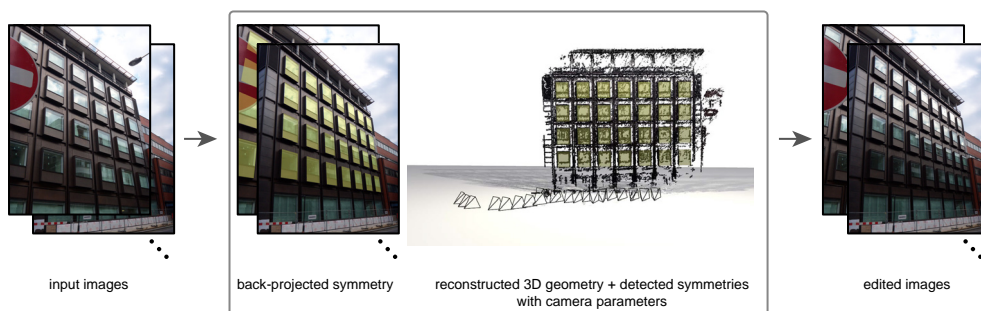


Figure 1.4: By injecting symmetry information early into the reconstruction process, our goal is to compute the camera parameters of a set of input images and detect 2D/3D symmetry patterns simultaneously. The symmetry information provides a novel link between the input images and the 3D output to enable interesting interaction possibilities.

By explicitly detecting regularities in the input images, our aim is to reduce the search space of possible geometric relations between image pairs and guide the correspondence search. By formulating this problem as a coupled optimization, we also aspire to refine the detected symmetry relations and obtain a globally consistent 3D reconstruction with explicit encoding of the regularities (see Figure 1.4). This is crucial to link the input images and the 3D scene which enables several interactive editing applications.

Exploiting the presence of exact repetitions of the same geometry arranged in 1- or 2- dimensional grids is particularly beneficial for the reconstruction of most buildings we see in our every-day lives. More complex architectural data sets, such as ornate historic buildings, on the other hand, also consist of elements that exhibit interesting variations of the same base geometry. For example, windows with similar top arches but varying height or width are common (see Figure 1.5). Detecting such structural relations between the elements of a building is a central goal of shape analysis. Performing such an analysis in raw data measurements such as image-based reconstructions, however, is a challenging task due to noisy and partial data measurements. Our goal is to understand such patterns of deformation and variation without making any prior assumption on the structure and the spatial arrangement of these patterns. By exploring these structural similarities in a general sense, we aim to provide high-level correspondences among the elements of a shape. These correspondences are useful for non-local consolidation of the data, providing guidelines for fitting new geometry, and editing.



Figure 1.5: Elements that exhibit variations of a base geometry are common in ornate historic buildings.

## Chapter 1. Introduction

---

While, we mainly focus on shape analysis and processing algorithms for the task of reconstructing the physical world, in the last part of this dissertation, we extend our findings to the problem of recreating the physical world from this acquired digital content. Development of sensing technologies that measure the shape and motion of objects has created a growing interest in creating physical replicas of both static and movable digital models. Many online services such as Shapeways [99] and Ponoko [88] offer manufacturing facilities, such as 3D printing and laser cutting, for casual users to manufacture the available digital content. However, due to the underlying technology, different manufacturing techniques impose specific constraints on the digital models that can be fabricated. For example, laser cutters are capable of cutting only planar pieces. 3D printers often have limitations on the size of the models that can be printed. Thus, a pre-segmentation is necessary when large models are desired to be printed. This segmentation is required to ensure each printed segment can later be assembled together. As a consequence, between acquisition and production, there is a need for a set of geometry processing algorithms concerned with modifying and processing the acquired digital content to satisfy these constraints [11]. Typical operations include shape simplification, filtering operations for noise removal, and geometry analysis. Such operations often require advanced optimization and processing techniques. Therefore, there is a need for tools that can automate this process and make it accessible for casual users.

We focus on automating the design and fabrication processes in the specific problem domain of fabricating movable models. In particular, our goal is to design and fabricate mechanical humanoid figures that mimic every-day actions such as walking and dancing. Such animation sequences can be acquired via motion-capture systems or commercial motion sensing input devices such as the Microsoft Kinect as shown in Figure 1.6. Preparation of such acquired digital content for fabrication becomes particularly challenging if the captured digital content consists of noisy data measurements as in the case of the Microsoft Kinect output. Our goal is to overcome these challenges by providing a high-level understanding of the input animation sequences by detecting patterns in the target motions that are desired to be realized. Such patterns enable the design of generic mechanical components with configurable parameters that can be automatically adjusted to approximate given target motions. Furthermore, we aim to consider ease-of-fabrication and physical validity during the design of such mechanical

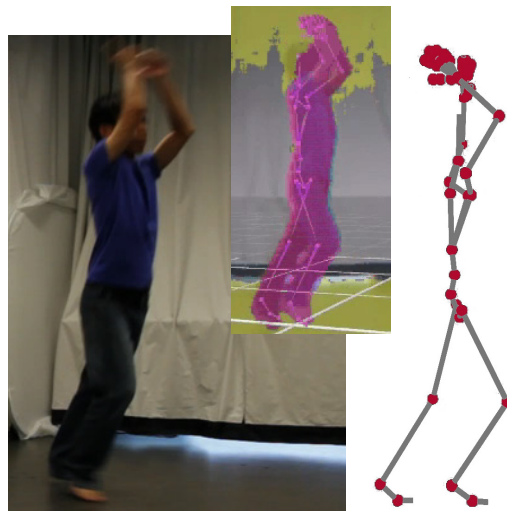


Figure 1.6: Commercial motion sensing input devices such as the Microsoft Kinect enable direct capturing of human motions.

components to ensure the creation of working physical prototypes.

## 1.2 Contributions

The principal contributions of this dissertation are summarized below.

- We present an image-based 3D reconstruction framework for urban scenes that integrates structure discovery and geometry consolidation. The key aspect of this framework is a coupled optimization that combines low-level geometric feature extraction in the form of line features with symmetry detection.
- We develop an algorithm to extract repeated elements in images of buildings while simultaneously computing the camera parameters corresponding to each image. Explicit detection of repetitions globally disambiguates the correspondence search across the images.
- We present an algorithm for detecting structured variations among the elements of a building. Such variations reveal which elements are exactly repeated, or how elements can be derived as structured variations from a common base element.
- We introduce an automated approach for designing and fabricating functional models, specifically mechanical figures that approximate a given target motion.

## 1.3 Organization

The remainder of this dissertation is organized as follows:

**Chapter 2, 3D Urban Modeling Revisited.** This chapter provides an extensive overview of state-of-the art techniques proposed to enable fast and accurate 3D reconstruction of urban spaces. We also introduce the key steps of a traditional image-based reconstruction pipeline, underlining the specific challenges encountered at each step. Finally, we provide a brief introduction on the notion of symmetry and how it can be utilized to overcome these challenges.

**Chapter 3, Factored Acquisition of Buildings.** This chapter covers an image-based 3D reconstruction framework for piecewise-planar buildings containing symmetric parts. Given a set of input images, together with the corresponding camera parameters, this framework utilizes geometric priors in the form of line and plane features to capture local spatial coherence in the data. We exploit large scale symmetries among the elements of the building, e.g. window frames, to provide structural priors that explore non-local coherence. Our reconstructions provide a factored representation where the individual building elements are nicely encoded. We provide evaluations performed on challenging data sets, both synthetic and real.

**Chapter 4, Symmetry and Structure-from-Motion.** Repeated structures are ubiquitous in buildings leading to ambiguity in establishing correspondences across sets of unordered images.

## Chapter 1. Introduction

---

This chapter presents a coupled approach to resolve such ambiguities by explicit detection of the repeating structures. We show that this approach simultaneously computes accurate camera parameters corresponding to each image and recovers the repetition pattern both in 2D and 3D. We evaluate the robustness of the proposed scheme on a variety of examples and provide comparisons with other structure-from-motion methods. We also show that the recovered repetitions patterns enable a range of novel image editing operations that maintain consistency across the images.

**Chapter 5, Understanding Structured Variations.** Many architectural data sets not only contain elements that are exact replicas of the same geometry, but also consist of elements that exhibit variations of a base geometry, e.g. windows with similar arch but varying height. In this chapter, we investigate the problem of understanding such variations in the context of *multi-view stereo reconstructions* of ornate historic buildings. Utilizing a database of template models, each equipped with a deformation model, we detect patterns across element instances of a building by matching them to templates and extracting similarities in the resulting deformation modes.

**Chapter 6, Designing Functional Models.** In this chapter, we introduce an automated system that takes a motion sequence of a humanoid character and generates the design of a mechanical figure that mimics the input motion. The generated designs consist of parts that can be easily fabricated or obtained. A central goal of our approach is to observe patterns typically occurring in human motions to provide a high-level understanding of the target motions. This understanding is useful both to guide the design process and identify the motion types that are better suitable for our system.

**Chapter 7, Conclusion and Future Directions.** We provide a summary of the dissertation with an emphasis on the take-home messages and suggest future research directions.

## 2 3D Urban Modeling Revisited

With the advances in the acquisition technology, we see an increasing interest in digitizing objects and scenes. 3D acquisition devices, such as the hand-held scanners or LIDAR scanners, enable to capture physical objects ranging from specific human body parts to large cities. The wide applicability of the collected data in various fields (e.g. industrial design, prototyping, prosthetics, entertainment industry) has triggered the development of advanced reconstruction and processing algorithms.

Urban reconstruction is one such field that is still under active research due to its wide spread domain (see Figure 2.1). Digital mapping and navigation systems, such as Google Earth and Microsoft Bing Maps, require 2D or 3D building models. 3D reconstruction of urban areas provide useful interaction metaphors for urban planning and design. Several movies and games rely on 3D digital cities. Virtual urban worlds are also useful for applications including emergency planning and virtual touristic tours. The variety of these potential applications has attracted a significant amount of attention from the researchers. In this chapter, we provide an overview of the various methods proposed for fast and accurate reconstruction of 3D buildings with a particular focus on image-based modeling techniques. We further describe the main steps of a traditional image-based reconstruction pipeline, underline the specific challenges, and outline how we plan to overcome these challenges. Please note that some of the methods we review are general-purpose and thus can be applicable for other problem domains as well.

### 2.1 Overview of Approaches

The modeling of urban spaces has been performed using a variety of different strategies (see Figure 2.2). *Procedural modeling* is one of these approaches which has been mainly utilized for fast generation of complex urban structures from a set of parameters and rules. In one of the early efforts, Wonka et al. [114] use split grammars and an attribute matching system to synthesize buildings with a large variety of different styles. Müller et al. [75] also use the idea of splitting to analyze single images of facades. They combine auto-correlation based analysis

## Chapter 2. 3D Urban Modeling Revisited

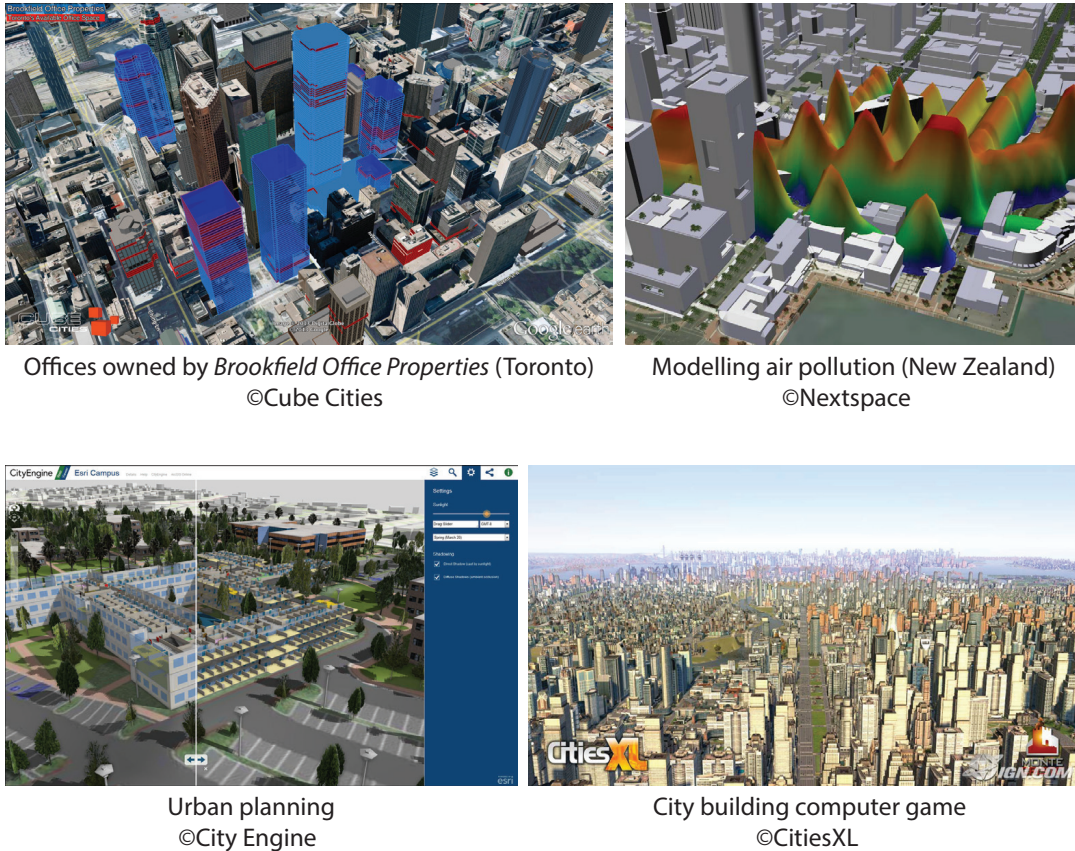


Figure 2.1: Urban reconstruction has a wide spread application domain including urban design, emergency planning and entertainment industry.

of rectified images with shape grammars to generate rules and extract their parameters. In a more recent effort, Kelly and Wonka [50] demonstrate an interactive approach for generating buildings from architectural footprints using procedural extrusions. Even though procedural rules provide an efficient way to create high quality detailed models, they are not useful for direct model acquisition.

To overcome this limitation, *inverse procedural modeling* has recently become a new and growing area. Here, the focus is to discover parameterized grammar rules and the corresponding parameters that can generate a given specific target example. In an early effort, Aliaga et al. [3] present a system to extract a repertoire of grammars from a set of images with user guidance and use this information to quickly generate modifications of the architectural structures. Bokeloh et al. [10] explore partial symmetries in a given model to generate rules that can be used to describe similar models. Given a large set of rules, Talton et al. [105] present an approach to select the rules and the parameter settings that can generate output resembling a high level description of desired productions. A common challenge for these methods is that the expressive power of procedural modeling makes the inverse problem extremely difficult. Thus, obtaining suitable generative



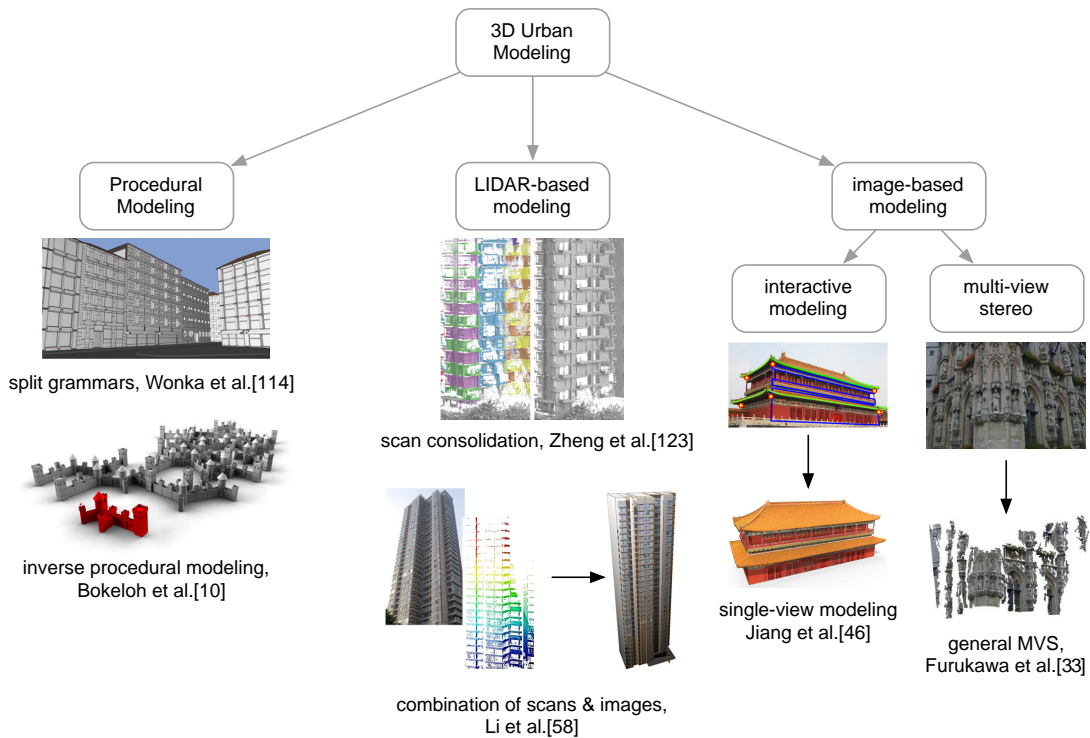


Figure 2.2: The modeling of urban spaces has been performed using a variety of different approaches. (Images courtesy of corresponding authors.)

procedures to capture target shapes still remains as an ambitious problem.

Direct acquisition of urban spaces, on the other hand, is becoming popular due to the variety of the input data sources. An important portion of the proposed 3D reconstruction methods use LIDAR scans. These scans are collected by measuring distance by illuminating a target region with a laser and analyzing the reflected light. This process creates 3D point clouds with significant accuracy, however the challenges in the practical acquisition process often lead to incomplete coverage. Therefore several automatic and interactive methods have been proposed for post-processing of such data.

Zhou and Neumann [126] present an automatic algorithm for creating building models from LIDAR scans. This algorithm first performs vegetation detection and then estimates the principle directions of the building roof patches. This information is used to fit parametric models to the data. In a similar effort, Pu and Vosselman [89] present an automatic method for reconstruction of building facade models from terrestrial laser scanning data by fitting polygonal models. The method first detects planar features in the input point clouds corresponding to wall, door, or window regions. A concave or convex polygonal model is then generated from these features. In a recent effort, Vanegas and colleagues [108] propose an approach for extracting volume descriptions from 3D point clouds based on the *Manhattan World* assumption, i.e. the presence of three mutually orthogonal directions in the scene.

In addition to fully automatic methods, several algorithms that utilize user interaction have been proposed for processing of scan data. Nan et al. [77] propose an interactive method where the users roughly define simple building blocks, called *smart boxes*, over the 3D point samples. The algorithm then snaps these boxes properly to the data by accounting both for data fitting and inter-box similarity. In a similar data consolidation framework, Zheng et al. [123] exploit large-scale repetitions to denoise the input data and complete missing parts. These repetitions are detected with the help of the user.

There are a multitude of urban reconstruction methods that combine LIDAR scans with images. Liu and Stamos [61] present a system that registers 2D images with 3D point clouds. By matching linear features in the images and the point clouds, the method aims to robustly register the camera parameters of the images to the 3D data. More recently, Li et al. [58] present an interactive system for combining information from images and LIDAR scans. They create a layered representation of input buildings which enables more robust detection of repeating structures that are used to enhance 3D data.

Although scanning devices are frequently used by land surveying offices, they are still not available for mass markets due to economical and practical reasons. On the other hand, the advances in the camera technology and the simplicity of the acquisition process has recently made images one of the obvious input sources. It is estimated that tens of billions of photos are taken each year, many of which depict urban sites [76]. As a result, image-based modeling methods has recently become one of the most popular 3D acquisition techniques. In the next section, we provide an overview of different approaches proposed using images as input.

## 2.2 Image-based Modeling

An important category of image-based methods rely on interactive modeling using single or multiple images as input. A seminal work in the field of multi-view interactive modeling is the system presented by Debevec et al. [23]. This system uses manually marked lines in photographs to fit parameterized polyhedral shapes to represent the input model. In a similar system, Liebowitz et al. [59] explore additional constraints such as parallelism and orthogonality of line features. In a more recent effort, Chen et al. [18] interpret freehand sketches to create texture-mapped 2.5D building models using a database of geometric models to fill in plausible details. Jiang et al. [46] present a single-view modeling system where the user annotates architectural elements of the input building and the symmetry information is used to recover the 3D information.

For interactive modeling systems, often there is a trade-off between quality and scalability. While more user interaction results in high-quality models enriched with semantic information, such approaches do not scale well with large data sets. Therefore, it is critical to automatically extract 3D information from a set of images in order to develop semi-automatic or fully-automatic modeling approaches. To perform this task, we need to understand the process of image formation, the formation of a 2D representation of a 3D-world. This understanding enables us to deduce



the 3D structure of what appears in the images.

### 2.2.1 Camera Model

The 2D representation of a 3D world is a *projection* process where we lose one dimension. The standard way of modeling this process is to use a *pinhole camera* model. A ray from a fixed point in space, representing the center of projection, is drawn to a point in 3D world. This ray will intersect a specific plane, the *image plane*. This intersection point corresponds to the image of the 3D point. This process is shown in Figure 2.3a, where a point in space with coordinates  $\mathbf{X} = (X, Y, Z)^T$  is mapped to a point  $\mathbf{x} = (x, y)^T$  on the image plane. Assuming, the camera centered at point  $\mathbf{C}$  is looking towards the positive  $Z$ -axis, called the *principle axis*, and the  $Y$ -axis denotes the up-direction, the image plane is placed perpendicular to the principal axis. The distance from the image plane to the projection center  $\mathbf{C}$  is called the *focal length* and denoted as  $f$ . The point where the principal axis intersects the image plane is called the *principal point* and is depicted as  $\mathbf{p} = (p_x, p_y)^T$ . By similarity of triangles, we see that the 3D point  $(X, Y, Z)^T$  is mapped to the 2D point  $(x, y) = (fX/Z, fY/Z)^T$ .

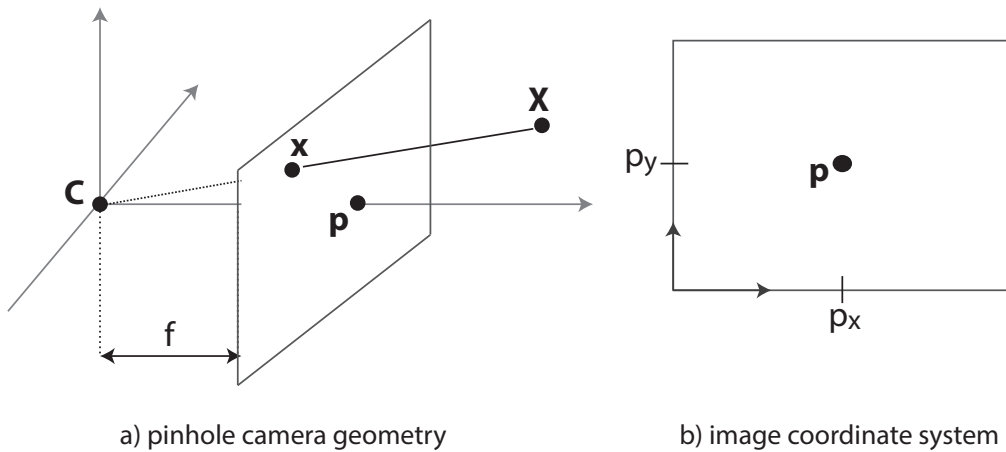


Figure 2.3: *Pinhole camera geometry*. The camera center,  $\mathbf{C}$ , is placed at the origin and the image plane is placed in front of the camera center. The principal point,  $\mathbf{p}$ , is the point where the principal axis meets the image plane.

If we represent the world and the image points in homogeneous coordinates, we can represent the central projection as a linear mapping as follows:

$$\begin{bmatrix} fX & fY & Z \end{bmatrix}^T = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T. \quad (2.1)$$

## Chapter 2. 3D Urban Modeling Revisited

The matrix in this expression is called the *camera projection matrix* and often represented as  $P$ . This expression assumes that the origin of coordinates in the image plane is at the principal point. If this is not the case (see Figure 2.3b), the projection of the 3D point  $\mathbf{X}$  corresponds to the 2D point  $\mathbf{x} = (fX/Z + p_x, fY/Z + p_y)^T$ . Thus, the camera projection matrix can be updated as:

$$P = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (2.2)$$

We can represent this matrix as  $P = K[I | \mathbf{0}]$ .  $K$  is called the *camera calibration matrix* representing the internal parameters of the camera, specifically the focal length and the principal point. In practice, there is the additional possibility of having non-square image pixels, and in rare cases the x- and y-axis of the camera sensor may not be orthogonal. Thus additional parameters can be added to the intrinsic matrix to represent these properties. For the purpose of simplicity, we assume this is not the case.

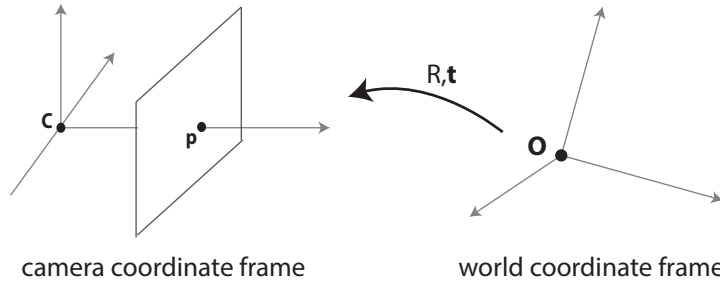


Figure 2.4: The transformation between the camera and the world coordinate frames.

Until now, we have assumed the camera to be placed at the origin of a Euclidean coordinate system and thus the 3D point  $\mathbf{X}$  is in fact represented in this *camera coordinate frame*. In practice, the center of projection and the 3D points are expressed with respect to a different coordinate frame, namely the *world coordinate frame*. The camera and the world coordinate frames are related by a rotation  $R$  and a translation  $\mathbf{t}$  (see Figure 2.4). Thus, the 3D point  $\mathbf{X}$  represented in world coordinates can be expressed as  $\mathbf{X}_{cam} = R\mathbf{X} + \mathbf{t}$  in camera coordinates:

$$\mathbf{X}_{cam} = \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T. \quad (2.3)$$

Substituting this in the previous projection equation, we obtain  $\mathbf{x} = K[R | \mathbf{t}]\mathbf{X}$ . Thus, the camera projection matrix is represented as  $P = K[R | \mathbf{t}]$ . The parameters  $R$  and  $\mathbf{t}$  which relate the orientation and the position of the camera to the world coordinate frame are called the *external*

camera parameters. A 3D world point  $\mathbf{X}$  can be expressed in camera coordinates using the position of the center of the camera  $\mathbf{C}$  in world coordinates as well. Specifically,  $\mathbf{X}_{cam} = R(\mathbf{X} - \mathbf{C})$ . Thus the camera projection matrix can also be written as  $P = K[R \mid -RC]$ .

### 2.2.2 Epipolar Geometry

Extracting 3D information from a set of input images requires to understand the relation between a pair of views of the same scene. Suppose a 3D point  $\mathbf{X}$  is viewed in two views as  $\mathbf{x}$  and  $\mathbf{x}'$ . As shown in Figure 2.5, the points  $(\mathbf{X}, \mathbf{x}, \mathbf{x}')$  and the camera centers form a plane  $p$ . Assume, we only know the image point  $\mathbf{x}$  and wish to find  $\mathbf{x}'$ . We know that the *unknown* point  $\mathbf{x}'$  lies on the plane  $p$  and thus lies on the intersection of this plane with the image plane. This line is the *epipolar line* corresponding to  $\mathbf{x}$ . Thus, the corresponding point to  $\mathbf{x}$  on the second image can be restricted to this line if the camera parameters for both images are known.

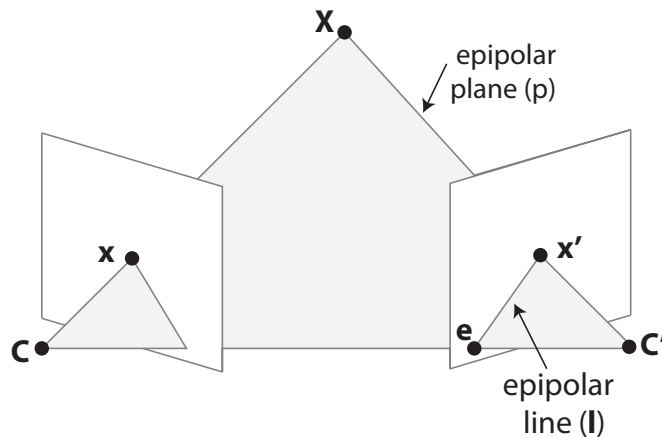


Figure 2.5: The geometric relations of point correspondences between a pair of images.

The epipolar line is the projection of the ray from the first camera center through  $\mathbf{x}$  onto the second image. Assuming the camera projection matrix corresponding to the first image is  $P$ , this ray is obtained by solving  $P\mathbf{X} = \mathbf{x}$ . From the definition of  $P = K[R \mid \mathbf{t}]$  we obtain that the direction of this ray is  $R^T K^{-1}$ . Thus, one-parameter family of solutions for this ray can be written as:

$$\mathbf{X}(\lambda) = \lambda R^T K^{-1} \mathbf{x} + \mathbf{C}, \quad (2.4)$$

where  $\mathbf{C}$  is the center of the first camera, i.e.  $P\mathbf{C} = \mathbf{0}$ , and the ray is parameterized by  $\lambda$ . Assuming the camera projection matrix of the second image is  $P' = K'[R' \mid \mathbf{t}']$ , any point on this ray projects

as follows:

$$P'(\lambda R^T K^{-1} \mathbf{x} + \mathbf{C}) = \lambda K' R' R^T K^{-1} \mathbf{x} + K'(R' \mathbf{C} + \mathbf{t}') \quad (2.5)$$

The last term  $K'(R' \mathbf{C} + \mathbf{t}')$  is the projection of the first camera center to the second image and can be denoted as  $\mathbf{e}$ . To further simplify the notation we can define  $A = K' R' R^T K^{-1}$  where  $A$  is an invertible  $3 \times 3$  matrix. Therefore, for a given point  $\mathbf{x}$  in the first image, the corresponding point in the second image lies on the line through  $\mathbf{e}$  and  $A\mathbf{x}$ . This epipolar line is represented as  $\mathbf{l} = [\mathbf{e}]_{\times} A\mathbf{x} = F\mathbf{x}$ , where  $[\mathbf{e}]_{\times}$  represents the cross product with  $\mathbf{e}$ . The matrix  $F = [\mathbf{e}]_{\times} A$  representing the epipolar geometry is called the *fundamental matrix*.

We know that if points  $\mathbf{x}$  and  $\mathbf{x}'$  correspond, then  $\mathbf{x}'$  lies on the epipolar line  $\mathbf{l} = F\mathbf{x}$ . Being on this line,  $\mathbf{x}'$  satisfies the following equality,  $\mathbf{x}'^T \mathbf{l} = 0$ . Thus,  $\mathbf{x}'^T F\mathbf{x} = 0$ . This is an important relation enabling to characterize the fundamental matrix only in terms of corresponding image points. Each pair of corresponding points yields one homogenous linear equation in the entries of the fundamental matrix. Thus, knowing at least *eight* correspondences between a pair of images enables to recover the fundamental matrix up to a non-zero scalar factor (see [45] for details).

### 2.2.3 Recovering Camera Parameters

In the previous section, we have seen that the fundamental matrix relating two images can be recovered from a set of image correspondences. An interesting reformulation of this analysis results in the following problem: Given a pair of images of a static scene with a set of image correspondences, how can we determine the position and orientation of each camera and the 3D world coordinate of each point for every pair of correspondences? We will start our discussion by first analyzing this problem in a two-view setup and then extend our findings to multiple views.

Given a 3D point  $\mathbf{X}$  and two images with camera projection matrices  $P_1$  and  $P_2$ ,  $\mathbf{X}$  projects to the images as follows:

$$\begin{aligned} \lambda_1 \mathbf{x}_1 &= P\mathbf{X} = K_1 R_1 (\mathbf{X} - \mathbf{C}_1) \\ \lambda_2 \mathbf{x}_2 &= P\mathbf{X} = K_2 R_2 (\mathbf{X} - \mathbf{C}_2), \end{aligned} \quad (2.6)$$

where  $\mathbf{x}_1 = [x_1 y_1 1]^T$  and  $\mathbf{x}_2 = [x_2 y_2 1]^T$  represent the image projection points and  $\lambda_1$  and  $\lambda_2$  represent the projective depth of  $\mathbf{X}$  with respect to the first and second cameras. Given a set of correspondences  $(\mathbf{x}_1, \mathbf{x}_2)$ , the goal is to determine the camera intrinsic (i.e.  $K_i$ ) and extrinsic (i.e.

$R_i, \mathbf{C}_i$ ) parameters and the 3D location of the correspondences (i.e.  $\mathbf{X}_j$ ).

Without loss of generality, we may introduce a variable  $\mathbf{X}'$  to replace the expression  $\mathbf{X}' = R_1(\mathbf{X} - \mathbf{C}_1)$  which represents a Euclidean transformation. This means,  $\mathbf{X} = R_1^T \mathbf{X}' + \mathbf{C}_1$ . Substituting the expressions for  $\mathbf{X}$  and  $\mathbf{X}'$  into the projection equations (Equation 2.6), we obtain:

$$\begin{aligned}\lambda_1 \mathbf{x}_1 &= K_1 \mathbf{X}' \\ \lambda_2 \mathbf{x}_2 &= K_2 R_2 R_1^T \mathbf{X}' + K_2 R_2 (\mathbf{C}_1 - \mathbf{C}_2),\end{aligned}\tag{2.7}$$

The term  $K_2 R_2 (\mathbf{C}_1 - \mathbf{C}_2) = \lambda_{e_2} \mathbf{e}_2$  in fact represents the projection of the first camera in the second image where as  $\lambda_{e_2}$  is the projective depth of  $\mathbf{C}_1$  in the second camera. With a sufficient number of correspondences, we can compute the fundamental matrix  $F$  of the image pair up to a scale factor. Thus we can compute  $\mathbf{e}_2$  as it is the right null vector of  $F$  (i.e.  $F \mathbf{e}_2 = \mathbf{0}$ ).

We can introduce an additional variable  $\tilde{\mathbf{X}} = \frac{1}{\lambda_{e_2}} \mathbf{X}' = \frac{1}{\lambda_{e_2}} R_1 (\mathbf{X} - \mathbf{C}_1)$ . This variable enables to represent  $\lambda_1$  and  $\lambda_2$  relative to  $\lambda_{e_2}$  and we obtain:

$$\begin{aligned}\tilde{\lambda}_1 \mathbf{x}_1 &= K_1 \tilde{\mathbf{X}} \\ \tilde{\lambda}_2 \mathbf{x}_2 &= K_2 R_2 R_1^T \tilde{\mathbf{X}} + \mathbf{e}_2,\end{aligned}\tag{2.8}$$

where  $\tilde{\lambda}_1 = \frac{\lambda_1}{\lambda_{e_2}}$  and  $\tilde{\lambda}_2 = \frac{\lambda_2}{\lambda_{e_2}}$ . We finally introduce the variable  $\tilde{\tilde{\mathbf{X}}} = K_1 \tilde{\mathbf{X}}$  and thus have  $\tilde{\tilde{\mathbf{X}}} = K_1^{-1} \tilde{\mathbf{X}}$ . This gives us,

$$\begin{aligned}\tilde{\lambda}_1 \mathbf{x}_1 &= \tilde{\tilde{\mathbf{X}}} \\ \tilde{\lambda}_2 \mathbf{x}_2 &= K_2 R_2 R_1^T K_1^{-1} \tilde{\tilde{\mathbf{X}}} + \mathbf{e}_2.\end{aligned}\tag{2.9}$$

Interestingly, we arrive at the expression  $A = K_2 R_2 R_1^T K_1^{-1}$  introduced in the previous section to define the fundamental matrix, i.e.  $F = [\mathbf{e}_2]_{\times} A$ . Unfortunately, the knowledge about  $F$  does not uniquely identify  $A$  creating a reconstruction ambiguity. This ambiguity results from the fact that the fundamental matrix is not changed by a projective transformation in 3D. Assume  $H$  is a matrix

representing such a projective transformation. Since,  $P_1\mathbf{X} = (P_1H)H^{-1}\mathbf{X}$  and  $P_2\mathbf{X} = (P_2H)H^{-1}\mathbf{X}$ , the 3D point  $\mathbf{X}$  and  $H^{-1}\mathbf{X}$  correspond to the same image points for the camera pairs  $(P_1, P_2)$  and  $(P_1H, P_2H)$  respectively. Thus, fundamental matrices corresponding to the camera pairs  $(P_1, P_2)$  and  $(P_1H, P_2H)$  are the same. Therefore, without any additional information, correspondences between an image pair determine a pair of cameras only up to a projective transformation. We refer the readers to the book by Hartley and Zisserman [45] for a more detailed discussion about the projective ambiguity and the additional information required to resolve this ambiguity.

In practice, a standard method to resolve this ambiguity is to obtain information about the intrinsic camera parameters. This information is often extracted from the Exif tags of the images and include the focal length, the image size, and the camera model.

If the intrinsic camera matrices  $K_1$  and  $K_2$  are known, the fundamental matrix takes a specialized form. Given the projection of a 3D point to an image,  $\mathbf{x} = P\mathbf{X} = K[R|t]\mathbf{X}$ , and the intrinsic camera matrix  $K$ , we can obtain the point  $\bar{\mathbf{x}} = K^{-1}\mathbf{x} = [R|t]\mathbf{X}$ .  $\bar{\mathbf{x}}$  is said to be expressed in *normalized image coordinates*. Another interpretation of the point  $\bar{\mathbf{x}}$  is the projection of  $\mathbf{X}$  to a camera with projection matrix  $P = I[R|t]$ , where the identity matrix represents the intrinsic camera parameters. If we consider a pair of such cameras with the projection matrices  $P_1 = [I|\mathbf{0}]$  and  $P_2 = [R|\mathbf{t}]$ , the previous definition of the fundamental matrix (Equation 2.5) converges to  $F = [\mathbf{t}]_{\times}R$ , and is defined as the *essential matrix*  $E$ . In other words, the essential matrix depends on the relative orientation and position change between the two cameras.

Similar to the fundamental matrix, the essential matrix relates the correspondences between an image pair expressed in normalized image coordinates  $\bar{\mathbf{x}}^T E \bar{\mathbf{x}} = 0$ . Thus, it is possible to compute the essential matrix from a set of image correspondences.

Once the essential matrix is known, it is possible to estimate the relative orientation and position of the cameras of an image pair. Assuming the first camera matrix is  $P_1 = [I|\mathbf{0}]$ , the goal is to estimate the second camera matrix  $P_2$ . From the definition of the essential matrix,  $E = [\mathbf{t}]_{\times}R$ , it is possible to decompose  $E$  into the product of a skew-symmetric and a rotation matrix as  $E = SR$ . The skew-symmetric matrix  $S$  has two non-zero and equal singular values and a third singular value equal to zero [38]. The multiplication with the rotation matrix  $R$  does not change the singular values, thus  $E$  also has two singular values which are equal and one which is zero. Given the singular value decomposition (SVD) of  $E, E = UDV^T$ , we can write the diagonal matrix  $D = \text{diag}(s, s, 0)$  where  $s$  denotes the nonzero singular value of  $E$ .

Introducing the orthogonal matrix  $W$  and the skew-symmetric matrix  $Z$ ,

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.10)$$

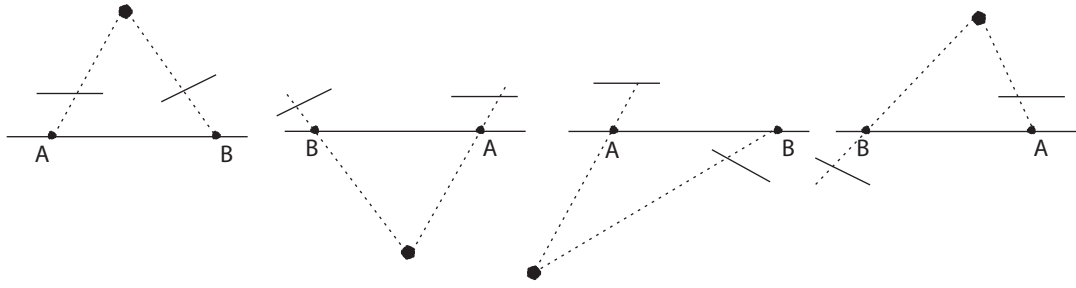


Figure 2.6: The four possible solutions for the camera position and orientations recovered from their essential matrix. Only in the first configuration, the reconstructed point is in front of both cameras.

$S$  can be written as  $S = UZU^T$  up to scale. We can show that this is true since  $S^T = UZ^T U^T = -U^T ZU = -S$ , a property of skew-symmetric matrices. Since the expressions  $E = SR$  and  $E = UDV^T$  should be equal, we obtain  $R$ :

$$E = UDV^T = SR = (UZU^T)(UXV^T) = U(ZX)V^T. \quad (2.11)$$

Thus we have  $R = UXV^T$ .  $ZX$  must be equal to the diagonal matrix  $D$  and since  $X$  should be a rotation matrix, we obtain that  $X = W$  or  $X = W^T$ . This factorization enables us to write  $S = [\mathbf{t}]_{\times}$  and since  $S\mathbf{t} = 0$ . In other words,  $UZU^T\mathbf{t} = 0$  and thus we can define  $\mathbf{t}$  as the third column of  $U$ :  $\mathbf{t} = \mathbf{u}_3$ . However, we cannot determine the sign of  $\mathbf{t}$ .

Thus, given the essential matrix  $E$  corresponding to the camera projection matrices  $P_1 = [I|\mathbf{0}]$  and  $P_2 = [R|\mathbf{t}]$ , there are four possible choices for  $P_2$ :

$$P_2 = [UWV^T|\mathbf{u}_3] \text{ or } P_2 = [UWV^T|-\mathbf{u}_3] \text{ or } P_2 = [UW^T V^T|\mathbf{u}_3] \text{ or } P_2 = [UW^T V^T|-\mathbf{u}_3]. \quad (2.12)$$

The possible solutions for  $\mathbf{t}$  denote that the direction of translation between the cameras is reversed. The possible solutions for  $R$  denote a rotation of  $180^\circ$  about the line joining the camera centers (see Figure 2.6). However, in only one of these configurations the reconstructed point will be in front of both of the cameras. Thus, testing with a single point is sufficient to determine the correct configuration.

### 2.2.4 Triangulation

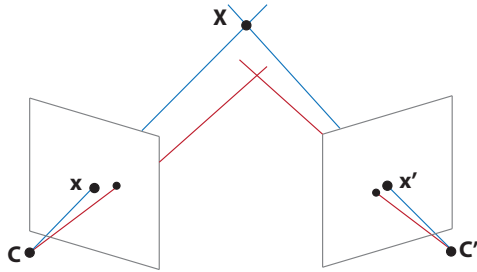


Figure 2.7: In the ideal case, the rays projected through corresponding points intersect resulting in a 3D point (shown in blue). In practice, these rays do not necessarily intersect (shown in red).

Once the camera projection matrices are recovered for input views, a point in 3D space can be computed given its matching correspondences. As seen in the inset figure, in case the corresponding points satisfy the epipolar constraints, the rays passing through the camera centers and the image points ( $\mathbf{x}, \mathbf{x}'$ ) will intersect and thus the 3D intersection point can be determined in a straightforward way. In reality, however, due to inaccurate measurements, the corresponding points may not satisfy the epipolar constraints. As a result, the projected rays do not necessarily intersect. In this case, the goal is to find the 3D point minimizing an error measure relating the 3D point to the correspondences. One choice for such an error

measure is the distance from the 3D point to the projected rays. It turns out that, this error measure results in a 3D point which is the midpoint of the shortest line segment joining the projection rays.

### 2.2.5 Structure-from-Motion

In the previous sections, we have described how correspondences between an image pair can be used to obtain information about the relative position and orientation of the cameras. Establishing reliable correspondences is a critical requirement to perform this task robustly. An important invention that has advanced this task into a hot research topic has been the robust feature-point detection algorithms (e.g. SIFT [64]). These algorithms have enabled efficient detection and matching of feature points across multiple views of a scene. Once established, such correspondences are used to register multiple images by extending the analysis provided in the previous section to multiple images. This process is called *structure-from-motion (SfM)* since 3D information about the scene is obtained by recovering the motion of the cameras.

A typical SfM pipeline starts by detecting and matching a sparse set of feature points across the given images. Such correspondences are used to obtain initial estimates for the camera parameters. By triangulating the detected correspondences, 3D points representing these correspondences are also computed. In practice, image measurements are noisy and thus the projection equations (i.e. Equation 2.6) are not satisfied exactly. Therefore, the goal is to refine the initial estimates of the camera projection matrices  $P^i$  and 3D point locations  $\mathbf{X}_j$  that minimize the image distances between the reprojected points and the initially detected feature points  $\mathbf{x}_j^i$  ( $j$ -th feature point as seen by the  $i$ -th camera):



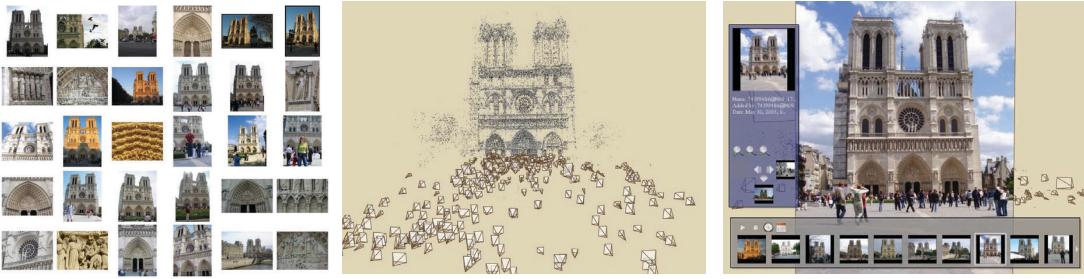


Figure 2.8: *Photo Tourism* system developed by Snavely et al. [103] is a seminal work in using large image collections from the Internet in a SfM pipeline. (Image courtesy of Snavely et al.)

$$\min_{P^i, X_j} \sum \|P^i X_j - x_j^i\| \quad (2.13)$$

This minimization problem is known as *bundle adjustment* as it involves adjusting the rays between the camera centers and the set of 3D points. This is a nonlinear optimization problem for which advanced solvers exist ([63, 116]). Often, SfM is performed iteratively, starting with an initial pair of images and adding new images to the system one by one. After several images are added, bundle adjustment is performed to refine the current estimates of the camera parameters and the 3D points.

A multitude of SfM algorithms have been developed in recent years. An important category of such algorithms operate on ordered image sequences such as video ([35, 87]). The known order of the images provides strong cues to determine candidate images that should be matched. The growing ability of large numbers of images of touristic places from the Internet has inspired researchers to develop SfM methods for large image collections ([103, 37, 1]). These methods utilize the SfM information for exploring and navigating through the captured environment. In a recent effort, Agarwal et al. [2] present a framework that reconstructs architectural places from over hundred thousand images. The problem of matching images is formulated as a graph problem where images that only depict the same object are connected. Object level information is obtained through multi-view clustering of scene objects [31].

Other methods have been proposed that focus on increasing the accuracy of the SfM pipeline. Govindu [40] uses the redundancy in the pairwise image relations to average multiple observations to produce a globally consistent motion estimation and later [41] randomly samples spanning trees from a graph encoding the image relations to prune out mismatches. Martinec and Pajdla [67] incrementally remove high-residual matches to increase robustness. Klopschitz et al. [54] propose an incremental framework that favors subsets of images with highest local connectivity.

SfM can be performed using other type of correspondences such as line features [95]. Micusik et al. [69] use rectangular structures in two-view matching as an alternative to feature points. These types of features are especially suitable for urban environments.

### 2.2.6 Multi-view Stereo

A SfM pipeline recovers the camera parameters of a set of input images together with a 3D representation of the scene. However, since only a sparse set of feature points are used to generate 3D samples, the resulting reconstructions are sparse and do not contain solid geometry. On the other hand, having recovered the camera projection matrices of the images, it is possible to match and triangulate any image pixel resulting in *dense reconstructions*.

Many dense reconstruction methods, called *multi-view stereo (MVS) methods*, use multiple images as input. Several successful MVS algorithms have been developed in recent years [98]. One common approach is to perform dense matching between pairs of images and generate a depth map for each image that encodes the projective depth of the matched points. The resulting depth maps are then combined in a process called *depth map fusion*. Several strategies have been proposed on how to fuse multiple depth maps together [36, 37, 29]. A different approach is based on matching feature points representing small patches on the surface of the target object. Such recovered surface patches are then used to propagate information and perform a denser matching across the input images [33].

Most general purpose MVS algorithms suffer from high noise levels along flat surfaces due to independent matching of points. Use of certain priors have been proposed in the context of urban reconstruction to overcome this limitation. One such prior is the assumption that most objects in the scene consist of piecewise planar elements. Micusik et al. [74] employ a super-pixel segmentation approach where each super pixel is labeled with a candidate plane. Based on the even stronger assumption that all planes are axis-aligned (Manhattan-world), Furukawa et al. [30] propose a method to generate depth maps for each input image based on matching each pixel to a candidate plane having one of the dominant plane orientations. The candidate planes are generated from an initial sparse point cloud. This method has later been extended to model building interiors as well [32]. Recently, Wan et al. [111] proposed a framework to reconstruct piecewise-planar buildings that incorporates constraints based on the relations between the facades of a building.

In addition to dense reconstruction methods, another common approach to obtain solid geometry from sparse SfM reconstructions is to explore user interaction. To illustrate, Sinha et al. [101] present an interactive system to generate textured piecewise-planar 3D building models. The 2D outline sketches provided on the input images are combined with the initial 3D information provided by SfM in an initial step. Similarly, Xiao et al. [120] use the output of SfM to decompose a building facade using horizontal and vertical splitting lines. Each resulting rectilinear structure is then assigned a depth.

### 2.2.7 Challenges

3D reconstruction of geometric models from a set of images is an easy, flexible, and economic method. We have reviewed a multitude of such methods in the previous section. Recent advances in the camera technology have led to significant improvements in the quality of the reconstructed models. Despite this success, various challenges still remain unsolved in the acquisition and reconstruction of clean and precise 3D models.



Figure 2.9: Several factors such as variation in illumination across images, reflective or textureless surfaces, large occlusions, and degrading resolution for upper floors of buildings make the correspondence problem very challenging.

Establishing reliable correspondences between the input images is a key step for both recovering the camera parameters relating a set of input images and computing dense reconstructions. The correspondence problem, however, is a difficult and often ambiguous problem. Several practical factors make this problem even more challenging (see Figure 2.9). For example, large illumination changes across the input images makes it difficult to robustly match feature points. Reflective surfaces contain a significant amount of outlier features coming from the reflections of the surrounding elements. Large textureless surfaces such as walls of a building avoid detection of reliable feature points. Street-level images of buildings often contain large occlusions such as trees and suffer from degradation of resolution for the upper floors of the building. All of these factors avoid establishing robust correspondences and thus lead to high noise levels and missing data in both sparse and dense reconstructions. These artifacts become especially visible when the output reconstructions are viewed to reveal the direction of the projective rays for the 3D points towards the camera centers as seen in Figure 2.10. This comes as no surprise since lifting a 2D image point to 3D is simply the process of estimating the depth of the point along its projective ray and thus wrong correspondences result in wrong depth estimations along this ray.

General image-based reconstruction methods, such as MVS reconstructions, often produce dense point clouds or polygonal meshes. A multitude of geometry processing algorithms have been proposed for manipulating both types of representations [11, 82]. On the downside, however, such low level representations do not capture any higher knowledge of the input scene. A representation limited to low-level geometric primitives, such as points and triangles, makes it difficult to interact with the underlying geometry. Architectural data sets, on the other hand, are very rich in structural and semantic relations. Alongside the dominant plane and line features,

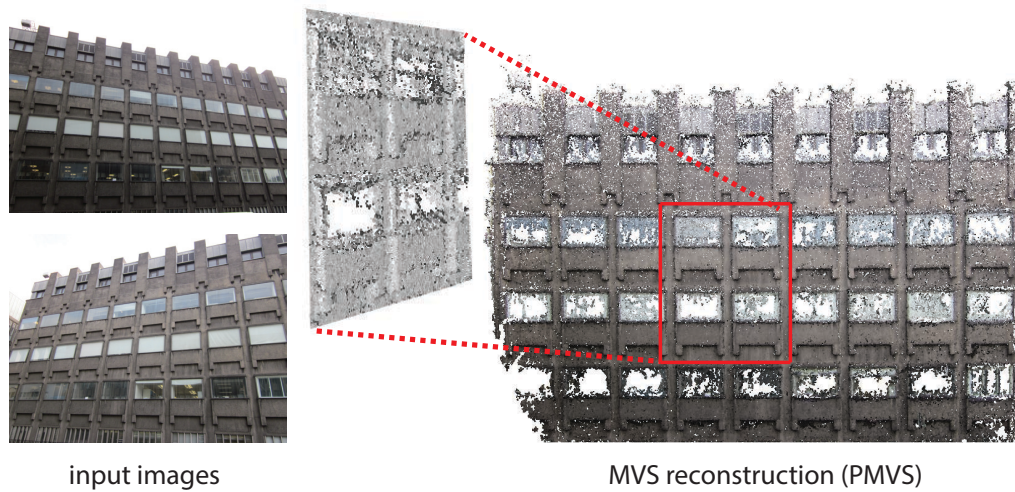


Figure 2.10: Even though MVS methods such as PMVS [33] output impressive results, they still suffer from high noise levels, specially along the direction of projective rays.

they often exhibit symmetric relations among their individual elements such as windows, columns, and arches etc. (see Figure 2.11). Use of repeating structures is reinforced to ease construction and provide aesthetics. Therefore, it is extremely important to explore such relations to enable a better understanding and processing of the captured scenes. Thus, finding symmetries in acquired geometric data is an important problem in geometry processing.

Symmetry is a general concept in mathematics that preserves certain properties of an object under some predefined operation [112]. The *group theory* formalizes this notion of invariance [92]. In the context of geometry, a symmetry relation is defined for a subset  $M$  of a shape  $S$  if there exists a transformation  $T$  (e.g. translation, rotation, reflection) that leave  $M$  invariant under the action of the transformation, i.e.  $M = T(M)$ . In case of regular repetitions, a transformation  $T$  produces replicated copies of  $M$ , i.e.  $T(M), T^2(M), \dots, T^n(M)$ .



Figure 2.11: Dominant repetitions in urban data sets and provide important structural priors to augment the reconstruction process.

There has been a significant amount of research effort to detect symmetries and regular patterns in both 2D and 3D data sets [62, 72]. The proposed methods for symmetry detection in 3D models share a common pipeline at an abstract level. Often, processing begins with feature selection

to restrict the computations to the relevant geometric features of the data set. Then, candidate transformations mapping subsets of selected features are generated. Finally, the local observations from these candidate transformations are accumulated to extract symmetries at larger scales. Among the main approaches for 3D symmetry detection, we can list the transformation space voting schemes of Mitra et al. [71] and Pauly et al. [84], the graph based approach of Bokeloh et al. [9], and the symmetry-factored encoding of Lipman et al. [60].

Once detected, numerous applications benefit from the extracted symmetry information. 3D reconstruction and analysis of architectural data sets are among such applications. In architectural data sets, each replicated copy of an element provides multiple observations of the same geometric piece. Combining these observations in a reconstruction framework can tremendously improve the reconstruction quality. Symmetry also plays a crucial role in creating semantic information of the underlying geometry. Most of us see a set of windows arranged in a grid-like structure when we look at a typical building facade. Explicit detection of such knowledge enables to develop intuitive interaction metaphors to perform editing tasks both in the image and the model space.

Due to these numerous benefits, we have witnessed several efforts in the last few years in bringing symmetry into the 3D reconstruction pipeline. Wu et al. [119] demonstrate that repetitive structures can be used for dense reconstruction from a single image by directly enforcing depth consistency between these structures. Similarly, Jiang et al. [46] exploit symmetry to enable interactive modeling from single images. Zhang et al. [123] consolidate given LIDAR scans of buildings by harnessing repetitions.

A key component of the algorithms we propose, in contrast to related work, is that instead of exploring symmetry priors as a post-processing tool to consolidate initial 3D reconstructions, we handle symmetry detection and reconstruction tasks in a coupled manner. The necessity of a coupled approach is due to the cyclic dependency between these two tasks. Compared to scan data, i.e. LIDAR scans, image-based reconstruction methods often produce more noisy and partial data measurements. We desire to find reliable symmetries to reduce noise and fill holes in these reconstructions. On the other hand, reliable symmetry detection often requires clean and complete data. A coupled approach breaks this cyclic dependency by simultaneously detecting symmetries and reconstructing 3D geometry. We investigate the benefits of this coupled analysis and reconstruction strategy through at each stage of the image-based reconstruction pipeline. We demonstrate that the coupling enables reliable detection of symmetries while increasing the reconstruction quality.

A common practice in the algorithms we will present is to exploit both input images and intermediate 3D reconstructions as complementary data sources. While images are better suited for capturing high resolution details of the underlying geometry they lack depth information. Intermediate 3D reconstructions, on the other hand, play a crucial role in accumulating observations across multiple images. Our algorithms focus on combining the advantages of each type of data source into a unified framework.





## 3 Factored Acquisition of Buildings

A typical image based 3D reconstruction framework consists of two main phases as explained in the previous chapter. Once the camera parameters relating the input images are computed in the structure-from-motion (SfM) step, dense image correspondences are triangulated to generate dense reconstructions as shown in Figure 3.1. Typically, such correspondences are obtained by local feature or window-based matching algorithms. Such local processing fails to obtain reliable correspondences under challenging conditions (e.g. reflections, occlusions, change in illumination) and leads to high noise-levels.

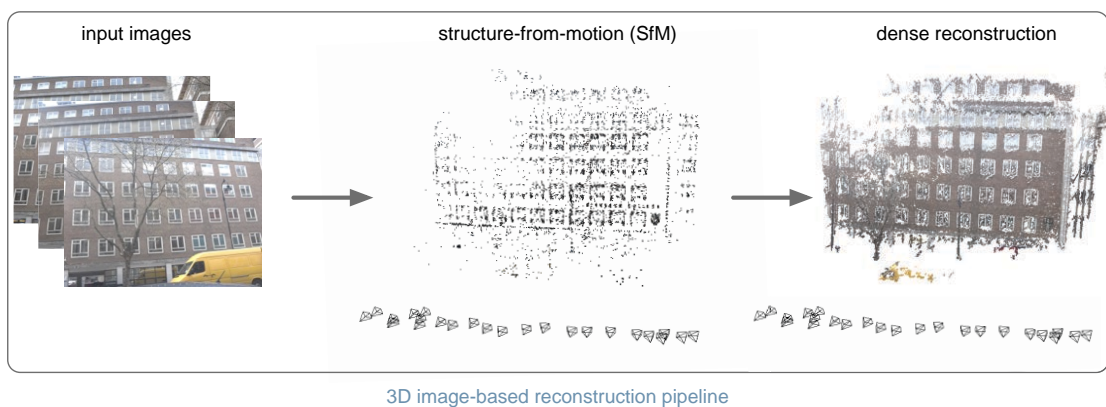


Figure 3.1: In a typical image-based 3D reconstruction pipeline, the structure-from-motion step is followed by a dense reconstruction of the captured scene.

Structural priors, such as symmetries, when available as in the case of urban scenes, enable to overcome some of these challenges in image-based 3D reconstruction and improve the output quality. A potential solution is to first extract structural information from the given SfM or dense reconstructions and then utilize this information to regularize the output. Such a decoupled approach, however, can fail since the low-quality 3D geometry makes robust structure detection difficult. Therefore, we face a cyclic dependency: to remove noise and outliers and fill holes, we desire to find reliable symmetries; yet to robustly estimate symmetries, we need clean and

## Chapter 3. Factored Acquisition of Buildings

---

complete datasets. Our goal is to avoid this dependency by integrating the reconstruction and structure detection tasks into a common framework and explore the benefits of structural priors in each stage of the 3D reconstruction pipeline. We start our discussion by focusing on the dense reconstruction stage and turn our focus to the SfM stage in the next chapter.

Given a set of input images of a building together with their camera parameters, our goal is to extract and explore structural information to produce high-quality 3D models. We make an important distinction between two types of structural information: (i) geometric primitives such as lines and planes enable to capture small and medium scale spatial coherence in the data, (ii) translational repetitions provide reconstruction priors that exploit non-local coherence. Both of these priors explicitly capture the dominant symmetries of the acquired building. Line and plane features exploit continuous symmetries, i.e. they contain infinitely many partial symmetries with a continuum of transformations. Repetitive elements, on the other hand, model discrete symmetries.

We recall that symmetry priors are neither given a priori nor explored as a post-processing tool. Instead, our goal is to directly learn them during the reconstruction process. We achieve this goal by simultaneously operating on the input images and the intermediate 3D reconstructions. We formulate a combined reconstruction-detection algorithm that iteratively propagates geometric and structural information to reinforce symmetries and 3D sample locations as will be described next.

### 3.1 Overview

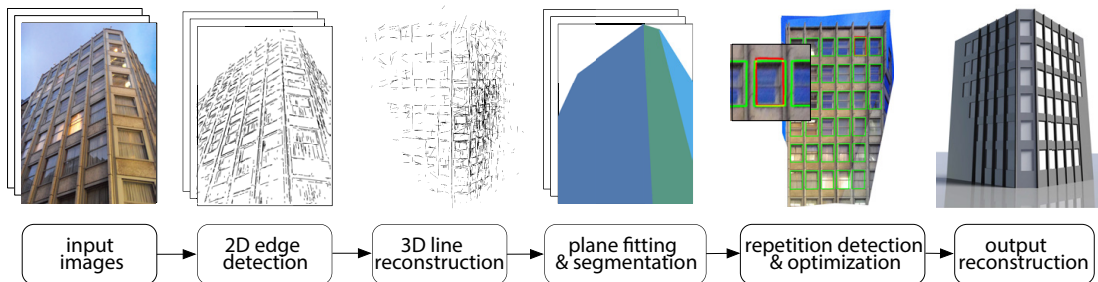


Figure 3.2: We propose a reconstruction pipeline where we explore structural priors at two levels: while line and plane features exploit continuous symmetries, repetitive elements model discrete symmetries.

We propose an image-based dense reconstruction pipeline for architectural datasets that exploits continuous symmetries in the form of lines and planes and discrete symmetries in the form of repeating elements (see Figure 3.2) [16]. Given a set of registered input images of a building, we start by performing image-space edge detection on each individual image. These linear features will be used as the basis for planar feature extraction and discrete repetition detection. However, not all the detected 2D edges correspond to relevant 3D line features of the geometry, they contain



many outliers arising due to shadows, occlusions, and reflections (see Figure 3.4). A critical step in our framework is to generate a 3D line feature set by matching and triangulating these 2D edges across multiple images. This step acts as a filter to remove outlier edge features as we aggressively prune out potential mismatching edges. Yet we retain sufficiently many correct 3D lines to create a set of candidate 3D planes that are used to generate intermediate piecewise-planar 3D reconstructions. We segment the input images into planar regions represented by these 3D planes and combine these segments to texture the planes. This helps to perform the discrete symmetry detection directly on the planes instead of the individual images as the planes act as proxies to link the images. Symmetry detection involves accurately extracting the contours of the repeating elements, such as the window frames, and computing the symmetry transform relating these elements. The resulting factored symmetric parts still lie on the corresponding planes and lack depth variations. However, the factored representation enables easy interaction possibilities such as extruding or retracting the elements based on user specified depth offsets.

## 3.2 Algorithm Details

We will next describe each step of the proposed reconstruction pipeline in more detail. The input to the pipeline is a set of input images  $\mathcal{S} = \{I_1, \dots, I_n\}$  and their camera parameters (computed with the Bundler framework [103]). The output is a piecewise-planar 3D model of the captured building with a factored representation of the detected repeating elements.

### 3.2.1 3D Line Reconstruction

A typical building has dominant line features that both reveal the contours of the individual elements, such as window frames, and the intersection of planar regions corresponding to each facade of the building. Therefore, we start our reconstruction process by extracting these linear features. We first detect edge features in each of the input images  $I_i$  by using the standard *Canny edge detection* method to identify the image pixels where an edge passes. Such pixels are later linked into line segments until junction points are reached (see the open-source EdgeLink function<sup>1</sup>). This gives us a collection of 2D edges  $\mathcal{L}^2(I_i)$  for each input image.

As stated previously, the 2D edge collection consists of many outlier features due to reflections, occlusions, and shadows. We use 3D line reconstruction to prune out such outliers as edges are matched across multiple images. Assume we have an edge  $e_k$  detected in image  $I_i$  and would like to find its matching edge in image  $I_j$  as seen in Figure 3.3. The two endpoints of the edge  $e_k$  define two epipolar lines in image  $I_j$ . The region defined by these epipolar lines identifies the candidate matching edges for  $e_k$  since the correct match should intersect this region. To find the correct matching edge, we compute a similarity score for each candidate and pick the edge with the highest similarity as the match. To compute the similarity score between the edge  $e_k$  and a candidate matching edge  $e_m$ , we uniformly sample  $e_k$ . For each sample point, we compute the

<sup>1</sup><http://www.csse.uwa.edu.au/pk/research/matlabfns/LineSegments/edgeline.m>

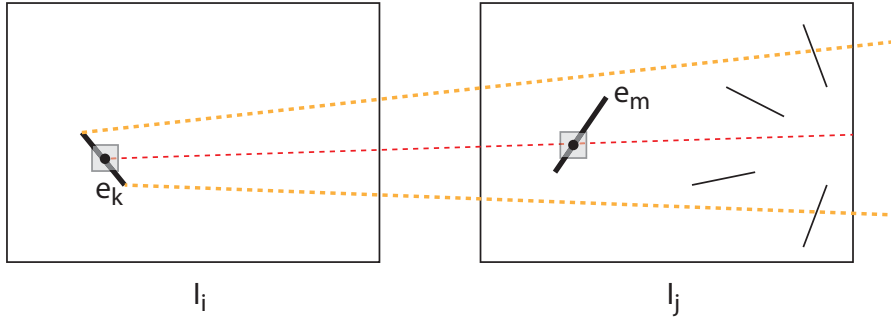
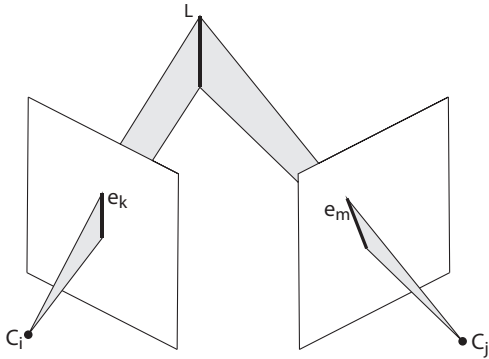


Figure 3.3: Given an edge  $e_k$  in image  $I_i$ , its matching edge in image  $I_j$  falls inside the region defined by the epipolar lines (shown in orange) corresponding to the endpoints of  $e_k$ . For each potential match, a similarity score is computed by comparing small patches (shown in gray) along the matching sample points on  $e_k$  and  $e_m$ .

intersection point between  $e_m$  and the epipolar line in image  $I_j$  corresponding to the sample, if exists. Then, we define small image-patches ( $7 - by - 7$  windows) centered around the sample point and this intersection point and compute the *normalized cross correlation (NCC)* score between the resulting patches. The similarity score between the two line segments is defined to be the average NCC score over all such patches.



Once a set of matching edges are identified across the images, the intersection of the planes passing through these edges and the corresponding camera centers define a 3D line. The inset figure shows this in case of two matching edges. In practice, however, the planes may not intersect exactly and we are looking for the 3D line that minimizes the distance to each of the planes. We define this line with a direction and a point,  $(\mathbf{d}, \mathbf{x})$ . Let us assume each of the matching edges define a

plane with normal  $\mathbf{n}_i$  and a point  $\mathbf{o}_i$ . We select the 3D point  $\mathbf{x}$  as the point that minimizes the total distance to each of the planes:

$$\min_{\mathbf{x}} \sum (\mathbf{n}_i^T (\mathbf{x} - \mathbf{o}_i))^2. \quad (3.1)$$

Taking the derivative of this expression with respect to  $\mathbf{x}$  and equating to zero shows that  $\mathbf{x}$  is the solution to the system  $A\mathbf{x} - b = 0$  where  $A = \sum \mathbf{n}_i \mathbf{n}_i^T$  and  $b = \sum \mathbf{n}_i \mathbf{n}_i^T \mathbf{o}_i$ .

Ideally, we want the direction  $\mathbf{d}$  of the line to be perpendicular to the plane normals  $\mathbf{n}_i$ . In other

words we want to minimize the following energy:

$$\min_{\mathbf{d}} \sum (\mathbf{n}_i^T \mathbf{d})^2 = \mathbf{d}^T A \mathbf{d}. \quad (3.2)$$

Given the SVD decomposition of  $A$ ,  $A = UDV^T = UDU^T$  (since  $A^T = A$ ),  $\mathbf{d}$  that minimizes the above sum is the last column of  $U$  corresponding to the smallest singular value of  $A$ .

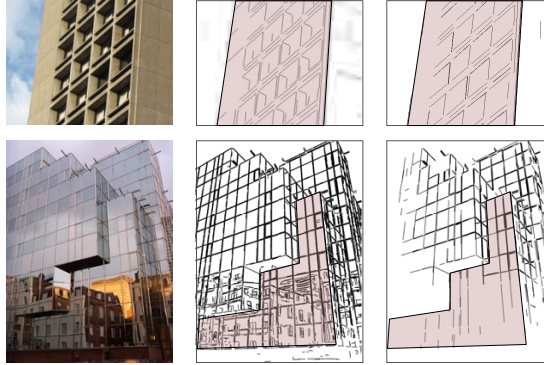


Figure 3.4: Edges detected directly on images often contain outliers from reflections, shadows, occluding elements etc. (middle). We employ matching and triangulation at the edge level to prune out such outliers (right).

During 3D line reconstruction, we aggressively prune out mismatching edges. Specifically, we consider a pair of edges to be matching only if they have a high similarity score (we use a threshold NCC score of 0.7) and generate a 3D line if we collect matching edges in at least 3 images. This aggressive matching acts as a filter to prune out most of the outlier edges and gives us a sparse but more reliable 3D line feature set  $L^3$  as shown in Figure 3.4. We aim to recover the information missed at this step via detected symmetries in the subsequent steps.

### 3.2.2 Piecewise-Planar Model Generation

At a coarse level, buildings can often be considered as piecewise-planar where each facade is represented with a plane. In order to obtain this coarse intermediate representation, we generate a set of candidate planes,  $\mathcal{P}$ , to fit the input geometry using the reconstructed 3D line features. Specifically, we randomly pick a pair of 3D lines  $l_i(\mathbf{v}_1, \mathbf{v}_2)$  and  $l_j(\mathbf{v}_3, \mathbf{v}_4)$  represented with their endpoints and test if they are coplanar as follows:

$$\left( \frac{\mathbf{v}_1 + \mathbf{v}_3}{2} - \frac{\mathbf{v}_2 + \mathbf{v}_4}{2} \right)^T \frac{(\mathbf{v}_1 - \mathbf{v}_3) \times (\mathbf{v}_2 - \mathbf{v}_4)}{\|(\mathbf{v}_1 - \mathbf{v}_3) \times (\mathbf{v}_2 - \mathbf{v}_4)\|} \approx 0, \quad (3.3)$$

i.e., the diagonals of the quadrilateral  $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$  intersect (we also test for the ordering  $\mathbf{v}_4, \mathbf{v}_3$ ). If the lines are coplanar, we compute the corresponding plane normal and the offset,  $\mathbf{n} = (\mathbf{v}_1 - \mathbf{v}_2) \times (\mathbf{v}_3 - \mathbf{v}_4) / \|(\mathbf{v}_1 - \mathbf{v}_2) \times (\mathbf{v}_3 - \mathbf{v}_4)\|$ ,  $d = -\mathbf{n}^T \mathbf{v}_1$  and detect the 3D lines that are inliers for this plane. If the plane has a sufficient amount of inliers (at least 5% of the 3D lines), we add it to the candidate plane set  $\mathcal{P}$  and discard the inlier 3D lines from subsequent planarity tests.

Once a candidate plane set  $P$  is generated, our goal is to fit these planes to the input geometry. We perform this task by segmenting each input image  $I_i$  into planar regions to compute the actual boundaries of the candidate planes. In other words, our goal is to label each pixel of  $I_i$  with the most likely plane in  $\mathcal{P}$ . We formulate this labeling problem by introducing two terms for each pixel  $p \in I_i$ :

(i) *Data term*: The data term is used to measure how well a plane  $\mathcal{P}_l$  fits a pixel  $p_k$ . Obviously, assigning a plane to a pixel defines its 3D position as the intersection of the plane with the ray from the camera center and passing through the pixel. An important observation we make is that the initial 3D line reconstruction helps to distinguish between the pixels for which robust depth estimates can be made. Specifically, pixels that do not lie on the projection of a 3D line are more likely to be found in regions where stereo matching fails, such as reflective surfaces. In order to avoid such pixels from voting for non-robust data terms, we assign a fixed data cost to labeling such a pixel:

$$E_{data}(p_k, \mathcal{P}_j) = e^0, \forall \mathcal{P}_j \in \mathcal{P}, \text{ if } p_k \notin l, \forall l \in \mathcal{L}^{3 \rightarrow 2}. \quad (3.4)$$

For any remaining pixel that lies on the projection of a 3D line, it is straightforward to compute the inconsistency of assigning that pixel to a certain plane. For any such pixel  $p_k \in l$  for some  $l \in \mathcal{L}^{3 \rightarrow 2}$ , we can deduce its 3D position,  $\mathbf{p}'_k$ , from the corresponding 3D line. For each candidate plane label  $\mathcal{P}_j$ , we project this 3D point sample to  $\mathcal{P}_j$  and project it back to the input image to generate a 2D coordinate  $p''_k$ . We define the cost of labeling  $p_k$  with the plane  $\mathcal{P}_j$  based on  $d(p_k, p''_k)$ , the Euclidean distance between  $p_k$  and  $p''_k$ . If  $d(p_k, p''_k)$  is greater than a threshold distance  $r$  (1 % of the maximum image dimension), we consider the plane assignment to be inconsistent with the known 3D position and set a high data cost:

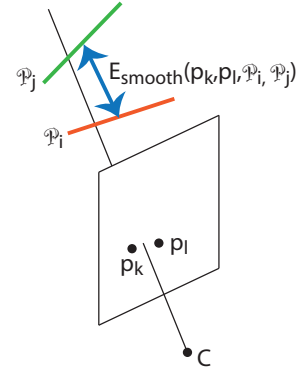
$$E_{data}(p_k, \mathcal{P}_j) = e^{1.0}, \text{ if } d(p_k, p''_k) > r. \quad (3.5)$$

If  $d(p_k, p''_k) \leq r$  on the other hand, we define a multi-view consistency score similar to Sinha et al. [100]. To be specific, we project the 3D point  $\mathbf{p}'_k$  to neighboring views and compute the average NCC matching score,  $s$ , of the local windows centered around  $p_k$  and the projection

pixels in neighboring views. We set the data cost based on this average matching score,  $s$ :

$$E_{data}(p_k, \mathcal{P}_j) = e^{-s}, \text{ if } d(p_k, p_k'') \leq r. \quad (3.6)$$

(ii) *Smoothness Term*: The smoothness term is defined to enforce consistent plane labeling for neighboring pixels as they are likely to be found on same planes. We expect abrupt changes in plane labeling only for pixels that are near intersections of plane pairs from  $\mathcal{P}$ . Thus, we identify the 3D line features that are near plane intersections. For any pair of neighboring pixels  $(p_k, p_l)$  that lie on two sides of the projection of such a 3D line (to handle rasterization artifacts, we work with a small approximation margin of 30 pixels), we set the smoothness cost to 0. For any remaining neighboring pixel pairs  $(p_k, p_l)$  and the candidate plane labels  $(\mathcal{P}_i, \mathcal{P}_j)$ , the smoothness cost is defined as follows. We cast a ray from the camera center passing through the midpoint of the pixels and compute the intersection of this ray with each of the planes  $\mathcal{P}_i$  and  $\mathcal{P}_j$ . As shown in the inset figure, the smoothness cost  $E_{smooth}(p_k, p_l, \mathcal{P}_i, \mathcal{P}_j)$  is defined as the depth difference between the resulting intersection points along the projected ray.



We combine the data and the smoothness terms and solve the pixel-plane labeling problem as a standard *Markov Random Field (MRF)* formulation [55]. Once solved, we obtain a plane assignment for each pixel in each image. However, the segment boundaries obtained may contain noise in regions where depth estimates cannot be made reliably and lack sufficient 3D lines. In order to extract accurate edges across noisy segment boundaries, we use the intersection lines of the neighboring planes in the segmented images (see Figure 4.1). At this stage, if necessary the user can manually make corrections on wrong segment boundaries. These corrections are made by sketching rough strokes on the images which are then snapped to the 2D edges to form the final image segmentation. This new boundary is propagated across the images by obtaining the 3D position of the boundary using the 3D plane assignments.

Once the input images are segmented into planar regions, we generate textures for each 3D plane using these regions. To do so, we first extract the boundaries of a plane  $\mathcal{P}_j$  in 3D by projecting all the pixels in the input images that have been assigned to  $\mathcal{P}_j$ . We assume there exists a dominant up direction and require the user to mark an edge with this orientation in *any* of the images. We compute the up direction in 3D by projecting this edge to its corresponding 3D plane (We note that computing the dominant up direction from the vertical vanishing point is also a possibility [118]). We then define a 2D grid on each plane  $\mathcal{P}_j$  that is oriented to be aligned with this up direction. We can consider this grid to represent the rectified texture image of the plane so the resolution of the grid is selected based on the desired texture resolution. In order to compute the color of each of the cell in this texture image, we project the corresponding 3D cell

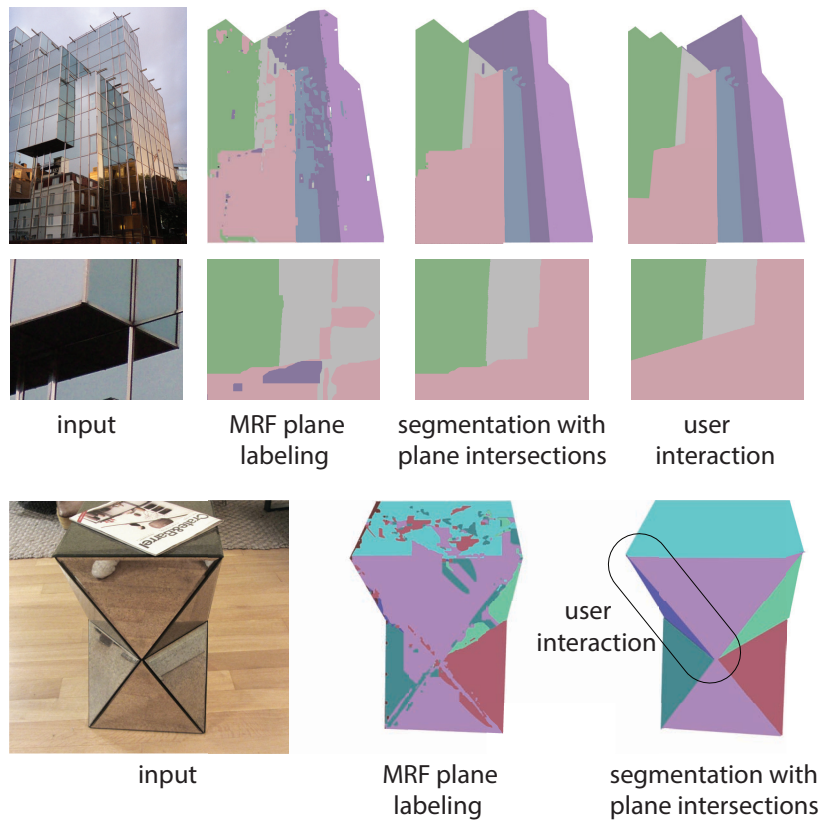
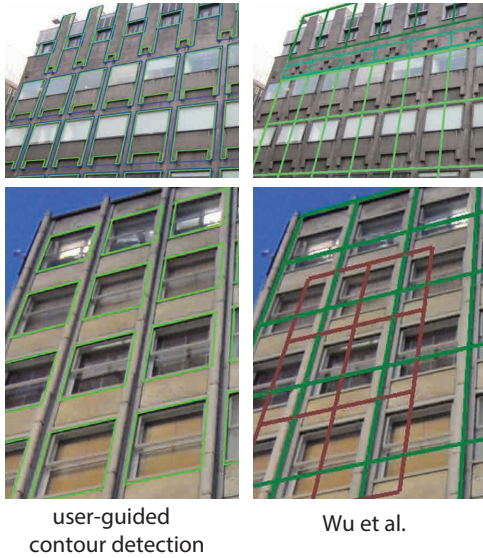


Figure 3.5: We use an MRF-based plane labeling method to segment the input images. The segmentation is cleaned using the intersection lines of neighboring planes. User input resolves regions of insufficient feature lines.

point to every input image it is visible in and simply select the color value from the image that has the least foreshortening factor, i.e. the absolute value of the dot product between the normal of the plane and the image viewing direction is closer to 1. This simple filtering based on the foreshortening factors enables to select the pixel colors from a few input images consistently. However, more advanced texturing methods ([101]) can also be used.

### 3.2.3 Discrete Symmetry Extraction

Once the 3D planes  $\mathcal{P}$  are fitted to the input data by computing their boundaries and textures, we are ready to explore the presence of discrete symmetries. Since the input images are already linked to the 3D planes, we perform the symmetry detection directly on the planes. The rectified texture of the planes eliminates the perspective problem during symmetry detection. Since in most building facades, we observe translational symmetry of elements (e.g. a grid of window frames), we restrict the following discussion to detection of repeating elements that are regularly spaced.



Symmetry detection involves both detecting the base element that is repeating and the transformation relating such repetitions. However, there is often an ambiguity when automatically choosing the repeating base element and its relevant scale. As a human being, on the other hand, when we see a typical building facade we perceive a row of repeating windows. In other words, we tend to assign semantics to the repeating elements. Even though, it is straightforward for the user to quickly scribble the desired base element, it is highly probable for fully automatic methods ([118, 46]) to perform this selection randomly (see the inset figure). Thus, we use a few user-annotated rough strokes denoting elements of interest to search for similar elements across the (rectified) plane images.

Standard image-based similarity measures focusing on pixel intensities often become unreliable in case of reflective or textureless surfaces. To overcome this challenge, we propose to perform image-level matching using a combination of two attributes: (i) normalized cross correlation (NCC) to compare the local image patches based on the user-marked region as a template, and (ii) the extracted edges to compare the gradient maps. As stated before, since the 3D line reconstruction helps to filter out noisy and outlier line features and yields a more reliable feature set, as an initialization we use the edges obtained by projecting the 3D lines onto the rectified plane images. Let  $\{l_i^t\}$  be the lines from the user strokes that define a template and let  $\mathcal{L}' = l_j^t$  denote all the lines containing the edge segments of the projected 3D lines. For each  $l^t \in l_i^t$ , we select all lines  $l \in \mathcal{L}'$  with similar orientation, i.e. the angle between the two line directions is less than 5-10 degrees, and  $dist(l, l^t) < \epsilon$ . In this expression,  $dist(l, l^t)$  denotes the average distance of the endpoints of  $l$  to the line  $l^t$  and  $\epsilon$  is a suitable threshold (e.g. 30 pixels). We then project the selected lines onto  $\{l_i^t\}$  and measure the percentage of the line lengths in  $\{l_i^t\}$  covered by the projected feature lines. This percentage measure gives us a line compatibility score. Finally, we compute the similarity score as a weighted combination of line compatibility and (absolute) NCC scores in the range  $[0.0, 1.0]$ . A match is accepted if this final score is above a threshold (e.g 0.8).

In man-made objects, especially in building facades, regularity is predominant not only across element-pairs but also across their mutual arrangement – typically in the form of 1D and 2D grid-like arrangements. This is not surprising since architectural guidelines give strong preference to such grid-structures both for aesthetic and economic considerations [27]. Having detected the repeating elements on the rectified plane textures, our goal is to estimate the vertical and horizontal transformations relating the repetitions. We first get initial estimates of the grid generating transformations by ordering the elements with respect to their horizontal (and vertical) positions and then look for the dominant horizontal (and vertical) transformations between the subsequent elements.

**Symmetry refinement:** Once we detect the repeating elements and get an initial estimate of the transformations relating them, for any plane, we obtain a collection of lines defining the contours of the elements that are linked by a transformation  $T$ . Without loss of generality, assume the set of lines  $\{l_1, l_2, \dots\}$  denote the corresponding part of the contours across the elements, i.e.  $l_i \approx T^{i-1}(l_1)$ . We represent the lines in the normal-intercept form as  $l_i = \{\mathbf{p} | \mathbf{n}_i^T \mathbf{p} + d_i = 0\}$ . At this stage, both the line parameters  $\{\mathbf{n}_i, d_i\}$  and the estimated transform generator  $T$  are often imprecise and our goal is to improve these initial estimates. In other words, we look for a base line parameter  $l'_1 = (\mathbf{n}, d)$  and a coupling symmetry transform that best explains the observed data. Without loss of generality, in the following we assume we have a set of elements arranged in a 1D grid.

Translational symmetry encodes the line offset  $o$  such that any other line is represented as  $l'_i = (\mathbf{n}, d + (i-1)o)$ . Let any of the original lines  $l_i$  have end points  $\mathbf{p}_i^1$  and  $\mathbf{p}_i^2$ . Extracting the best line parameters along with the coupling symmetry transform, then, accounts to minimizing

$$E(\mathbf{n}, d, o) = \sum_i w_i ((\mathbf{n}^T \mathbf{p}_i^1 + d + (i-1)o)^2 + (\mathbf{n}^T \mathbf{p}_i^2 + d + (i-1)o)^2), \quad (3.7)$$

with the side constraint  $\|\mathbf{n}\| = 1$  and  $w_i = \|\mathbf{p}_i^1 - \mathbf{p}_i^2\|$  represents a weighting between the lines based on their length. This weight helps to diminish the effect of very short noisy line segments. We solve this optimization problem in a two-step iterative approach to find the line parameters  $(\mathbf{n}, d)$  and the translational offset  $o$ . Specifically, we alternate between the computation of the transform parameter  $o$  and the line parameters  $(\mathbf{n}, d)$  using a least-squares and an eigen-value formulation respectively. Once converged (typically in 2 to 5 iterations), we recompute the set of close lines to the optimized template strokes and repeat the entire optimization procedure  $k = 3$  times (see Figure 3.6). The analysis is similar in case of a 2D translational grid where we optimize for both the horizontal and vertical transformations. We emphasize that this process is effectively performing symmetrization [70] in the space of lines.

**Structure completion:** So far, we have used the projected 3D line edges for structure discovery as these edges are considered to be more reliable features compared to the original edges. On the other hand, the 3D line feature set is much sparser due to the aggressive pruning of the mismatching edges. This sparsity leads to missing of certain grid elements to be detected. However, the refined symmetry information helps to identify the outlier edges and thus we can make use of all the remaining 2D edges.

Let  $\mathcal{L}^2$  denote the set of image-level edges that fall into any 3D plane. Assuming the detected regularity pattern is under a 1D or 2D grid structure, we propagate the detected grid structures and also test in regions of missing elements, this time using the edges from  $\mathcal{L}^2$  instead of the projected 3D line edges. After, any missing element is detected, we perform the simultaneous line and transformation optimization using all the repeating elements to further refine the symmetry



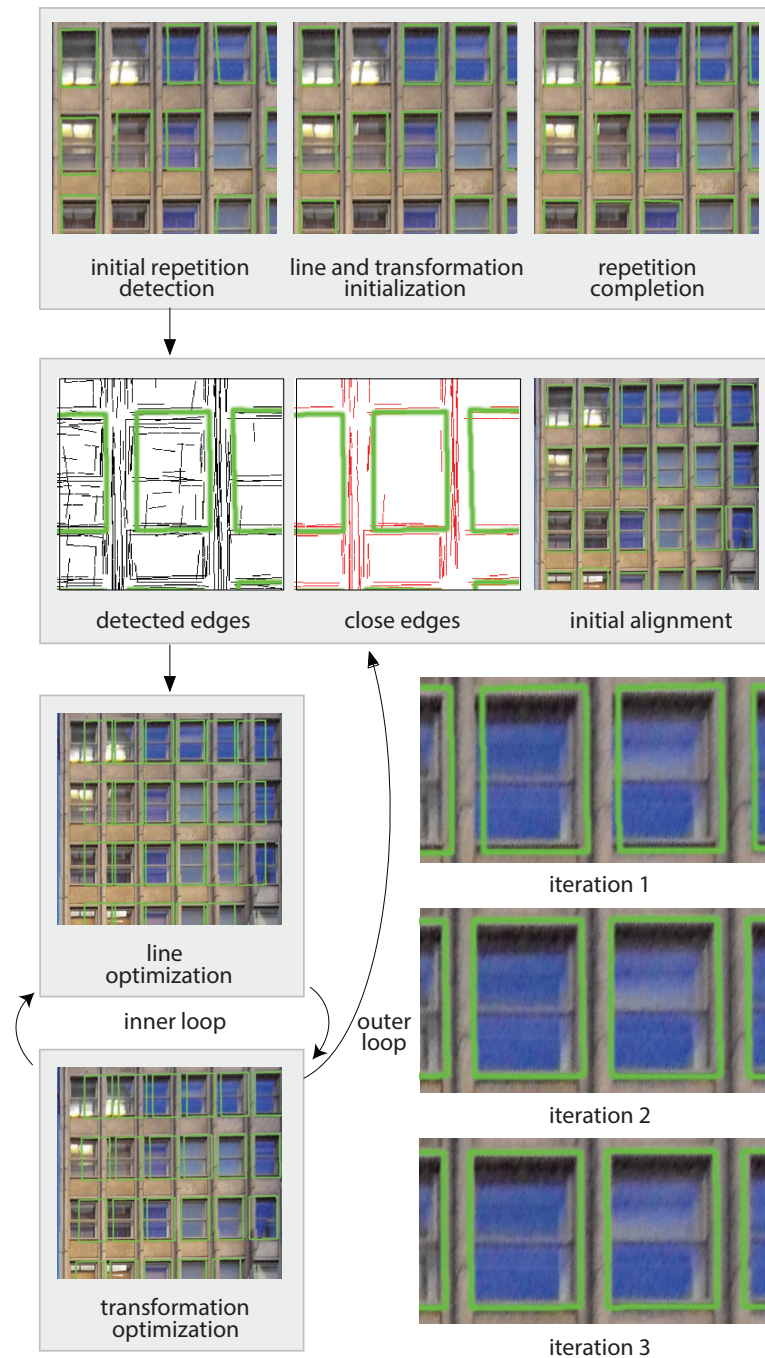


Figure 3.6: Symmetry refinement is performed on the initially detected repetitions to initialize the line and the transformation parameters for grid fitting. After the missing elements are detected using the refined symmetry information, the optimization procedure is repeated to get the final grid alignment. This procedure consists of an *inner* loop of successive iterations of line and transformation optimization and an *outer* loop of updating the template strokes and reselecting the close edges.

information as shown in Figure 3.6.

### 3.2.4 Procedural Depth Refinement

At the end of the discrete symmetry extraction step, we have a set of elements with respective repeating patterns on each of the 3D planes. Each element is encoded with its contour lines. This explicit encoding of the elements provides a *factored* model making subsequent image- or model-space editing operations easy and intuitive.

In practice, detected element patterns typically are offset surfaces from their embedding planes. Hence, we require to capture depth offsets for such extrusions to produce a 3D model. This depth information can be recovered automatically as follows. For each element, we can perform a 1D depth search in an offset range  $[-\sigma, \sigma]$  along the direction of the normal of its embedding plane. Specifically, we discretize this search space and at every sample, we move the contour lines of the elements to the specified depth, project it back to the original images  $I_i$  and compare the projected edges with the original 2D edges in this image. The correct depth offset will result in the highest similarity between the projected and the original edges. In case of insufficient image resolution or subtle depth changes, however, this automatic method fails to recover the correct depth of the elements. However, the factored representation enables the user to manually prescribe a depth assignment for a single element that can be propagated to all the other symmetrically coupled elements (see Figure 3.7).

## 3.3 Evaluation

We evaluate the proposed reconstruction framework on a variety of challenging real and synthetic scenarios such as non-Lambertian surfaces and abrupt changes in lighting. We provide a collection of results in Figure 3.8 and provide statistics about these results in Table 3.1. We now summarize our main findings below.

**Synthetic Evaluation:** We perform evaluations on the synthetic data sets to measure the accuracy

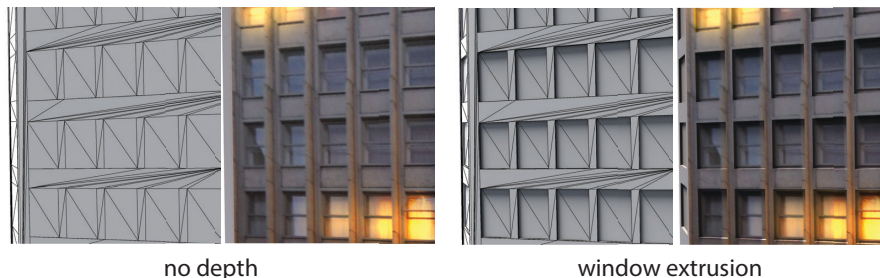


Figure 3.7: User-guided depth refinement based on the extracted symmetric elements helps to recover shallow depth features. We show recovered geometry with and without texture.

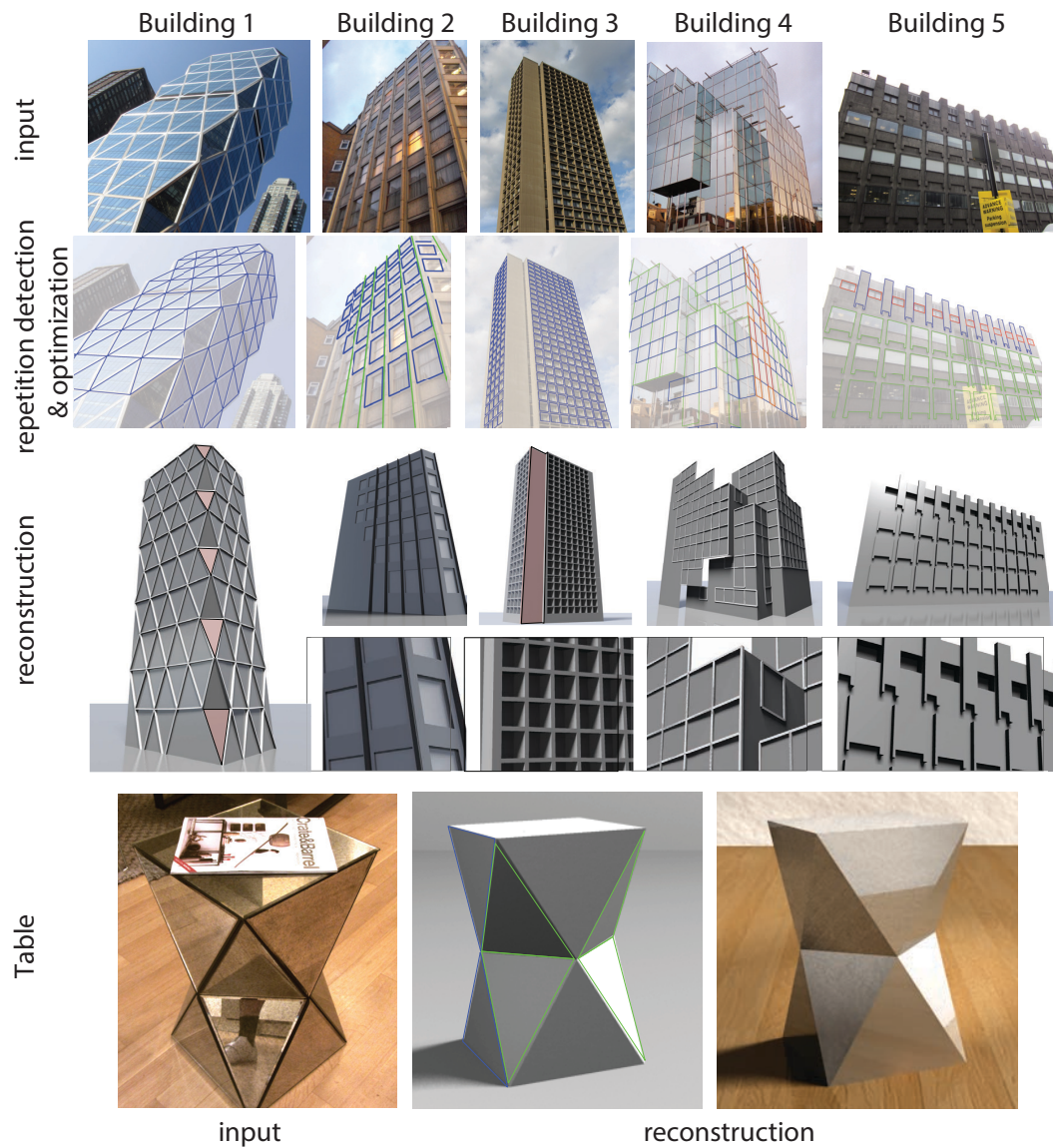


Figure 3.8: For each example, we show the optimized repetition patterns with different colors indicating separate structures. The red planes shown in the final reconstructions have been added with user assistance due to lack of stable line features. *Building 4* and *table* examples have highly reflective surfaces.

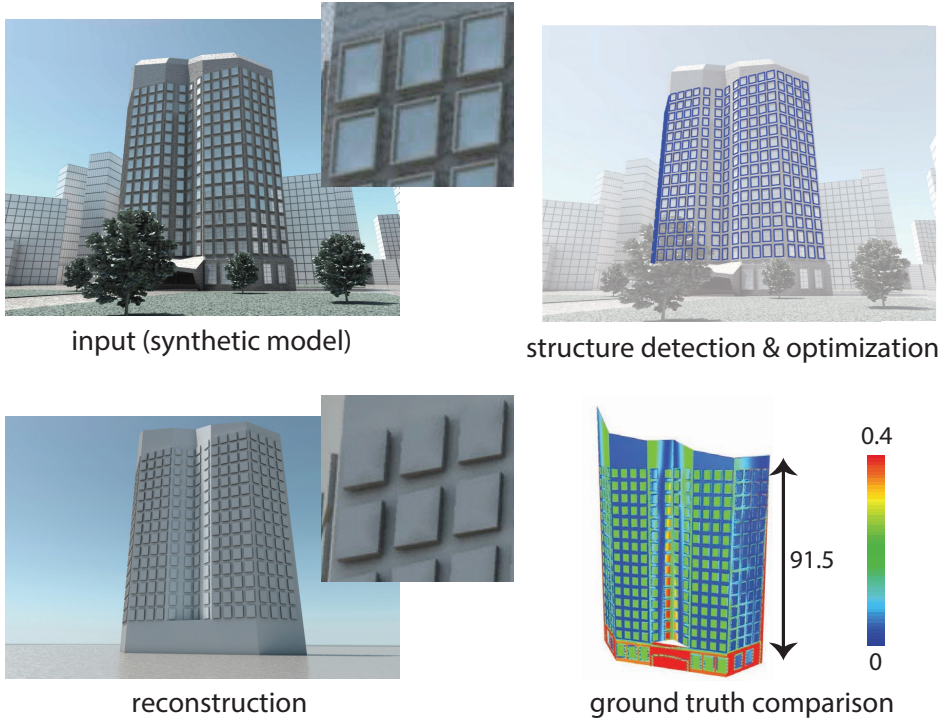


Figure 3.9: We perform synthetic evaluations to measure the accuracy of our approach.

of the symmetry-based optimization step in recovering the correct boundaries of the repeating elements (see Figure 3.9). Specifically, we render images of a synthetic module and use these images as input for our approach. We perform a comparison between the ground truth model

	Building 1	Building 2	Building 3	Building 4	Building 5	Table	Synthetic
$\#N_i$	25	26	13	9	27	13	24
res	5.7	6.2	7.6	6.2	5.0	5.7	3.0
$N_e$	2600	1200	2000	2400	1500	750	3400
$N_l$	2891	1570	457	1128	1822	173	3763
$N_p$	2	3	2	6	1	7	5
$N'_p$	4	0	2	0	0	0	0
$N'_r$	1	2	1	4	3	2	1
$N_r$	102	80	300	156	57	8	300
$T_l$	55	25	35	40	40	3	45
$T_p$	6	5	4	6	-	16	6

Table 3.1: The table shows the number of input images ( $N_i$ ), the resolution of the images in megapixels (res), the average number of 2D edges detected ( $N_e$ ), the number of 3D lines reconstructed ( $N_l$ ), the number of automatically fitted planes ( $N_p$ ), the number of manually selected planes ( $N'_p$ ), the number of elements marked by the user ( $N'_r$ ), and the total number of repeating elements detected ( $N_r$ ) for each data set. The computation times for 3D line reconstruction ( $T_l$ ) and plane-based image segmentation ( $T_p$ ) are given in minutes measured on a 3.33 MHz 24-core machine.

and our reconstruction obtained by optimizing for the contours of the repeating windows and extruding them to the correct depth. We set the maximum distance to 0.5% of the height of the building and provide a color-coded distance measure between the models. Typically, we observe small error around the boundaries of the windows and slightly higher error inside the windows due to the subtle depth changes in these regions in the ground truth model. The highest error is produced around the door region where we have missing planes. In this example, the user defined template element is used to detect repeating windows across multiple planes.

**Regularity Initialization:** When detecting discrete symmetries, our 2D-3D coupled repetition detection algorithm uses a weighted combination of image-based normalized cross correlation (NCC) score and line-based similarity to compare elements. For examples where there are sufficient image features, e.g. *Building 3* and *5* in Figure 3.8, NCC matching provides a good initialization of the present regularity. On the other hand, as the surfaces become more reflective and textureless, e.g. *Building 4* and *table*, image-based comparisons become inaccurate, while 3D line features provide a more reliable result. Hence, we normally use an equally weighted combination of image- and line-based similarity measures but rely only on line-based similarity for highly reflective surfaces to initialize the regularity discovery. The symmetry-based optimization aids the initialization and helps to discover the remaining missing repeating elements, which are otherwise challenging to detect.

**Comparisons:** In a recent effort, Wu et al. [119] has proposed a method that exploits symmetry priors in the significantly more challenging scenario of single-view reconstruction. We compare our approach to this method as shown in Figure 5.10. This example illustrates the benefits of a multi-view approach that couples symmetry information across multiple images, leading to faithful reconstructions in general.

Discrete symmetries provide multiple observations of the same piece of geometry to reduce noise and perform hole filling. This is performed implicitly in our framework since we integrate information across different symmetric pieces into one consistent contour representation that is then copied to all instances. This symmetry-aware reconstruction approach outperforms general MVS algorithms as shown in Figure 5.10. As a standard MVS algorithm operates on 3D point samples, a surface reconstruction algorithm such as Poisson Surface Reconstruction (PSR) [49] is necessary to extract surfaces. The MVS algorithm produces noisy and sparse point sets, especially for reflective surfaces and PSR creates a smooth surface while filling the holes with blobs. In contrast, our initial edge-based stereo approach enables to distinguish between the spurious and real features and initializes a consistent reconstruction that preserves sharpness. Additionally, we obtain a compressed representation that enables not only efficient data storage, but can directly be used for structure-aware edits of the geometry.

**User Interaction:** Our framework supports several types of user interactions:

- After the automatic planar-based reconstruction step, there might be image regions labeled with wrong plane assignments. In this case, the user can indicate rough strokes in one of



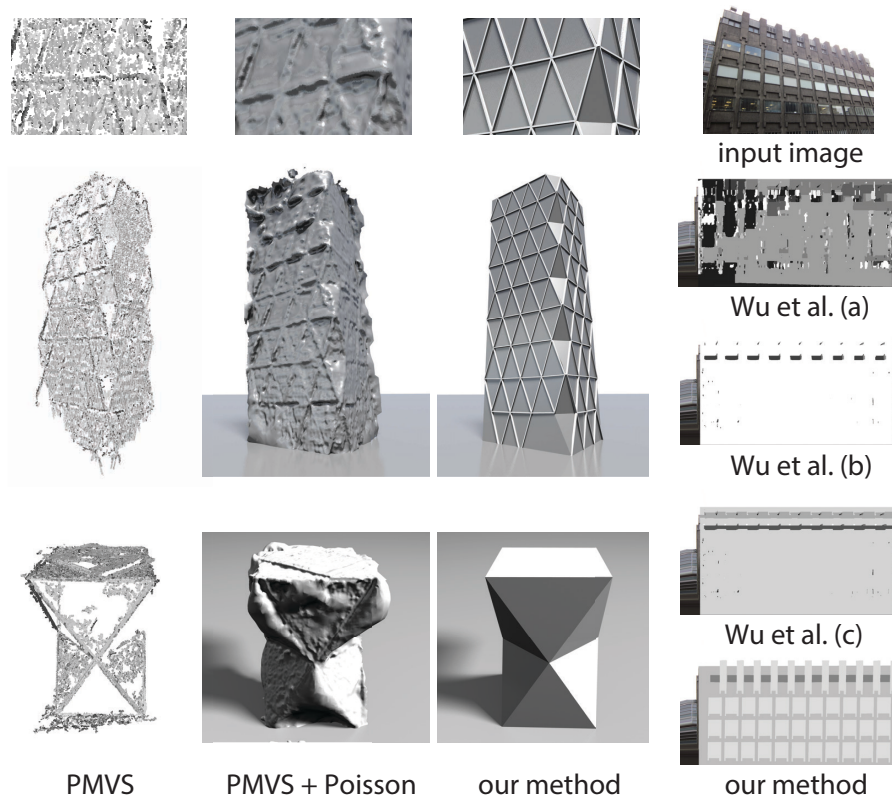


Figure 3.10: The comparison with the patch-based MVS method [33] illustrates that symmetry priors and non-local consolidation are essential for objects with complex materials and repetition patterns. The method of Wu et al. [119] fails to recover the depth of the repeating elements if the depth change with respect to the main plane is too small. We provide depth assignments computed by different weighting terms ((a) no repetition term, (b) repetition and smoothness terms weighted equally, (c) smoothness term weighted more). We refer the reader to the paper for details.

the mislabeled images that get snapped to the 2D edges to define new segment boundaries. Such corrections in plane boundaries are then propagated to the other images. A correction of this type has been used for the *Building 4* example as shown previously in Figure 4.1.

- Lack of sufficient 3D lines might lead to missing candidate 3D planes, e.g. thin plane regions with little support. In order to generate these missing planes, we require the user to mark two edges to define the plane in two images which are converted to 3D lines to compute the plane parameters in 3D. Intersections with the current planes are used to define the boundaries of the new plane. We have used this mode in the *Building 1* and *4* examples to indicate new planes which are shown in red in Figure 3.8.
- Often there is an ambiguity in selecting semantically correct element boundaries and the scale of the repeating elements that is difficult to resolve automatically. For humans,

however, it is trivial to roughly mark a representative element of the intended regularity. Therefore, we allow users to roughly indicate a single element, which is then used to detect other repeating instances. As seen in Table 3.1, with only a few user marked elements our algorithm can detect almost a complete set of repetitions.

- We enable the users to indicate shallow extruded depth assignments for the detected elements similar to Müller et al [75].

**Limitations:** Even when exploiting symmetry priors, surface reconstruction from images remains a challenging problem. Our method is based on reliable 2D edge detection and 3D line reconstruction. Thus, if we don't have such sufficient features to describe the geometry, our method will fail. Similarly, symmetry detection will be ineffective in cases of limited repetition or strong variations in the repeating elements, e.g. due to weathering. We focus on piecewise planar surfaces bounded by straight edges, as are most common modern buildings. Curved edges or surfaces are not handled by our method.

### 3.4 Closing Remarks

We have presented a coupled formulation for detecting symmetric line arrangements and 3D reconstruction for producing *factored* models. This approach benefits from large-scale model repetitions and can handle inputs with reflections or outlier objects. The coupled formulation simultaneously improves symmetry detection and reconstruction quality. We bootstrap the reconstruction process using rough image-space user markings. The factored facade models provide an effective encoding of the individual building elements. This allows to perform simple editing tasks. We have shown examples by extruding these elements to desired depth locations. The refined element contour detection also opens up the possibility to replace the original elements with synthetic counterparts. In this framework, we assume the initial camera calibration to be given. We next focus on redefining the camera calibration problem by coupling it with symmetry detection and 3D modeling.





## 4 Symmetry and Structure-from-Motion

In Chapter 3, we have demonstrated that exploring symmetry priors in an image-based reconstruction framework enables to exploit non-local coherence of symmetric elements to generate precise model reconstructions. The first step of such a framework is to obtain camera parameters relating the input images which we assumed to be provided as input. The core challenge of this step is to establish reliable image correspondences as demonstrated in the previous chapters. This problem becomes particularly difficult in presence of repeated elements that give rise to multiple and ambiguous correspondences as in the case of building facades. Often a significant amount of discriminating features coming from the surrounding of the buildings are necessary to resolve the ambiguities (as was the case for the examples shown in the previous chapter). Lack of sufficient amount of such discriminating features in combination with wide spread repetitions, unfortunately, make stable correspondence estimation difficult, potentially leading to poor reconstruction results.



Assume our goal is to register the pair of images shown on the left and we have detected a feature point on the corner of a window frame in one of them. The presence of repeating window frames give rise to multiple candidate feature matches in the other image leading to ambiguity. In case sufficient descriptive features are not detected to resolve such ambiguities, traditional image-based methods often contain one of the following artifacts: (i) large-ambiguities due to content repetition cause standard Structure-from-Motion (SfM) meth-

ods to produce poor and noisy 3D output, or (ii) SfM produces apparently reasonable 3D output, but with an incorrect number of repeated elements. Even if the camera calibration is seemingly successful, the results can be suboptimal, producing sparse, incomplete 3D reconstructions that often accumulate error leading to drifts (e.g. straight facades appearing curved).

Thus, we are faced with an *ironic* situation. On one hand, multiple observations of the same geometry provide non-local consolidation of data resulting in improved reconstruction quality. On the other hand, presence of repetitions lead to ambiguous feature matches making it challenging to register a set of input images. We propose to turn this situation in our advantage by injecting symmetry information early into the reconstruction process. Specifically, assume we have an initial guess about the repetition grids present in the image pair given above. This information can be used to guide the correspondence search between these images since at a high level registering the two images is equivalent to establishing correspondences between the elements of the common repetition pattern, e.g. a grid of window frames.

We can state our goal as to simultaneously detect regularities and establish correspondences across the images. Thus, we aim to address the cyclic dependency of the problem of 3D reconstruction with repeating structures: stable 3D symmetry detection requires accurate camera calibration to obtain correct 3D point samples, while accurate camera calibration requires stable symmetry detection to resolve ambiguities. The size of the repetition pattern is not known a priori, however, and the pattern is not necessarily visible as a whole in any of the images. Thus, the image registration problem amounts to recovering the repetition pattern while computing the camera parameters of the images that view this pattern.

We have witnessed other parallel research efforts in recent years that aim to establish globally consistent relations between a set of input images. To illustrate, Zach et al. [122] use a graph to encode visual relations in image collections and infer false matches based on inconsistencies of cycles in this graph. However, they do not explicitly model repetitions as in our approach, which we found to improve the quality of the results significantly. In a follow-up work, Cohen et al. [20] propose to use symmetry priors with collinearity and orthogonality constraints to reduce drifts in a given SfM output. They use the method of Zach et al. [122] to compute this initial SfM reconstruction which is assumed to be free of ambiguity. Roberts et al. [90] focus on a specific instance of the image matching problem where large duplicate structures are present in the scene. They explore non-geometric cues such as image timestamps to resolve ambiguities. Jiang et al. [48] eliminate the dependency on image timestamps by formulating the problem as finding the spanning tree of an image matching graph minimizing a global energy function and propose a greedy search algorithm. More recently, Wilson and Snavely [113] aim to identify wrong feature correspondences across images with repeating structures by analyzing the local structure of a bipartite graph where feature correspondences are represented as edges from a set of images to anticipated 3D points. In this analysis, contextual cues obtained from the background environment play a crucial role.

Our approach is inspired by these recent efforts to establish globally consistent image relations. Unlike all other efforts, however, we *jointly* focus on regularity detection and constrained SfM formulation. We demonstrate that, with this explicit coupling we achieve significant improvements both in terms of robustness and accuracy.

## 4.1 Overview

We propose a Structure-from-Motion (SfM) framework focusing on images of buildings that exploits symmetry priors [17]. More specifically we explore repetitions arranged as planar grids as these are the dominant type of regularities in typical building facades. The key to our approach is to couple camera calibration with symmetry detection. Such a coupling has several advantages. Focusing on planar repetitions significantly reduces the search space of geometric relations across images since any transformation induced by correspondences between images should overlap the boundaries of the repeating elements. Exposing these constraints explicitly increases the reconstruction accuracy. As output, we obtain a globally consistent 3D reconstruction with explicit encoding of the extracted regularities. This information acts as a link among the input images and the 3D reconstruction and thus enables several interactive editing possibilities.

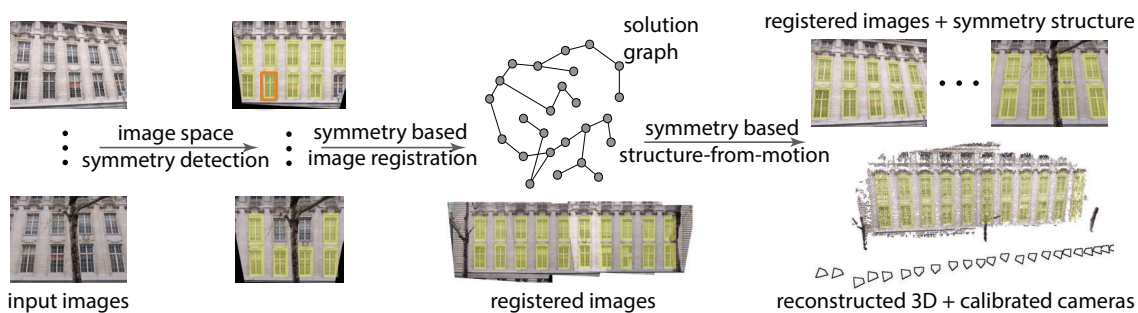


Figure 4.1: Given a set of images, we perform symmetry detection in each image based on a user marked template (in orange). We use this symmetry information (shown in yellow) to solve for a consistent global repetition pattern using a graph-based optimization. We then use a symmetry-based SfM method to simultaneously calibrate the cameras and extract a 3D reconstruction.

Our framework takes as input a set of unordered images  $\mathcal{I} = I_1, I_2, \dots, I_n$  of a building, which we assume to contain one or multiple 1- or 2- parameter repetition patterns on each facade. We do not require the full facade repetition grid to be completely visible in any single image and the repetitions can be arranged in multiple facades.

The user roughly indicates elements of interest, for example, a repeating window frame visible across the input images, on any *one* of the images. This is the only manual assistance required in our pipeline. We then detect repetitions of these elements and compute initial estimates for the grid generators of the detected patterns.

Next, in a key algorithmic stage, we pose correspondence search across images in  $\mathcal{I}$  as solving for offset positions for each image on an (unknown) global repetition grid. We simultaneously solve for offset positions and estimate the associated camera parameters for each image, while also detecting the 3D repetition pattern. The optimization is formulated as selecting a subset of consistent edges from an *image matching graph*, where a node corresponds to an image in  $\mathcal{I}$  and

each edge denotes an estimated image-pair alignment, which can possibly be wrong. We solve for a consistent set of edge alignments by progressively refining the alignments.

Subsequently, we perform bundle adjustment with the extracted symmetries as constraints that encode the grid arrangements of the repeating structures. Our algorithm outputs the final camera poses and a sparse 3D reconstruction of the scene together with the refined symmetry parameters in 3D. We illustrate that this information is useful for exploring a range of editing possibilities coupled across multiple images via the detected symmetries.

## 4.2 Algorithm Details

We now provide the details of the individual components of our pipeline.

### 4.2.1 Initial Grid Estimation

Given a set of input images  $\mathcal{I}$ , our goal is to bring these images into correspondence. Our main observation is that the underlying facade repetition structure can be explored to restrict the continuous search space of alignments to a discrete set of possibilities, that is the correspondence across an underlying 2-parameter repetition grid. Thus, we first aim to detect such repetitions for each image, which are later used to globally establish correspondence across all the individually extracted grids.

As demonstrated in the previous chapter, building facades typically contain dominant repetitions arranged along vertical and horizontal directions. Therefore, in order to eliminate the effect of perspective projection, we first detect dominant vanishing points in each image  $I_i \in \mathcal{I}$  using the cascaded Hough transform [106] and use these vanishing points to rectify the original images to be fronto-parallel. With slight abuse of notation, we will denote these rectified images as  $I_i$  as well. For images where multiple facades are visible, we detect multiple vanishing points and perform rectification with respect to each of these candidate vanishing points.

Our next goal is to detect repeating elements in the resulting rectified images. As in the case of the reconstruction framework presented in the previous chapter, we allow the user to prescribe the element of interest in *one* of the input images. This user assistance allows to identify a semantic element which can later be used in various editing applications. Since the rectified images can be at different scales, we match features across them to estimate their relative scalings. Specifically, as rotation changes are already compensated through rectification, we extract SIFT features with fixed upright orientation. We estimate the scale change between a pair of rectified images by clustering the scale differences between the feature matches (see also the work of Baatz et al. [5]). We make a note that ambiguous feature matches arising from repetitions *do not* affect the scale estimation since SIFT features detected in a rectified image have similar scales across the repeating elements (see Figure 4.3).

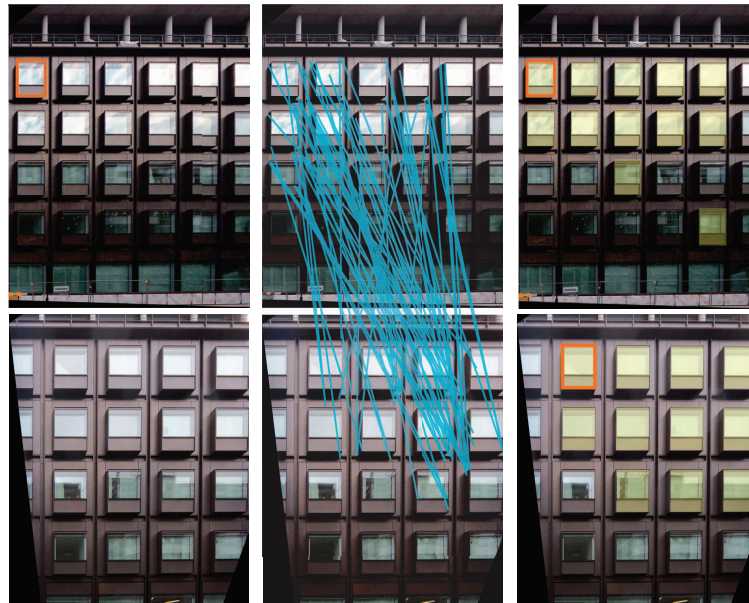


Figure 4.2: Given a pair of rectified images (left) with a repeating element  $T$  marked in one image (top-left), we use SIFT features to estimate the scale factor  $s$  relating the image pair (middle). We then create a scaled template  $sT$  suitable for the other image. Thus, we detect image-level repetitions (in yellow) across each individual rectified image (right).

Once the scale changes across the rectified images are computed, the user marked element is scaled accordingly for each rectified image  $I_i$  and used as a template for repetition detection. We perform template matching in the images by comparing local image patches based on the (scaled) template using Normalized Cross Correlation. Finally, for each image we complete the initialization by fitting a grid to the detected elements and estimate the corresponding grid generating transformations as introduced in the previous chapter.

#### 4.2.2 Symmetry-based Image Registration

The key to successful SfM computation is correctly establishing image correspondences. For images of facades with dominant symmetries, correspondence extraction is difficult as repeating structures create multiple locally consistent matches, many of which are wrong. To avoid this challenge, we establish globally consistent correspondences by explicitly using the extracted repetition information. Intuitively, our goal is to position each rectified image on a common regularity grid with spacing as extracted in the previous section. Using the current symmetry estimates for each image, positioning the images amounts to assigning discrete index positions to each image on this repetition lattice. In other words, the transformation between the images amounts to calculating shifts in rows and columns required to align the corresponding grids in these images. In order to consistently assign these shifts, we collectively analyze the images.



Figure 4.3: Given a pair of images with their detected grids (left), for each candidate alignment we detect the overlapping grid regions (shown in green). We compute feature matches outside the overlapping regions and count the number of matches supporting the candidate alignment. Top-right shows the correct alignment with highest support (51 matches) and the bottom-right is a wrong candidate alignment (25 supporting matches).

For each image pair  $(I_i, I_j)$ , a candidate alignment can be encoded as the number of rows and columns the corresponding grids should be shifted over each other. Detection of the image-space grids enables to list all possible candidate alignments between  $I_i$  and  $I_j$  as a discrete set of shifts in rows and columns. Our goal is to rank such candidates to find the most likely alignment for the image pair. We use SIFT feature matches detected in the original images to perform this evaluation. The detected SIFT features can be grouped into two categories: (i) features that are found outside the repetition regions are most likely to help disambiguate the correct alignment between an image pair; and (ii) features that are detected inside the repetition regions are likely to result in ambiguous matches. In practice, we observe that grid regions also contain discriminating features due to small random variations such as the shape of curtains, window customization, etc. Therefore, instead of discarding all the features detected inside the repetition regions we only discard those features that are most likely to cause ambiguous matches. Specifically, each candidate alignment between an image pair overlaps certain grid regions according to the encoded shift in rows and columns. We assume that a correct alignment will map the grid regions that are most similar in both images. Further, the feature matches obtained from the remaining regions should agree with the candidate alignment. Therefore, for each candidate alignment, we discard the features detected only inside the overlapped grid cells and match the remaining features. Each pair of feature matches produces an estimate of the scale and translation that maps the corresponding rectified images on which the grids lie. We convert each of these estimates to shifts in rows and columns between the two grids by computing the grid cells they map. We



consider two grid cells to be mapping to each other if the distance between the cell centers after mapping is below a certain threshold (2% of the image width in our experiments). The feature matches, which suggest the current row/column shift being evaluated, provide support for this candidate (see Figure 4.3). After evaluating all possible candidate alignments for an image pair  $(I_i, I_j)$ , we pick the candidate alignment that receives the highest support and assign a weight  $w_{ij}$  equal to the number of supporting feature matches. After all pairwise candidate alignments are detected, we normalize the support weights of all the selected image alignments by dividing by the maximum number of support matches.

We observe that exploring the grid information during feature matching improves the quality of the detected candidate alignments. However, considering only local pairwise relations is unlikely to resolve all the ambiguities. Instead, in a subsequent step, we analyze collections of pairwise relations.

**Image matching graph.** We encode the detected pairwise image alignments as a matching graph  $G = (V, E)$ , where a node  $n_i \in V$  represent the image  $I_i$ , while an edge  $e_{ij} \in E$  represent the alignment picked between the image pair  $(I_i, I_j)$ . We recall that each edge is weighted by the corresponding support  $w_{ij}$  for the image alignment. The edges in this graph are constructed using information only from image pairs, and hence can contain spurious matches due to ambiguity arising from repetitions. In order to detect such spurious edges, we look for consistency among edges in cycles in this matching graph to assess the reliability of the image alignments.

A key observation about the image matching graph is that accumulated alignments along the edges in any cycle in this graph represent a mapping from an image to itself. Thus, the corresponding cumulative transform should be the identity, i.e, accumulated corresponding grid shifts should result in zero row and column shift. Any cycle where the accumulated shifts do not cancel out indicates the presence of at least one incorrect alignment edge in the cycle. We call such cycles *inconsistent*. Our task is to identify such spurious edges in inconsistent cycles and remove them from  $G$ , while still retaining the consistent alignments.

**Optimization setup.** Based on the preceding observations, we now select a consistent set of alignment edges while discarding the wrong alignments. Effectively, we identify the wrong alignments based on the corresponding supporting weights and the inconsistencies involved. We introduce a binary penalty cost  $\chi_e \in \{0, 1\}$  for each edge  $e \in E$ , where a penalty cost of 1 denotes a wrong alignment and a penalty cost of 0 indicates a correct alignment. Our goal is to extract a globally consistent penalty cost assignment for all the edges in  $G$  via a joint formulation. We extract such a set of consistent assignment of costs  $\{\chi_e\}$  for edges  $e \in E$  as:

$$\begin{aligned} & \min_{\chi_e} \sum_{e \in E} w_e \chi_e \\ & \text{subject to } \sum_{e \in L_i} \chi_e \geq 1, \forall L_i \in L \quad , \end{aligned} \tag{4.1}$$

where  $L$  denotes all the detected inconsistent cycles in  $G$ . In order to make this aforesaid integer problem convex, we relax the constraints  $\chi_e \in \{0, 1\}$  to be  $\chi_e \in [0, 1]$ . We solve the resultant problem using CVX, a package for specifying and solving convex programs [22].

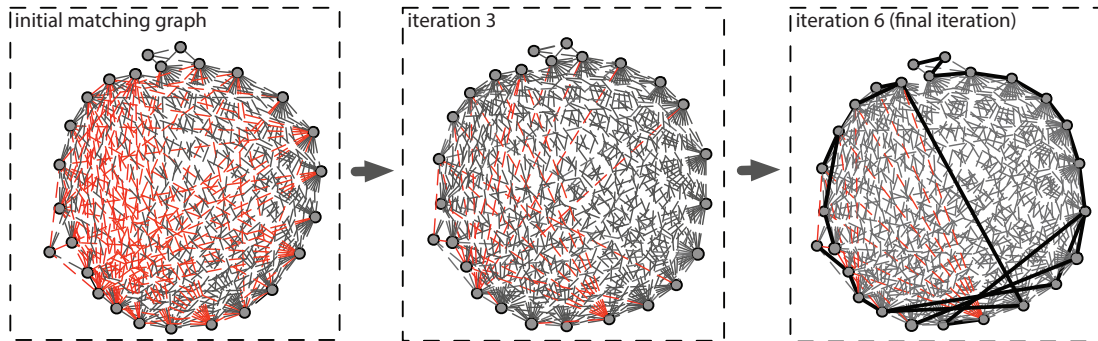


Figure 4.4: Starting from all candidate pairwise matches, we introduce an optimization that iteratively improves the quality of the alignments (wrong alignments are shown in red). The minimum spanning tree of the final graph (shown as solid edges) provides the final image alignments.

In order to save computation, we only look at 3-cycles in the matching graph  $G$  and extend this analysis to longer cycles in an iterative approach (see also the work of Nguyen et al. [80]). Intuitively, we first resolve 3-cycles in the graph  $G$  and then use the extracted consistent edges to improve estimates of the other alignments. In order to balance between global consistency and local feature-level image-pair matching, we discourage rejection of high-confidence edges, i.e. edges with high weights. Hence, we normalize the computed edge penalties as  $\chi_e \leftarrow \chi_e / w_e$  for all the edges. (We update the range of  $\chi_e$  to be  $[0.1, 1]$  to avoid getting continuous penalties of 0.) Next, we use the current edge costs  $\chi_e$  to improve image alignments. Specifically, for any edge  $e_{ij} \in E$  we compute the shortest path between image nodes  $n_i$  and  $n_j$  using the edge costs  $\chi_e$  in  $G$ . If the total cost of such a shortest path is less than the cost of the original edge, we replace the alignment denoted by  $e_{ij}$  by composing the alignments along the shortest path, thus potentially improving the alignment for images  $I_i$  and  $I_j$  (see Figure 4.5). We also update the weight  $w_{ij}$  to be the minimum of the weights along the shortest path. Note that the alignments replaced in this manner implicitly represent longer paths and hence the 3-cycles considered in the successive iterations actually end up as longer cycles in the original graph [80]. After performing the necessary alignment replacements, we resolve a new global optimization using Equation 4.1 and continue the process. This iterative algorithm converges when no more alignment gets replaced (8–10 iterations in our experiments). After convergence, we select the minimum spanning tree of  $G$  based on the final edge costs  $\chi_e$  to obtain the final image alignments (see Figure 4.4). We use these alignments to filter the pairwise feature matches and preserve only the matches that support the corresponding alignments.

**Non-grid alignment:** In case of images where no grid is detected or a candidate grid matching alignment with sufficient support (a minimum of 20 supporting feature matches in our tests) is



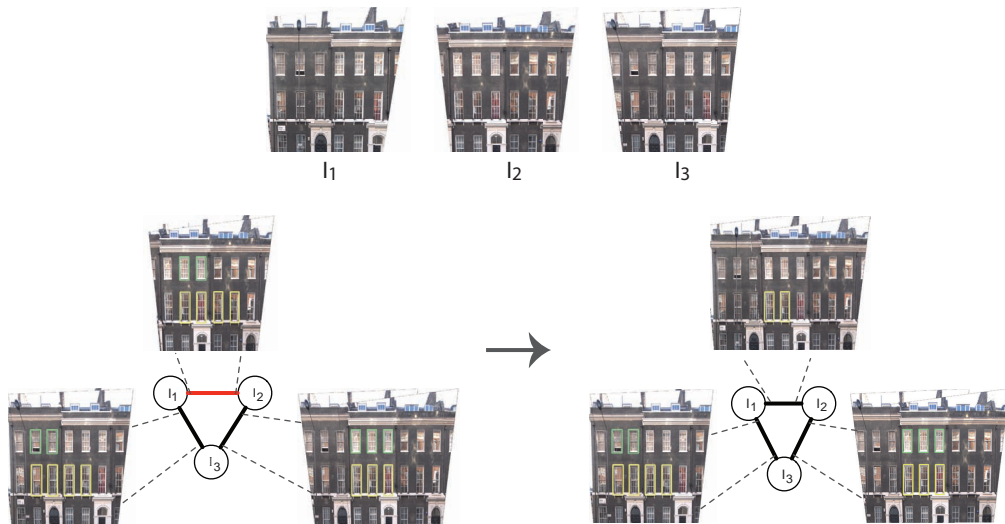


Figure 4.5: For the images  $I_1$ ,  $I_2$ , and  $I_3$  (in top), the wrong candidate alignment between  $(I_1, I_2)$  (in red) is replaced by the correct accumulated alignment along the shortest path  $I_1 \rightarrow I_3 \rightarrow I_2$  during the iterative grid optimization. For each alignment, the overlapped images are shown together with the mapped grid regions (yellow and green).

not found, we perform traditional SIFT feature matching and encode the candidate alignment as a transformation matrix. In order to evaluate the consistency of a cycle involving non-grid alignments, we compute the rotation associated with each of the candidate alignments and compute the composite rotation  $R_L$  along the cycle [122]. In a consistent cycle,  $R_L$  should be equivalent to the identity transformation. However, in a noisy setting this equivalence holds only approximately. We compute the rotation angle  $\alpha_L$  of  $R_L$  using a quaternion representation and mark the cycle as inconsistent if  $|\alpha_L|$  is greater than a threshold angle ( $10^\circ$  in our tests). Such non-grid alignments are often invoked for the images that view the corners of a building with multiple facades.

### 4.2.3 Symmetry-based Structure-from-Motion

In the previous section, we established globally consistent feature correspondences across the input images. In addition, we compute the center point of each extracted grid element and form correspondences across these points over all the images. We now use these correspondences to estimate the extrinsic camera parameters, while using the EXIF tags to obtain the internal parameters such as the focal length of the cameras. Using these initial camera parameters and the grid point correspondences across the images, we obtain rough 3D grid points and corresponding grid generators. Note that at this stage, both the camera parameters and the estimated grid generators are only approximate. We refine these parameters by a nonlinear bundle adjustment algorithm.

We first organize the feature matches and the grid correspondences between the images into tracks where each track represents a connected set of matching points across the images. Given such a set of feature tracks and the estimated camera parameters, our goal is to recover the 3D position corresponding to each track. We refine the camera positions and orientations to minimize the reprojection errors, that is, the distance between the projection of a track point and its corresponding image matches. More importantly, we use the estimated symmetry relations between the grid points as a regularizer. Thus, instead of independently computing each 3D grid point, we simultaneously look for the 3D grid parameters, the position of a reference point, and the generators, which minimize the sum of the reprojection errors of all the grid tracks.

Assume we have a set of  $m$  cameras parameterized by  $A_k$ . Let  $P(A_k, \mathbf{p})$  denote the projection function mapping a 3D point  $\mathbf{p}$  to its 2D projection  $\mathbf{q}$  in the  $k$ -th camera with parameters  $A_k$ . Furthermore, assume a grid with  $r$  rows and  $c$  columns is represented by the reference point  $\mathbf{o}$  and the grid generators  $(\mathbf{t}_h, \mathbf{t}_v)$ . Minimizing the reprojection error of the grid tracks corresponding to this 3D grid is equivalent to minimizing the following energy:

$$E_{grid} = \sum_{k=1}^m \sum_{i=1}^r \sum_{j=1}^c \lambda_{ij}^k \|\mathbf{q}_{ij} - P(A_k, \mathbf{x}_{ij})\|, \quad (4.2)$$

where the 3D grid point at the  $i$ -th row and  $j$ -th column of the grid is represented as  $\mathbf{x}_{ij} = \mathbf{o} + (i-1)\mathbf{t}_h + (j-1)\mathbf{t}_v$ .  $\mathbf{q}_{ij}^k$  denotes the projection of this point in the  $k$ -th camera and  $\lambda_{ij}^k$  is an indicator variable equal to 1 if this grid point is visible in the corresponding image and 0 otherwise.

We formulate a similar energy function for the remaining  $n$  non-grid feature tracks parameterized by the 3D points  $\mathbf{b}_i$ . Minimizing the sum of the reprojection errors for these tracks is equivalent to minimizing the following energy:

$$E_{other} = \sum_{k=1}^m \sum_{i=1}^n \beta_i^k \|\mathbf{q}_i^k - P(A_k, \mathbf{b}_i)\|, \quad (4.3)$$

where  $\beta_i^k$  is an indicator variable with  $\beta_i^k = 1$  if point  $\mathbf{b}_i$  is visible in the  $k$ -th image and 0 otherwise.

Finally, we combine the energy terms for the 3D grid and the non-grid tracks and minimize the resulting objective function using the Levenberg-Marquardt method [63]:

$$E = E_{grid} + E_{other}. \quad (4.4)$$

At the end of this enhanced bundle adjustment step, we obtain the refined camera parameters, a sparse 3D representation of the scene, and the refined grid parameters.

#### 4.2.4 Vanishing Line Refinement

The results obtained at this stage might retain any bias introduced in the initial vanishing line detection used for rectification. To reduce this bias, we update the rectification of the input images using the computed 3D geometry. Specifically, we fit planes to the reconstructed 3D point cloud where each plane corresponds to a facade of the building. Then we find the facade plane on which each detected 3D grid lies and update the position of each 3D point representing a grid cell by projecting to its facade plane. We reproject these 3D points to the input images to refine the grid correspondences across the images. For each 3D point  $\mathbf{x}$  representing a grid cell in a 3D grid, we compute the average reprojection error as:

$$P_{error} = \frac{\sum_{k=1}^m \lambda^k \|\mathbf{q}^k - P(A_k, \mathbf{x})\|}{\sum_{k=1}^m \lambda^k}, \quad (4.5)$$

where  $\lambda_k$  is an indicator variable equal to 1 if this grid point has been detected in the  $k$ -th image, and  $\mathbf{q}_k$  denotes this detected 2D grid point if  $\lambda_k$  is 1. We exclude the grid cells for which  $P_{error}$  is above a certain threshold (5% of image width in our experiments) and update the number of rows and columns of a grid accordingly. We observed that this evaluation helps to fix any minor errors that might have occurred during the grouping of the image grids (due to small variations in grid transformations which have not been detected in image space). With these refined correspondences, we rerun the constrained bundle adjustment step to obtain the final camera parameters and the 3D grids. In our tests, we found a single iteration of refinement to be sufficient (see Figure 4.6).

#### 4.2.5 Extension to Multiple Grids

Our SfM pipeline can handle buildings that contain one or multiple 1- or 2- parameter repetition grids. We now describe, in detail, how we adapt our framework to handle such multiple grids. We observe that multiple grids can occur as: (i) multiple arrangements of the same base element, or (ii) grids of different user-indicated base elements.

Given any image pair  $(I_i, I_j)$ , let us assume multiple grids have been detected in these images. We consider any grid pair  $(g_i, g_j)$ , where  $g_i \in I_i$  and  $g_j \in I_j$ , that shares the same base element as a potential projection of the same 3D grid. Therefore, during the image feature matching step, we list all the potential matching grid pairs across the images. For each such grid pair  $(g_i, g_j)$ , we perform the symmetry guided feature matching step as explained in Section 4.2.2. Specifically, we evaluate all the candidate alignments corresponding to different shifts in rows and columns

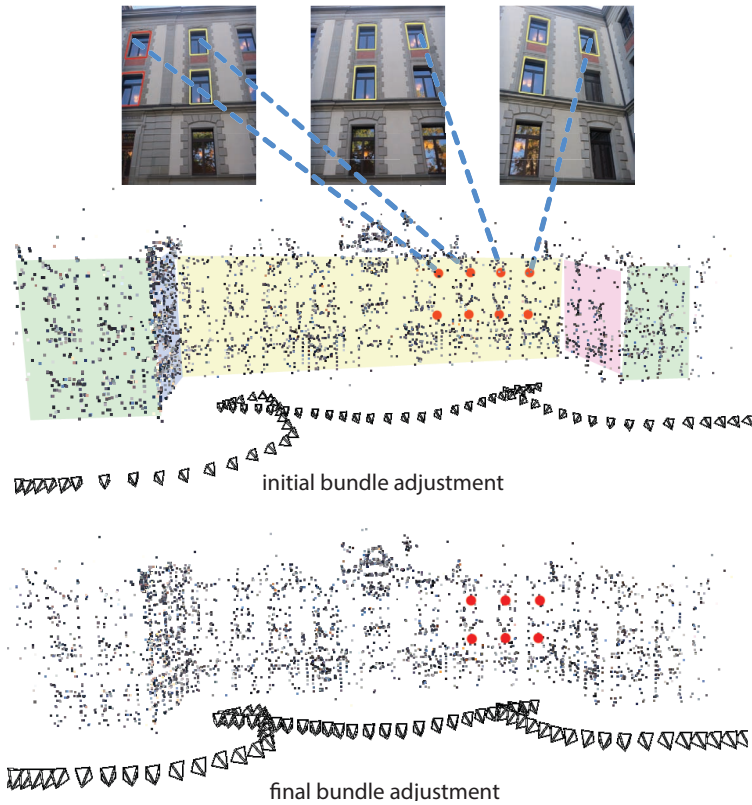


Figure 4.6: Once we obtain the initial 3D representation, we refine the rectification of the input images and repeat the symmetry-based SfM step. In the initial image matching step, the windows in red have been grouped together with the windows in yellow (resulting in a 2-by-4 grid) but have been discarded due to high projection error (resulting in a 2-by-3 grid).

between the grids  $g_i$  and  $g_j$ . Once the candidate alignments have been evaluated for all the grid pairs, we pick the alignment with the highest support. Using the feature matches that support this selected alignment, we detect the remaining matching grids in the images ( $I_i, I_j$ ) that are the projection of the same 3D grid and find the corresponding row/column shifts between them. We add the selected alignment to the matching graph  $G$  as edge  $e_{ij}$  encoding the grid shifts between all the matching grids between the corresponding images.

Later, in the iterative optimization setup, for each cycle in the image matching graph, we consider the accumulated alignments between all the common grids in the images participating in the cycle. More specifically, for a cycle between the images  $I_i, I_j$ , and  $I_k$ , grids  $g_i \in I_i, g_j \in I_j$ , and  $g_k \in I_k$  are common grids if the edge  $e_{ij}$  encodes a grid shift between  $g_i$  and  $g_j$ ,  $e_{jk}$  encodes a grid shift between  $g_j$  and  $g_k$ , and  $e_{ki}$  encodes a grid shift between  $g_k$  and  $g_i$ . If the accumulated grid shifts between any such common grid do not cancel out, the cycle is marked as inconsistent. Similarly, if we swap an alignment represented by an edge with an accumulated alignment along the shortest paths in  $G$ , we update all the grid shifts represented by this edge between the common grids in the images involved in the path.

At the end of the iterative optimization, the final alignments computed for each image pair  $(I_i, I_j)$  encode the correct matching image grids and the shifts between them. Using these alignments, all the image grids matched to each other across the input images are grouped together where each group represents the image projections of the same 3D grid. For each 3D grid, we organize the grid correspondences as grid tracks. We update the bundle adjustment objective function to include a grid energy term for each of the  $g$  3D grids and refine the parameters of each 3D grid with this bundle adjustment step:

$$E = \sum_1^g E_{grid} + E_{other}. \quad (4.6)$$

### 4.3 Evaluation

We evaluate our framework on several datasets with varying complexity of the underlying symmetries (see Figure 4.7 and 4.8). We provide a complete set of results in Appendix A and list the performance statistics of the proposed method in Table 4.1. We now summarize our main findings.

	$N_i$	res	$N_r$	$T_s$ (mins)	$T_o$ (mins)	Ours	Bundler	Zach et al. 10
Building 1	26	6.2	28	10	5	yes	poor	no
Building 2	27	7.7	42	15	2	yes	conf.	poor
Building 3	26	3.5	10	8	2	yes	yes	yes
Building 4	24	7.7	44	6	1.5	yes	yes	yes
Building 5	25	6.2	22	20	1.5	yes	conf.	poor
Building 6	32	6.2	22	40	1	yes	yes	poor
Building 7	51	6.2	101	45	6	yes	mult..	mult.
Building 8	72	6.2	36	50	8	yes	no	mult.
Building 9	13	6.2	0	4	1	no	no	no
Building 4 <i>lowRes</i>	24	0.5	44	1	0.5	yes	no	conf.

Table 4.1: The table shows the number of input images ( $N_i$ ), the resolution of the images in megapixels (res), and the total number of repeating elements detected ( $N_r$ ) for each data set. We also report how our method, Bundler, and the method of Zach et al. perform: a correct reconstruction is produced (*yes*), the output is poor in quality (*poor*), there is a confusion in the number of repeated elements (*conf.*), or reconstruction contains multiple misaligned components (*mult.*). The computation times for image-based symmetry detection ( $T_s$ ) and a single iteration of symmetry-based SfM ( $T_o$ ) are given in minutes measured on a 2.8 GHz 4-core machine.

**Comparisons.** In our proposed framework, we explicitly detect repeating elements in the input images and use this information both to extract reliable image correspondences and estimate camera parameters accurately. We compare this approach to a standard SfM pipeline [103] and

## Chapter 4. Symmetry and Structure-from-Motion



Figure 4.7: For each example, we provide a sample input image, the user marked template in a single image (orange), the extracted repetition pattern and the calibrated cameras. This information is used for a range of image manipulations.

the inference-based ambiguity detection method of Zach et al. [122], which also has been used as an initializer in the followup work of Cohen et al. [20]. To illustrate the effect of accurate camera pose estimation for dense reconstruction, we use a state-of-the-art multi-view stereo method [33] to produce dense reconstructions of the input scenes using the camera parameters estimated by each of the methods. We use the EXIF tags of the images to estimate the internal camera parameters in all three cases. In Table 4.1, we report how these three methods perform on each data set marking the output based on if: (i) a correct reconstruction is obtained, (ii) output is poor in quality, (iii) wrong number of repeated elements is reconstructed, or (iv) multiple sub-models corresponding to different subsets of the input images are reconstructed. Our algorithm accurately extracts the camera parameters in most of the examples leading to accurate dense reconstructions, while Bundler [103] or the approach of Zach et al. [122] fail or often produce



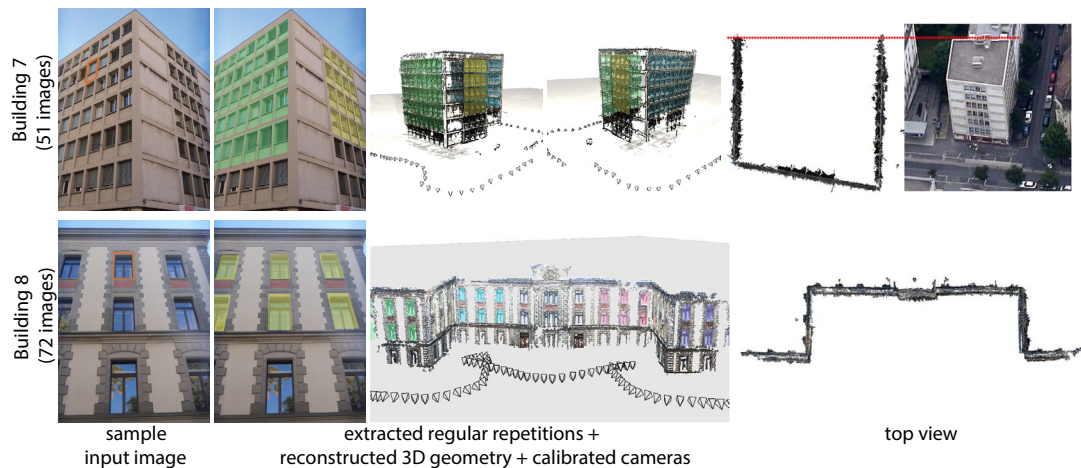


Figure 4.8: Our approach handles buildings with multiple facades while preserving the orientation of the individual facades both for orthogonal (*Building 8*) and non-orthogonal (*Building 7*) relations. We provide a satellite imagery of *Building 7* for reference.

sparser reconstructions. We provide detailed comparisons in Appendix A.

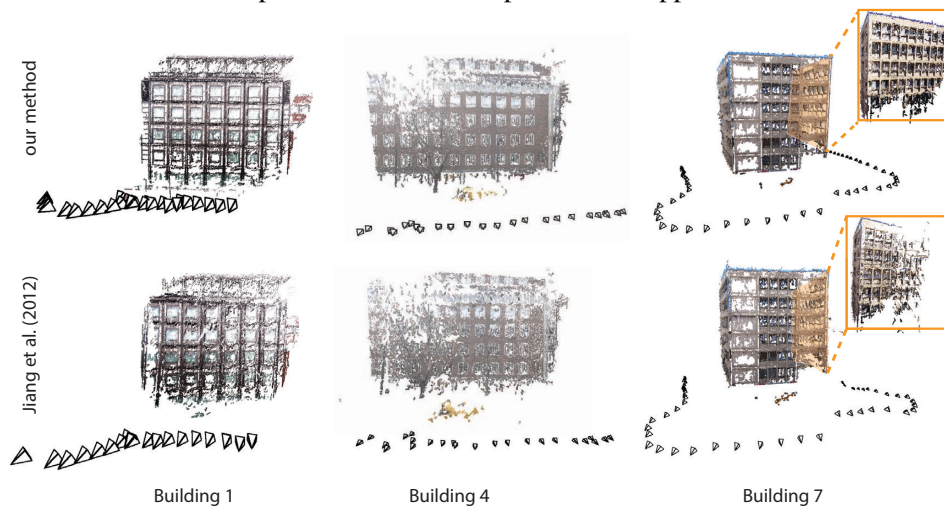
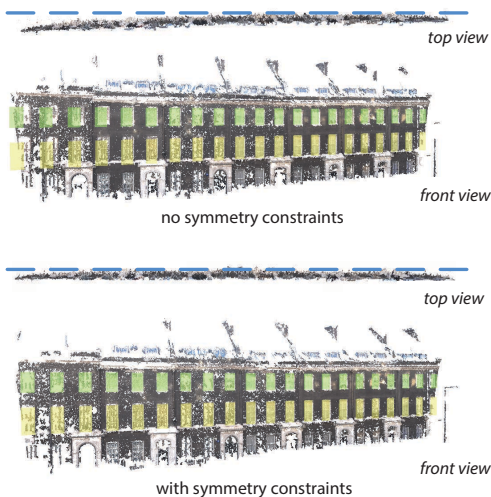
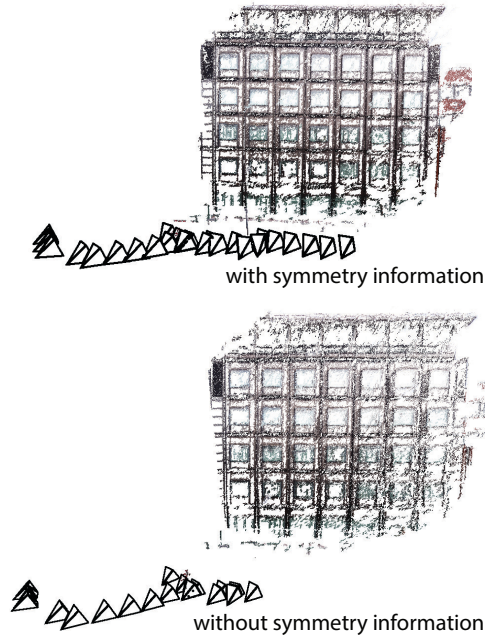


Figure 4.9: For the *Building 1*) data set, the method of Jiang et al. [48] registers 21 out of 26 images. For the *Building 7* example, our method produces significantly higher-quality output especially for the right facade of the building highlighted in orange.

We also compare our method to the recent approach of Jiang et al. [48] that formulates the image matching problem as finding the spanning tree of the image matching graph minimizing a global energy function (see Figure 5.10). They propose a greedy search algorithm that resolves an important portion of the image matching ambiguities. They do not explicitly model any particular form of symmetry or repetition. In contrast, by injecting symmetry priors into every step of the reconstruction pipeline, our method effectively resolves the remaining ambiguities and significantly improves the quality of the reconstructions. Further, in contrast to all other methods,

our approach produces the repetition patterns directly as part of the output.

**Effect of symmetry.** Explicit encoding of the extracted symmetry information enables our method to effectively evaluate the reliability of the alignments between the input image pairs. In the symmetry-guided feature matching step, we first list all the candidate alignments between an image pair, and then discard the potential ambiguous matches. We observe that the remaining sparse set of feature matches arising due to non-repeating regions and the random variations in the non-discarded repeating regions (such as ornaments, weathering, etc.) often provide sufficient support for the correct alignments. Moreover, during the global optimization performed on the matching graph, we iteratively improve the quality of the alignments by discarding the discovered inconsistencies. The inset figure illustrates the effectiveness of the grid constraints during the iterative graph optimization step. We compare our results to the case where consistency of the image alignments are evaluated based on the corresponding rotations only. We observe that in the latter case some ambiguities remain unresolved and only a subset (18 of 26) of the input image set is registered.



In the inset figure, we illustrate the effect of using additional symmetry constraints in bundle adjustment on the quality of the final reconstructions. Since we work with a repetitions arranged as planar grids, explicitly enforcing the symmetry relations across the grid correspondences acts as a regularizer and significantly reduces drift, especially in long image sequences. Unconstrained solution produces a distorted facade (blue dotted line shown for reference) with the repeated elements drifting from the correct solution. Further, our algorithm successfully recovers the correct orientation of the individual facade planes of a building without any additional assumption on the orientation relations like orthogonality (see

Figure 4.8).



**Robustness to low-resolution inputs.** Our symmetry-aware correspondence search makes the approach robust to degrading image resolution. To illustrate this, we tested our framework on an image set at two resolutions (see Figure 4.10). Both Bundler and the method of Zach et al. [122] performed poorly in the low-resolution setting because a significant amount of features are only seen in the high-resolution images due to small random variations in the facade elements. However, our method extracts the correct relations among the input cameras using the sparse feature set by exploring the initially detected repetitions in the images.

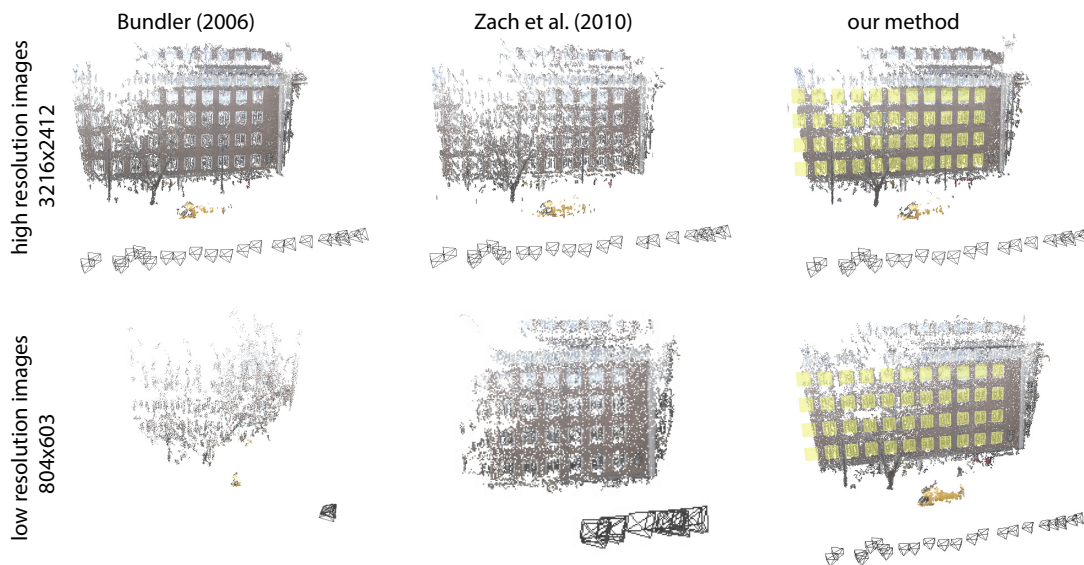


Figure 4.10: For the *Building 1* data set, the method of Jiang et al. [48] registers 21 out of 26 images. For the *Building 7* example, our method produces significantly higher-quality output especially for the right facade of the building highlighted in orange.

**Limitations.** Although we handle a range of diverse data sets, our approach still suffers from various limitations. We assume facade elements to be repeated along 1- or 2-parameter regular grids and do not handle rotational symmetries as found on domes, churches, etc. In certain cases repeated elements have non-uniform gaps with sufficiently small variations that cannot be recovered neither in image space nor the 3D reconstruction step. As a result, image-based regular structure detection might fail.

Even though the symmetry-guided correspondence search and the iterative global optimization improves the quality of the image alignments significantly, we do require a sufficient amount of discriminating features to bootstrap the process. In case of insufficient discriminating features, our method will fail to resolve all the ambiguities (see Figure 4.11).

Although we do not require the repetition grids to be visible in full from any single image, we do expect a reasonable overlap between the images so that we get a connected solution graph. In the absence of sufficient overlap, the solution graph can have multiple components leading

to multiple reconstructions. While we did not encounter such a problem in our examples, we believe that further investigations to better characterize the requirements on the input images are necessary.

We focus on facades with dominant facade planes. If facade elements show significant depth variations, image-based repetition detection cannot be performed reliably [47]. While we didn't observe serious artifacts, our 3D reconstructions capture limited depth information, especially around sharp features and statues, etc., which limits the scope of subsequent editing possibilities.

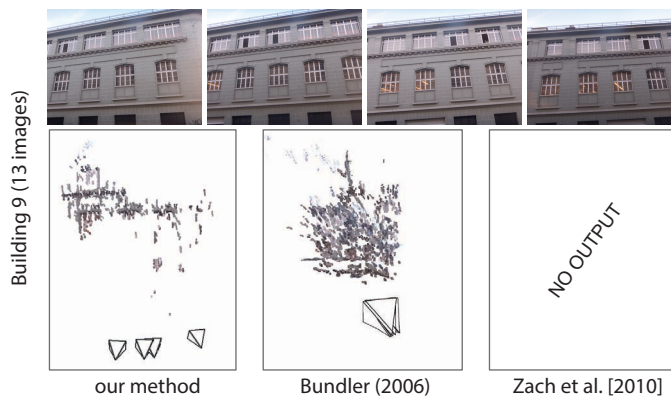


Figure 4.11: For the *Building 9* data set, due to lack of sufficient discriminating feature matches our method fails to resolve the ambiguities. The dense reconstructions computed with the camera parameters obtained from our method and Bundler are shown. The method of Zach et al. [122] does not produce any camera parameters.

## 4.4 Applications

Once we register the set of input images, the extracted symmetries of the scene allow the user to perform various image editing applications, while the changes are automatically propagated to all the images. Besides demonstrating various use cases, these applications are important to highlight the robustness and accuracy of the reconstruction algorithm as the applications heavily rely on precise symmetry boundaries.

**Occlusion Removal.** Often street-level images of building facades contain many foreground objects such as street lights, trees, and cars, which partially occlude the background facade plane. Redundancy in the form of multiple images and repeated elements in the input set allows us to synthesize seamless textures to remove such occlusions.

Given a set of input images  $(I_1, \dots, I_n)$ , we require the user to provide rough strokes on a single reference image  $I_i$  to denote the occluding object that is to be removed. We use GrabCut [91] to extract an accurate mask  $B_i$  for the occlusion area. If the occluder object has delicate structures, like tree leaves, we require the user to provide more refined strokes. Typically, each pixel in this

bounded region occludes a point that lies on the main facade plane. Our goal is to synthesize the texture for this occluded area of the facade plane, which we approximate by using the extracted 3D grid information. We back-project each pixel  $p \in B_i$  in the occlusion region to the facade plane to obtain the 3D position  $\mathbf{q}$  of the corresponding occluded point. We then project  $\mathbf{q}$  to the other images to find a set of candidate pixel colors that can be used to replace  $p$  in the reference image. Once a set of candidate colors has been computed for each pixel, the desired texture is synthesized by estimating a label  $L(p) \in [1, \dots, n]$  for each pixel  $p$  denoting which image should be used as the source color for  $p$ . We formulate this texture synthesis problem as a Markov Random Field (MRF) optimization by minimizing an energy function consisting of data and smoothness terms. The data term measures the cost of assigning the label  $L(p)$  to pixel  $p$ :

$$E_{data}(p, L(p)) = |I_{L(p)}(p) - \rho|,$$

where  $I_{L(p)}$  denotes the color of pixel  $p$  in the labeled image and  $\rho$  is the median of the candidate colors for pixel  $p$ . To reduce the seams in the synthesized texture, we define a smoothness term to assess the labelings of neighboring pixels (we consider 4-connected neighbors) by evaluating the color and the gradient differences similar to Sinha et al. [101]:

$$E_{smooth}(p, q, L(p), L(q)) = \begin{cases} 0 & \text{if } L(p) = L(q) \\ d_I + d_G & \text{else} \end{cases}$$

where  $d_I = |I_{L(p)}(p) - I_{L(q)}(p)| + |I_{L(p)}(q) - I_{L(q)}(q)|$  and  $d_G = |\nabla I_{L(p)}(p) - \nabla I_{L(q)}(p)| + |\nabla I_{L(p)}(q) - \nabla I_{L(q)}(q)|$ . We use the graph cut algorithm [12] to solve this optimization problem. Once we choose the source images for each pixel in the synthesized texture, we use Poisson blending [86] to further reduce the seams across the occlusion boundaries in the original image (Figure 4.12).

Importantly, once the user-marked occluding area  $B_i$  is removed from the reference image, we can propagate the edit and remove the occluder also from the other images. To enable this operation, we need depth estimates for the occlusion object in the reference view to determine its projection in the other images. Since the occluding object lies in front of the main facade plane, we construct a depth volume bounded by the reference camera position and the depth of the facade plane and use an MRF energy minimization approach to optimize for the depth of each pixel in the occluded area. We formulate the data term of the MRF energy based on NCC matching costs of image patches (see Campbell et al. [2008]):

$$E_{data}(p, x) = e^{-\beta\{p, x\}}, \tag{4.7}$$



Figure 4.12: Occlusion mask marked in one image (top-left) is propagated to the other images. Occlusion removal using propagated information from other images provides significantly improved results compared to the single-image based state-of-the-art *PatchMatch* [8] method.

where  $x$  denotes a discrete depth sample inside the depth volume.  $\beta\{p, x\}$  denotes the average NCC matching score of the image patches generated by projecting the 3D point obtained by assigning the depth  $x$  to  $p$  to the other views. We define the smoothness energy as the sum of the depth deviation between a pixel and its 4-connected neighbors [Szeliski et al. 2008]. Similar to Sasaki et al. [2006], we define the data term in a multi-resolution framework to increase the accuracy of the depth estimates. In other words, we start with a coarse sampling of the depth volume, and iteratively generate finer samples around the current depth label for each pixel. Once the depth estimates for each pixel in the reference occluding region are computed, this region is projected to the other images and the texture is synthesized in these projected areas as described before (see Figure 4.12).

**Grid Editing.** Our framework extracts 2D/3D symmetry information of the input scene in the form of planar repetition grids. This information allows the user to directly manipulate the grids such as changing the repetition count in the grid as previously shown by Wu et al. [2010b] or editing the appearance of the grid elements on a single image. Our system automatically propagates the changes to the other images as the relation between the 2D image grids and the global 3D grid is already computed. Technically, we first extract accurate boundaries of the repeating elements by snapping their initial contours to the image edges. Similar to the reconstruction framework presented in the previous chapter, we then optimize for the common contour by a line fitting approach. In contrast, we only optimize for the contour lines as we have already extracted the refined 3D grid generators. Once the contour of each grid element is computed, we change the number of repetitions in the 3D grid by keeping the boundaries of the grid fixed and appropriately scaling the grid elements. This amounts to editing only the relevant parts of each image where the 3D grid projects (see Figure 4.13). We synthesize the texture



for the new grid elements by scaling the original elements. To minimize seams and lighting variations, for each new grid element, we use the texture from the spatially closest original grid element. In case of occlusion, we require the user to roughly annotate the occluding object, while we perform necessary image completion as described before. The occluding object is encoded as a separate layer and later composed with the edited images.



Figure 4.13: Extracted facade grid patterns (left) are changed and then composited with the foreground (e.g. lamp post). These changes are propagated to the other images.

We would like to emphasize that these editing applications are only as powerful as the detected symmetries. Also, lack of sufficient depth information can lead to artifacts near 3D elements on the original images (e.g., window ledges, etc.). Thin structures (e.g., tree branches, leaves) can be difficult to annotate in the images and also are challenging to propagate across images since corresponding 3D points are few and sparse. A complete 3D reconstruction framework is necessary to obtain the depth of such structures.

## 4.5 Closing Remarks



We have presented a structure-from-motion framework that detects and conforms to structural regularities, while simultaneously recovering 3D geometry. A novel graph based global analysis yields a globally consistent 3D geometry reconstruction with explicit encoding of the facade regularities. These regularities can then be used for a range of novel image manipulations, while maintaining consistency across the images.

Even though we have focused on planar repetition grids, a natural extension is to incorporate rotational symmetries as found on arches, domes etc.

## Chapter 4. Symmetry and Structure-from-Motion

---

With the growing demand for simple, fast, and accurate acquisition methods for digital cities, we expect to see increasing research efforts in this direction. Specifically, we believe that simultaneously editing image collections brings up exciting opportunities for interactive and dynamic interfaces. We illustrate a first result where the extracted scene geometry along with symmetry information can be used towards new editing possibilities such as inserting synthetic objects as shown in the inset figure. Advanced appearance matching methods ([124]) can potentially be used for better color blending.

## 5 Understanding Structured Variations

In the previous chapters, we have explored symmetry priors in both stages of a traditional image-based reconstruction pipeline. We have demonstrated that use of such priors simultaneously improves the reconstruction quality and provides semantic information about the acquired data. Focusing on more modern style buildings, however, we made strong assumptions on the type of symmetries to detect. We concentrated on regularities arranged as 1- or 2-dimensional planar grids. Even though these type of regularities are among the most common type of symmetries for modern building facades, they are not sufficient to capture the type of structural relations we see in other types of buildings.



Figure 5.1: For many buildings, similar elements can be arranged by varying transformations, located at different facades of a building, and exhibit certain structured variations.

To begin with, for many buildings, similar elements are not necessarily found in planar grid arrangements; they can possibly be related with varying transformations on the same facade, located at different facades of a building, or linked by rotations. A more important observation is that, certain repeating elements might exhibit structured variations. We observe such variations commonly in historic ornate buildings, for example window elements might have the same shape but vary in height (see Figure 5.1.) Exploring such structural similarities from raw and noisy data measurements as obtained in multi-view stereo (MVS) reconstructions is challenging. On the other hand, once detected these similarities enable a high-level understanding of the underlying

geometry and thus provide effective priors for augmenting the MVS algorithms.

We investigate this problem by utilizing a set of template models of element types commonly found in buildings such as windows. The advantages of using templates are two fold. First, each template is equipped with a deformation model that allows to capture the variations of a base geometry. Second, buildings designed with the same architecture style or serve the same functionality often resemble similarities among their elements. Thus, the style of the target buildings can be taken into consideration during the selection of the templates to capture such similarities.

Given such a set of templates, we formulate similarity detection as a labeling problem. Our goal is to identify the best matching template for each element and compute the corresponding deformation parameters. Patterns extracted in the resulting deformations reveal element similarities. This approach enables us to make a general definition of similarity: elements deformed from the same template with similar deformation parameters are similar. This general similarity definition eliminates the need for making prior assumption on the spatial arrangement of similar elements and the type of symmetry relations that exist among them.

A naive approach to detect such similarities is to first perform an independent template fitting for each element and then detect similarities in the resulting template deformations, e.g. with a standard clustering method. However, noise and outliers in MVS reconstructions prevent robust template fitting. Hence, such an independent analysis fails to detect reliable element similarities. Instead, we present a coupled approach where our goal is to simultaneously perform template fitting and extract patterns in the resulting template deformations. We repeat template fitting by consolidating data across elements based on the detected patterns. Performing this analysis iteratively reveals which elements are exact replicas of the same geometry and which share partial similarities. The extracted similarities provide semantic information about different parts of the elements and their relationships. This facilitates high-level analysis, but also paves the way to generative modeling and structure-aware editing.

There has been prior work on exploring the use of templates during the reconstruction process. In an early attempt, Dick et al. [25] use a set of parameterized part templates to create 3D building models from single images. Subsequently, Schindler et al. [96] fit segmented image measurements to a set of predefined shape templates to create CAD-like 3D facade reconstructions; Pauly et al. [83] warp selected models from a database of 3D shapes and combine suitable parts towards object completion; Chen et al. [18] propose an interactive setup to lift freehand sketches to 2.5D using a database of architectural elements. Typical man-made objects have also been used for indoor scene understanding [53, 78]; while, Bao et al. [7] use anchor points to deform and fit shape templates to poor quality MVS reconstructions. All these approaches, however, either assume that exact shape templates are available, or only allow limited deviations (i.e., warps) from the templates. Furthermore, multiple templates are fitted independently across the scene making it challenging to ensure robustness for partial inputs or limited template database. In contrast, we propose to couple the template fitting via the extracted deformation parameters.



## 5.1 Overview

We present a coupled template matching and deformation algorithm to understand structural variations between the elements of a building. In a one-time pre-processing stage, we construct a set of template models of common element types, e.g. windows. Each such template is characterized by a set of deformation parameters that define variations of the model (see Figure 5.4). Our algorithm takes as input a set of images  $\mathcal{S} = \{I_1, \dots, I_n\}$  of a building and computes its 3D MVS reconstruction. The user roughly indicates elements of interest such as a window frame in *one* of the input images. For each of these elements, we identify the best matching template and compute the best fitting deformation of this template, which we call a *template instance*. A key element of the proposed algorithm is to reveal geometric similarities among the elements by detecting patterns in the deformation parameters of their matching template instances.

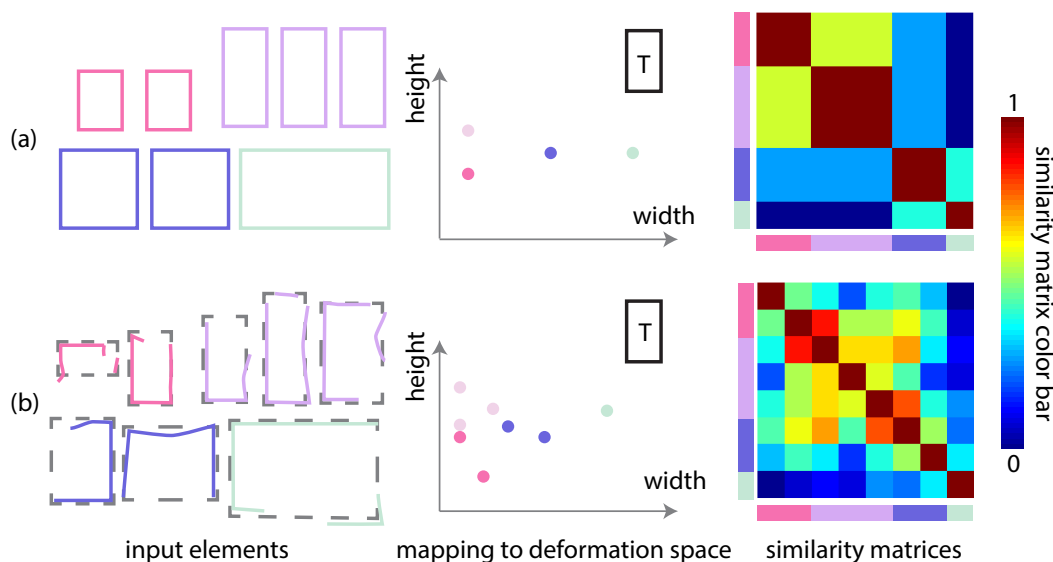


Figure 5.2: (a) In case of perfect input data, elements with the same geometry are mapped to a single point in the 2-dimensional deformation space of a rectangular template  $T$ . (b) The presence of noise and missing data, however, makes it challenging to observe clear clusters in the deformation space. Similarity matrices computed using the pairwise element distances in the deformation space reveal this behavior.

Typically, we observe two types of relations between given elements. First, elements that are replicas of the same geometry should be matched to the same instance of the same template. Second, elements that exhibit variations of a base geometry, e.g. windows with the same arch but varying height, should be matched to different instances of the same template. In this case, deformation parameters defining the instances will be partially the same. An intuitive approach for detecting such relations is to compute the matching template instance for each element independently and then detect patterns across the resulting deformation parameters. Assume a template is parameterized with  $k$  deformation parameters. Each instance of this template is represented as a point in a  $k$ -dimensional deformation space with nearby points representing

instances with similar deformation parameters. In the ideal case, elements with the same geometry will map to a single point in this space. Elements that are variations of a base geometry, however, will map to nearby points (see Figure 5.2a). This is revealed in similarity matrices computed based on pairwise element distances in the deformation space (see Figure 5.2, right). Elements that are exact replicas are represented as red blocks whereas elements with partial similarities are represented by colors closer to red.

In case of real data, however, due to noise and partial data even elements that are exact replicas are often matched to different instances of the same template (see Figure 5.2b) or, even worse, different templates. As a result, similar elements map to scattered points in the template deformation space, making it challenging to observe clear clusters.

While robust template deformation is necessary to detect element similarities, consolidating information across similar elements improves template deformation. However, neither the template deformations nor element similarities are known upfront. Hence, we propose a coupled algorithm to simultaneously find the matching template instances while detecting similarities in the corresponding deformation parameters (see Figure 5.5). We deform a set of templates to fit the elements. We combine observations from template deformations to map each element to a common subspace representation. Intuitively, similar elements are expected to map to nearby points in this subspace resulting in small pairwise element distances. Using these distances as constraints, we consistently label each element with a deformed template instance. We repeat template deformation by consolidating data across elements matched to similar template instances. Iterating between these steps progressively brings similar elements closer in the common subspace and reveals clear clusters as if in the perfect input data case. Being independent of the specific choice of the deformation model, this iterative analysis exposes which elements are exact replicas and which share partial similarities.

## 5.2 Algorithm Details

We first describe the template deformation model we have adopted in our evaluations and then provide the details of the proposed iterative approach.

### 5.2.1 Template Models

Buildings designed with similar architectural style or serving related functionalities often exhibit significant similarities in their individual elements such as windows. Thus, in our analysis we use a set of template models of such element types. Each template is equipped with a deformation model that defines structured variations of a base geometry.

Since architectural data sets consist of dominant line features, we adopt a deformation model that preserves the salient feature lines of the templates similar to the iWires framework [34]. In a pre-processing stage, we extract feature lines, called wires, in each template based on the dihedral

angles between the edges (see [34]). Each feature wire is a collection of atomic wires that can be one of the *straight line*, *circular*, or *elliptic arc* types (see Figure 5.3). Each atomic wire is defined with a set of parameters such as the length and the direction of a straight line, or the center, radius, and the opening angle of a circular arc. Each template is parameterized with the union of the parameters of its atomic wires.

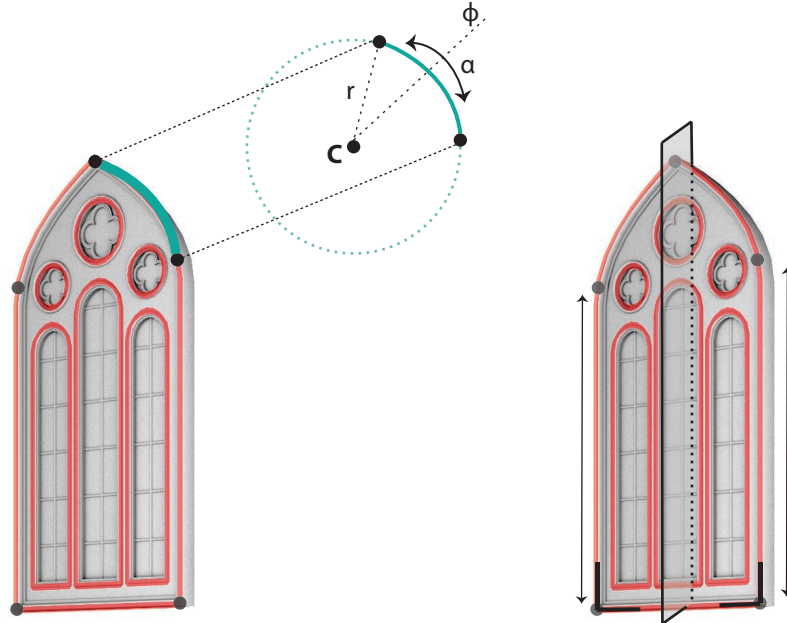


Figure 5.3: The feature lines of the given template model are shown in red. Each feature line consists of individual atomic wires characterized by a set of parameters. For example, circular atomic wires are characterized by the center ( $C$ ), radius ( $r$ ), mid-angle ( $\phi$ ), and the opening angle ( $\alpha$ ) of the arc. We detect orthogonality, equal-length, and reflection constraints among the feature wires of the model (right).

When deforming a template to fit a given element, our goal is to find the parameters of the atomic wires of the template while preserving certain structural relations between them. We identify a set of inter- and intra-wire relations characteristic to the type of templates we utilize (see Figure 5.3). Specifically, for a compound wire consisting of several atomic wires, we identify sets of atomic wires that have equal length. We further identify pairs of adjacent atomic wires that are connected orthogonally. Finally, we detect planar wires where all atomic wires lie on a common plane. During template deformation, we preserve these equal length, orthogonality, and planarity properties of the atomic wires. In addition, many architectural elements exhibit global reflectional and rotational symmetries. Thus, in the pre-processing stage, for each axis-aligned template model, we detect the presence of any reflectional symmetry with respect to one of the three axis planes and discrete rotational symmetry with respect to one of the three axes. During deformation, we enforce the symmetric atomic wires to share the same parameters adjusted according to the symmetry relation.

Once the parameters of each atomic wire are computed, we adopt a 3D volumetric deformation

approach to obtain the final geometry of the template model. Specifically, we construct a regular 3D deformation grid,  $V_g$ , for the template model and sample corresponding points on the original and the updated wires of the model. Using these correspondences as handle constraints, we solve for the new width, height, and depth of each cell in  $V_g$  with respect to smoothness constraints on the size of the neighboring grid cells (see [56] and [81]). Once the updated grid is computed, the geometry of the deformed template is constructed by preserving the local coordinates of each vertex of the model with respect to its enclosing grid cell. Such a volumetric deformation enables us to handle templates that may not be watertight meshes.

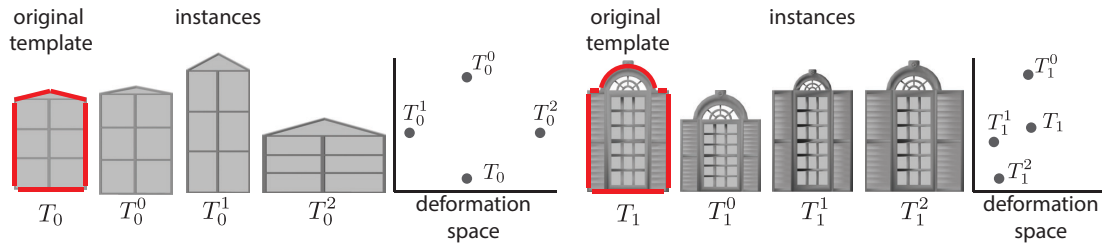


Figure 5.4: For the templates  $T_0$  and  $T_1$ , we illustrate various instances  $(T_0^0, \dots, T_0^2, T_1^0, \dots, T_1^2)$  with different parameters of the detected feature wires (shown in red). Each instance is represented as a point in the deformation space of the corresponding template based on these parameters. A multi-dimensional scaling projection of the deformation space of the templates is shown.

A deformed instance of a template can be represented by the collection of the parameters of its individual atomic wires. Such a set of parameters can be used to map varying instances of a template to its deformation space. However, having parameters that represent different quantities, e.g. the length of a straight line and the opening angle of a circular arc, a direct comparison of these parameters is difficult. Therefore, we instead use a geometric realization of these parameters. Particularly, we translate each straight line such that its starting point coincides with the origin of the Cartesian coordinate system and represent the atomic wire with its endpoint. Similarly, we translate circular and elliptic arcs so that their centers lie at the origin. We then represent a circular arc with the two endpoints of the arc and an elliptic arc with four points sampled at equal angles. By stacking these points, we define a descriptor vector  $\mathbf{v}$  for each instance of a template (see Figure 5.4). We compare different template instances by computing the Euclidean distance between the corresponding descriptors.

### 5.2.2 Template Fitting

**Pre-processing of the Input.** Given an input set of images  $\mathcal{I} := \{I_1, \dots, I_n\}$  of a building, we first compute the camera parameters for each image using the VisualSFM tool [115] and a 3D reconstruction of the scene with PMVS [33], a state of the art MVS algorithm. To perform template deformation based on salient line features of the scene, we perform standard image-space edge detection on each individual image  $I_i$  and apply multi-view stereo matching on the 2D edges [6]. The resulting set of 3D line segments,  $\mathcal{L}^3 := \{l_1, l_2, \dots\}$ , drives template deformation

by establishing correspondences between these segments and the feature wires of templates.

Our goal is to detect geometric similarities between the elements of a building. Even though there exist automatic facade parsing methods exploiting the presence of horizontal and vertical splitting lines and 2D regular grids (see [76]), we observe that such methods fail to identify the elements of more complex architectural scenes. Therefore, we instead adopt the semi-automatic approach discussed in Chapter 4 which requires the user to only roughly mark an element of interest, e.g a window frame, in *one* of the input images and automatically detects its repetitions. However, standard image-based similarity methods might fail to detect elements exhibiting strong variations or in presence of large occlusions and appearance changes. In these cases, we revert to user input to mark the missing elements.

We identify the 3D points whose projection falls into each marked or detected image region and fit a bounding box to these points. In the remaining of the text, we refer to each such subset of the point cloud as an *element*. We align the orientation of the bounding boxes of the elements to the common up direction of the MVS reconstruction (see [117]). In the rest of the pipeline, we consider the elements to be provided as input and do not assume any prior information obtained through the element selection process.

**Template Deformation.** For a given template  $T_j$  and an element  $s_i$ , our goal is to compute the deformation parameters,  $\mathbf{d}_j^i$ , that define the instance of  $T_j$  that match  $s_i$  as closely as possible. We first apply a similarity transformation to  $T_j$  that aligns the bounding boxes of  $T_j$  and  $s_i$ . We axis-align all templates in the pre-processing stage so that the  $y$ -axis corresponds to the up direction. We compute correspondences between the axes of the element and template bounding boxes by matching their up directions.

Once  $T_j$  and  $s_i$  are roughly aligned we setup an optimization to compute  $\mathbf{d}_j^i$ . Our goal is to align the corresponding feature lines of  $T_j$ , i.e. wires, and  $s_i$ . Thus, we sample 3D points both on the feature wires of  $T_j$  and the 3D line segments falling inside the bounding box of  $s_i$  to produce the sample point sets  $Q_j$  and  $P_i$  respectively. We establish 3D correspondences between these point sets and minimize the distance between them:

$$c(\mathbf{d}_j^i) = \sum_{\mathbf{q}_j \in Q_j} \|\mathbf{q}_j - \mathbf{p}_i\|^2 + \sum_{\mathbf{p}'_i \in P_i} \|\mathbf{p}'_i - \mathbf{q}'_j\|^2, \quad (5.1)$$

The first term measures the distance from the template to the element where  $\mathbf{q}_j$  and  $\mathbf{p}_i$  are corresponding points in  $Q_j$  and  $P_i$  respectively. The second term measures the symmetric distance from the element to the template where  $\mathbf{p}'_i \in P_i$  and  $\mathbf{q}'_j \in Q_j$  denote the correspondences.

We solve for  $\mathbf{d}_j^i$  by minimizing the following energy:

$$E_{fit}(T_j, \mathbf{d}_j^i, s_i) = c(\mathbf{d}_j^i) + \alpha c^R(\mathbf{d}_j^i) + c^W(\mathbf{d}_j^i), \quad (5.2)$$

where  $c^R(\mathbf{d}_j^i)$  is added as a regularization term ( $\alpha = 0.01$ ) to minimize the difference between the

current and initial sample points on the feature wires of the template. The term  $c^W(\mathbf{d}_j^i)$  is used to preserve the relations, e.g. planarity, orthogonality etc, between the wires of  $T_j$  (see [34]). We solve this optimization iteratively, updating the sample point set  $Q_j$  and the 3D correspondences at each iteration. We use the Ipopt package [110] to solve the non-linear optimization.

Having defined how a desired template is deformed to fit an element, our next goal is to identify the best fitting template instance for each element as described next. We note that this analysis is independent of the adopted deformation model.

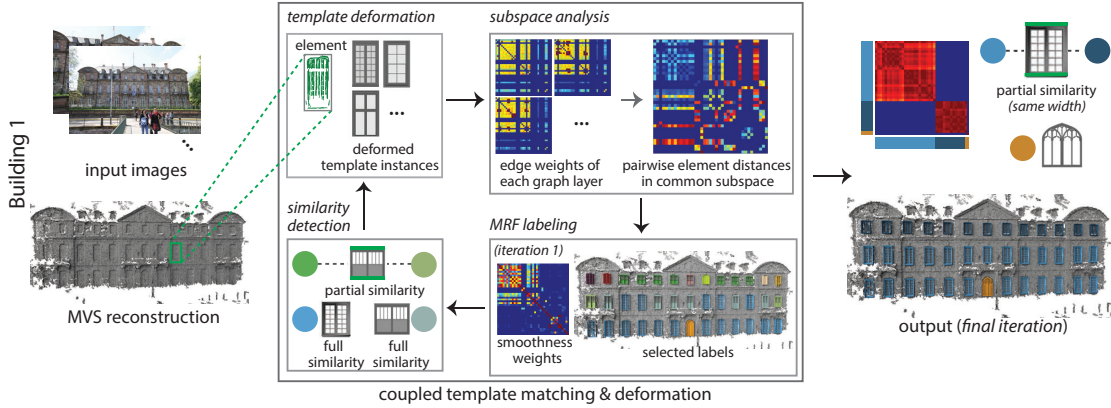


Figure 5.5: Given a MVS reconstruction of a building, we utilize a set of templates to match its elements, i.e. windows. We combine observations from template deformations via a subspace analysis to extract element relations. Using these relations as constraints, we label each element with a deformed template instance (same instances are denoted in same color). We repeat template deformation by consolidating data across elements matched to similar template instances. This analysis reveals elements that are identical (represented as red blocks in smoothness weight matrices) or share partial similarities (highlighted in green on the matching templates).

### 5.2.3 Coupled Template Matching and Deformation

Given a set of elements  $\mathcal{S} := \{s_i\}$  and a set of templates  $\mathcal{T} := \{T_j\}$ , our first goal is to label each  $s_i$  with the tuple  $(T^i, \mathbf{d}^i)$  where  $T^i$  is best the matching template and  $\mathbf{d}^i$  represents the deformation parameters of the fitting instance. We then aim to detect patterns in these resulting deformation parameters that represent geometric similarities between the elements. Once detected, such similarity relations allow to consolidate data across multiple elements. This consolidation progressively improves template deformation and enables extraction of more reliable similarities.

**Subspace Analysis.** We begin our analysis by deforming each template  $T_j$  to fit each element  $s_i$  to yield the deformation parameters  $\mathbf{d}_j^i$ :

$$\mathbf{d}_j^i = \operatorname{argmin}_i \sum E_{fit}(T_j, \mathbf{d}_j^i, s_i) + E_{sim}, \quad (5.3)$$

where  $E_{fit}(T_j, \mathbf{d}_j^i, s_i)$  measures how well template  $T_j$  fits  $s_i$  as explained in the previous section.

$E_{sim}$  measures the difference between the deformation parameters of  $T_j$  detected as being similar across multiple elements and will be discussed shortly. Initially  $E_{sim}$  evaluates to 0 since we assume no prior knowledge about element similarities.

The computed template deformations provide observations about the geometry of the given elements. We represent such observations in a multi-layer graph  $M$  where each individual graph layer,  $G_j = (S, E_j, W_j)$ , encodes the deformation parameters obtained by fitting the template  $T_j$  to each element (see Figure 5.5). Elements are represented as nodes in each graph layer where the edges  $E_j$  between the nodes are weighted by the vector  $W_j$  computed based on the distances between the corresponding deformation parameters. An edge  $e_{ik} \in E_j$  connecting the nodes  $s_i$  and  $s_k$  is weighted by:

$$w_{ik} = e^{-D(\mathbf{T}_j^i, \mathbf{T}_j^k)/m_D} \cdot E_{fit}^{(i,j)}, \quad (5.4)$$

where  $D(\mathbf{T}_j^i, \mathbf{T}_j^k)$  measures the distance between the instances of  $T_j$  that are matched with  $s_i$  and  $s_k$ .  $m_D$  denotes the maximum of such distances and is used for normalization. As discussed in the previous section, a direct comparison between the deformation parameters is challenging as each of these parameters denotes a different quantity. We instead measure the distance between two instances of a template by computing the Euclidean distance between their descriptor vectors  $\mathbf{v}_j^i$  and  $\mathbf{v}_j^k$ . We multiply the edge weights with the template fitting errors,  $E_{fit}^{(i,k)} = \min(E_{fit}(T_j, \mathbf{d}_j^i, s_i), E_{fit}(T_j, \mathbf{d}_j^k, s_k))$ , to diminish the effect of possible local minima resulting from imperfect template matches, e.g due to noise.

Each graph layer in  $M$  captures different observations of the relations between the elements obtained from the corresponding template deformations. Our goal is to combine the information revealed at each layer of  $M$  to extract a set of consistent relations. We achieve this goal by adopting the subspace analysis approach of Dong et al. [26] which uses ideas from spectral clustering theory. This approach first computes a subspace representation of each graph layer. This is achieved by computing the normalized Laplacian matrix,  $L_j$ , of each graph layer  $j$ :

$$L_j = D_j^{-1/2}(D_j - A_j)D_j^{-1/2}, \quad (5.5)$$

where  $D_j$  and  $A_j$  denote the degree and the adjacency matrices of the graph at layer  $j$  respectively. A meaningful subspace representation  $U_j$  of the vertex connectivity at each graph layer is then represented by the  $k$ -dimensional spectral embedding of the graph, i.e., the smallest  $k$  eigenvectors of the corresponding graph Laplacian [66]. Ideally,  $k$  represents the ground truth number of clusters between the graph vertices and this  $k$ -dimensional embedding enforces the clusters to become more visible. However, the information about the number of element clusters is not available in our case and thus we choose the value of  $k$  based on the *eigengap* heuristic. We select the first  $k$  eigenvalues where the gap between the consecutive eigenvalues ( $e_{k+1} - e_k$ ) is maximized. While this low dimensional subspace information is traditionally used for finding

clusters of the graph vertices, it is also useful for analysis tasks as in our case.

Once a subspace representation  $U_j$  is computed for each graph layer, the goal is to effectively combine these subspaces into a common representation  $U$ .  $U$  is desired to be close to all the individual subspaces and at the same time preserve the information about vertex connectivity in each graph layer. Closeness between  $U$  and each  $U_j$  is measured as the squared projection distance between them [26]:

$$d_{proj}^2(U, \{U_j\}) = \sum_{j=1}^N d_{proj}^2(U, U_j), \quad (5.6)$$

where  $N$  denotes the number of graph layers. We refer the reader to the work of Dong et al. [26] for a more detailed discussion about the projection distance between two subspaces. At the same time, we want  $U$  to preserve the information about vertex connectivity in each graph layer. This is achieved by defining a trace minimization problem by drawing insights from the spectral clustering theory [79]:

$$d_{tr} = \sum_{j=1}^N \text{tr}(U^T L_j U) \text{ s.t. } U^T U = I, \quad (5.7)$$

where  $L_j$  is the graph Laplacian at layer  $j$ . Thus, finding a common representative subspace  $U$  accounts to solving the following minimization problem:

$$U = \text{argmin } \alpha d_{proj}^2(U, \{U_j\}) + d_{tr}, \quad (5.8)$$

where  $\alpha$  ( $\alpha = 0.5$  in our examples) balances the trade-off between the two terms. As discussed by Dong et al. [26], it turns out that the solution  $U$  of the above minimization problem can be obtained by first forming a common graph Laplacian  $L = \sum_{j=1}^N L_j - \alpha \sum_{j=1}^N U_j U_j^T$  and then defining  $U$  as the smallest  $k$  eigenvectors of  $L$ . Once computed, each element is represented in this low dimensional subspace  $U$ . This representative subspace summarizes the information captured in each graph layer and reveals the intrinsic relations between the elements that resemble their similarities. However, these relations do not directly map to actual labels of the elements. Hence, in a subsequent step, we solve a labeling problem by incorporating the relations captured in  $U$  as constraints.

**Labeling problem.** Each element  $s_i$  is represented as  $s'_i$  in the common low-dimensional space  $U$  obtained from the multi-layer graph  $M$ . Elements that share geometric similarities appear as



nearby points in  $U$ . Thus, pairwise element distances in this space (see Figure 5.5) provide an indication of their degree of similarity. Our goal is to find a consistent labeling for elements that have small pairwise distances.

Recall that we want to label each element  $s_i$  with a tuple  $(T^i, \mathbf{d}^i)$  consisting of both the matching template and the deformation parameters defining the matching instance. Even though we have a set of discrete labels with respect to the template type, the deformation parameters are defined in a continuous deformation space. A naive approach to discretize the space of possible labels is to represent each template deformation space with a sparse set of samples. However, it is challenging to obtain a good coverage of the large deformation space with a sparse set of samples while ensuring the samples are sufficiently close to the instances matching with the given elements. On the other hand, the deformation parameters obtained by fitting templates to elements provide a good set of initial labels. Therefore, we formulate this labeling problem as a Markov Random Field (MRF) optimization [24] where the label set  $\mathcal{L} = \{(T_j, \mathbf{d}_j^i)\}$  consists of the current set of template instances obtained by fitting each template  $T_j$  to each element  $s_i$ . (We choose the best 5 template instance for each element in our experiments.) The MRF optimization consists of data, smoothness, and label costs:

$$\sum_{s_i} E_d(s_i, L(s_i)) + \sum_{s_i, s_k} \alpha_{ik} E_s(s_i, s_k, L(s_i), L(s_k)) + \sum_{T_j} \lambda_{T_j} E_L. \quad (5.9)$$

The data term,  $E_d(s_i, L(s_i))$ , measures the cost of fitting the template instance defined by the label  $L(s_i)$  to the element  $s_i$ . The smoothness term,  $E_s(s_i, s_k, L(s_i), L(s_k))$ , evaluates the consistency of the labeling of each pair of elements. If two labels  $L(s_i) = (T^i, \mathbf{d}^i)$  and  $L(s_k) = (T^k, \mathbf{d}^k)$  belong to the same template ( $T^i = T^k$ ), the smoothness term measures the distance between the two corresponding instances of the template,  $D(\mathbf{T}^i, \mathbf{T}^k)$ . Otherwise, a fixed smoothness cost (two times the maximum distance between any two instances of the same template in our experiments) is assigned. The smoothness weights  $\alpha_{ik} = e^{-\|s'_i - s'_k\|^2 / \sigma^2}$  are determined from the distances between the elements mapped to the representative subspace  $U$ . Intuitively, similar elements have small pairwise distance in  $U$  resulting in high smoothness weights (revealed as colors closer to red in pairwise smoothness weight matrices shown in Figure 5.5 and 5.12). These elements are expected to be assigned to similar labels, i.e. instances of the same template. The constant  $\sigma > 0$  determines how rapidly the smoothness weight drops with increasing pairwise element distances and is set to 0.1 in our experiments. The last term in Equation 5.9 penalizes each unique template that appears in the final labeling. Specifically, we group the candidate labels coming from the same templates into subsets and a fixed label cost  $E_L$  (equal to  $1/5^{th}$  of the average data cost in our experiments) is induced if at least one label is used from such a subset. The indicator variable  $\lambda_{T_j}$  is set to 1 if an instance of the template  $T_j$  appears in the final labeling.

Due to noise and partial data, elements that are exact replicas often map to scattered points in the representative subspace  $U$  instead of a single point. The smoothness term progressively enforces these elements to be assigned to the same label and thus brings them closer. This behaviour is revealed as the formation of red blocks in the pairwise smoothness matrices in the

final iterations of our algorithm (see Figure 5.12). If there are similar templates, similar elements might get assigned to instances of different templates. The label cost favors the use of as few unique templates as possible and thus enables a consistent labeling. Both smoothness and label costs enforce the selection of fewer templates. Hence, elements that exhibit variations of a base geometry are matched to different instances of the same template.

**Similarity Detection.** The formulated MRF optimization assigns a label to each element consisting of the matching template and the deformation parameters of the fitting instance. We evaluate these labels to extract a set of similarity relations  $R = \{r\}$  between the elements. In particular, if elements  $s_i$  and  $s_k$  are matched to two instances of template  $T_j$ , we define the relation  $r = (s_i, s_k, \mathbf{c}_j, T_j)$ .  $\mathbf{c}_j$  is a binary vector which contains a 1 for the descriptor components of these template instances that have a Euclidean distance below a certain threshold (0.5% of the length of the diagonal of the bounding box of the input reconstruction). These components indicate partial similarities between  $s_i$  and  $s_k$  with respect to template  $T_j$ . A vector  $\mathbf{c}_j$  consisting of all ones ( $\mathbf{c}_j = \mathbf{1}$ ), on the other hand, indicates that  $s_i$  and  $s_k$  are exact replicas. In this case, we define an additional relation for these elements for all other templates with  $\mathbf{c}_j = \mathbf{1}$ .

Once we extract a set of relations,  $R$ , we repeat template deformation while using these relations as additional constraints. When deforming a template  $T_j$ , we minimize the energy given in Eqn. (5.3):

$$\mathbf{d}_j^i = \operatorname{argmin}_i \sum_i E_{fit}(T_j, \mathbf{d}_j^i, s_i) + \underbrace{\sum_{(r) \in R} E_{dist}(r)}_{E_{sim}}, \quad (5.10)$$

where  $E_{dist}(r) = \mathbf{c}_j^T (\mathbf{v}_j^i - \mathbf{v}_j^k)^2$  for  $r = (s_i, s_k, \mathbf{c}_j, T_j)$  measures the difference between the coupled descriptor components of the instances of  $T_j$  that fit  $s_i$  and  $s_k$ . This coupling consolidates data across multiple elements by enforcing the coupled atomic wires of a template to deform similarly. With the new deformation parameters, we update the multi-layer graph  $M$  and repeat the subspace analysis and the MRF optimization.

We iterate between these steps until no change is observed in labels of the elements (typically 5-6 iterations). Intuitively, at each iteration we improve template fitting by consolidating data across multiple elements via the detected similarity relations. This creates an enhanced candidate label set for the subsequent MRF optimization. In addition, potential element clusters in template deformation spaces become clearer as similar elements are progressively pulled closer (see Figure 5.6).

**Extension to Large Template Sets.** When performing the proposed coupled analysis, it is possible to consider every template-element pair for deformation, especially if the number of utilized templates is low. However, as the size of the template set grows, a pre-organization of the templates becomes necessary. This organization not only reduces computational complexity but also provides high-level relations among the templates that can be used to guide the template matching process.

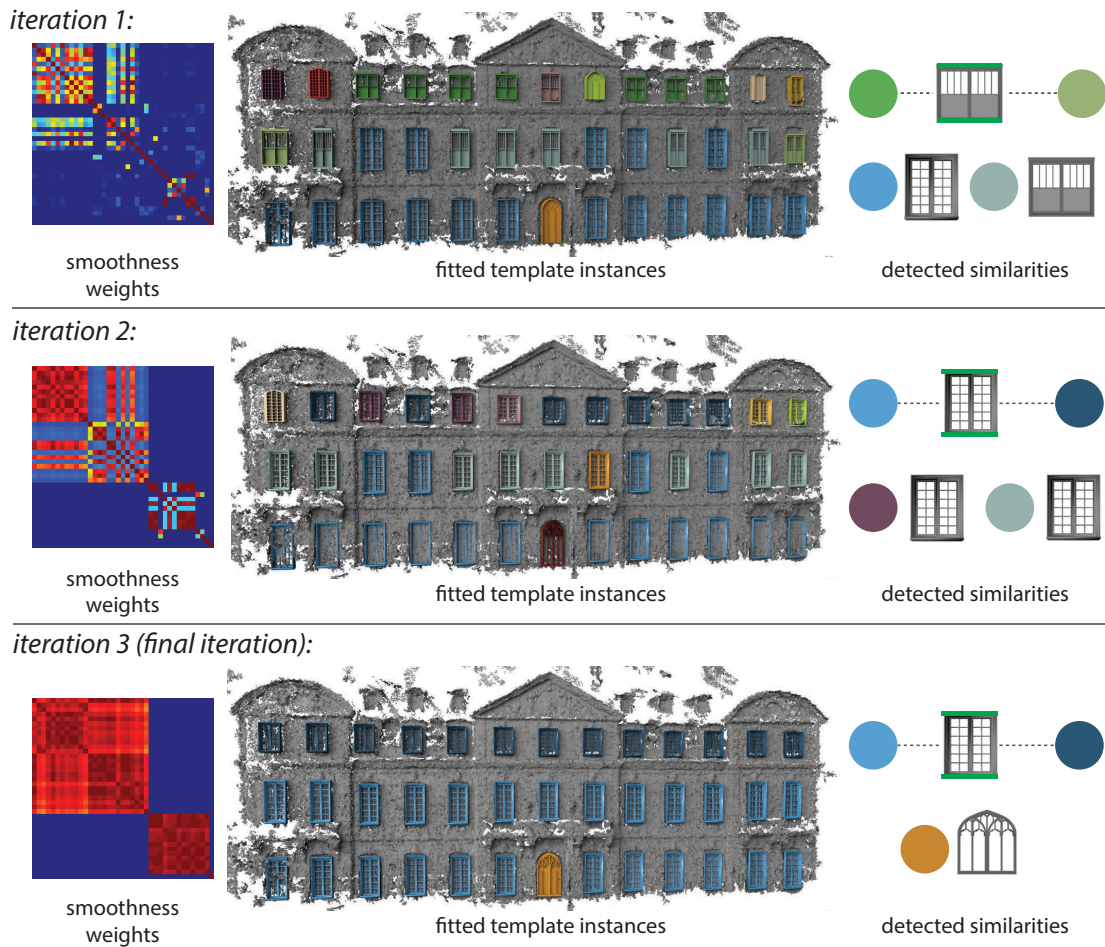


Figure 5.6: We propose an iterative approach to simultaneously find the matching template instances of a given set of elements and compute similarities among them. Intuitively, at each iteration we improve template fitting by consolidating data across multiple elements via the detected similarity relations. Thus, potential element clusters become clearer as observed by the formation of red element blocks in the smoothness matrices.

In our experiments, we propose a simple strategy to group the templates based on their deformation capabilities. We deform each pair of templates  $T_i$  and  $T_j$  to fit the other and define a pairwise distance measure equal to the maximum of the resulting fitting errors. When measuring the fitting errors, we only use the correspondences between the outer wires of the templates which are visually more discriminating compared to the inner substructures. We then cluster the templates based on these pairwise distances. For window templates, this simple strategy successfully distinguishes between circular, arch, and rectangular windows but leads to confusion between arch and triangle-top windows. We revert to user input to regroup the few misclassified templates resulting in four groups of templates (see Appendix B). We note that it is possible to propose a different organization of the templates, e.g based on architecture style.

Given a grouping of the templates, for each element we find the best fitting group and deform only the templates in this group. Initially, for each element-template pair, we compute a fitting error after aligning their bounding boxes via a similarity transformation. We identify the matching template group of an element as the group with the minimum average of such fitting errors. Since each group of templates are deformed only to a subset of the elements, we perform the subspace analysis independently for each such subset by combining the observations from the corresponding template deformations. For every pair of elements mapped to the same subspace, we compute a pairwise smoothness weight as explained before. For any pair of elements mapped to different subspaces, we set the corresponding smoothness weight to 0. We note that each element is represented by normalized coordinates when mapped to a subspace. Hence, the smoothness weights between the elements mapped to different subspaces are comparable. During labeling optimization, we construct a global candidate label set for all the elements irrespective of their matching template groups. Thus, an element may possibly be matched to a template instance that does not belong to its current matching group. This results in an update of the matching group of the element in the next iteration of the algorithm. This update mechanism enables to recover from any errors that might occur during the initial group matching.

### 5.3 Evaluation

We evaluate our algorithm on several synthetic and real datasets with varying complexity and data quality. In our evaluations, we focus on window elements as these are the most common element type that exhibit full and partial similarities. We use a template set consisting of 60 window models downloaded from the Digimation Model Bank and Trimble 3D Warehouse. The templates and their detected feature wires can be downloaded from the following link (*removed for the review process*). We classify the templates into 4 groups of *arch*, *rectangle*, *triangle-top*, and *circular* windows (see Appendix B).

We demonstrate how our algorithm performs when changing the amount of variation across input elements, the number of utilized templates, and the data quality. In order to assess the effect of each factor independently, we perform some of these evaluations on synthetic data annotated with ground truth 3D line features. We next discuss each of these experiments in detail.

**Effect of element deformations.** One of the main steps of our algorithm is to perform a labeling optimization that matches each element to a template instance. This optimization is driven by data and smoothness terms. While the data term evaluates the individual label assignments, the smoothness term favors similar labels for similar elements. Due to this coupling, the amount of variation among the elements plays a significant role in determining the final label selection. We illustrate this effect on a set of synthetic elements created by gradually increasing the variation among them (see Figure 5.7). When using a set of three templates including a template capable of capturing all of these variations, all elements are labeled with different instances of this template (Figure 5.7a), i.e. the smoothness term has no effect. When we remove this template, however, none of the remaining templates is capable of perfectly capturing the element variations and the

labeling is affected by the smoothness relations. We first consider the first six elements, where four of them prefer the first template based on data term only. Even though the fifth and sixth elements prefer the second template based on data term, the smoothness relations among the elements enforce them to pick labels from the first template (Figure 5.7b). Nonetheless, addition of two more elements that also prefer the second template individually is perceived as a strong indication that the second template is also a likely assignment. Thus, the last three elements are now assigned to labels from the second template (Figure 5.7c).

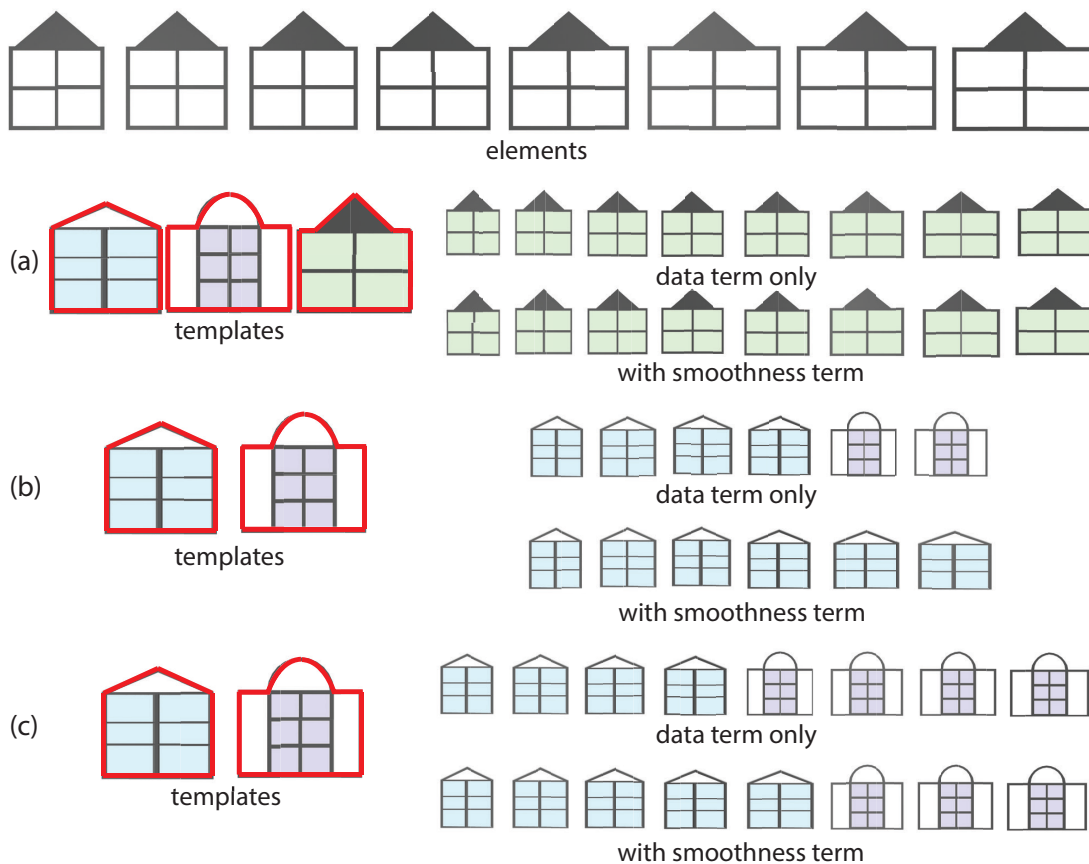


Figure 5.7: Due to the coupling introduced by the smoothness term, the amount of variation among the elements affects the final selection of template instances. For different set of templates (with feature wires shown in red) and elements, we show the selected template instances once based on data term only and in the second row additionally considering the smoothness term.

**Effect of number of templates.** A unique feature of our algorithm is to combine observations from multiple template deformations. Since each template represents variations of its base geometry, combining observations from multiple templates leads to a better coverage of the template deformation space. Moreover, templates deform similarly to fit similar elements. Thus, regardless of their suitability, each template contributes to detection of element similarities. We illustrate this effect on a synthetic house model annotated with ground truth 3D line features. This model has two types of windows showing variation in height and width respectively (see

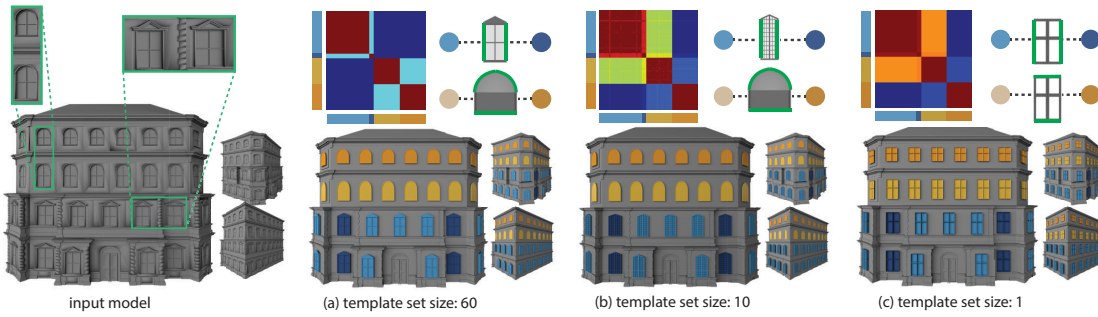


Figure 5.8: We show the selected template instances for a synthetic house model (consisting of 38 narrow triangle-top, 4 wide triangle-top, 23 long arch, and 23 short arch windows) when using different number of templates. For each case, we also show the color-coded smoothness matrices and the partial similarities detected between the elements (highlighted in green). Note that the removal of the triangle-top template selected in (a) results in a selection of another triangle-top window in (b).

Figure 5.8). We run our algorithm on this model using an increasing template set size of 1, 10 (selected templates are shown in Appendix B), and 60 (templates organized as four groups). We show the pairwise element smoothness weights computed in each case in color-coded matrices. For each block of identical elements, the side color bars denote the color of the corresponding matching template instance. We show the partial similarities detected between such template instances in a graph by highlighting the coupled parts of templates in green. We observe that even a single template is capable of distinguishing the variation among the elements resulting in the selection of four distinct template instances (Figure 5.8c). With additional templates, the two window types are also identified resulting in the selection of two instances of each template (Figure 5.8b). When using a grouping among the templates (Figure 5.8a), the two type of window elements are initially matched to different template groups which results in no smoothness relation between them, i.e. blue blocks in the corresponding smoothness matrix. With a single rectangular template, however, the similarity between the height of the triangle-top and long arch windows is reflected as a reasonably high smoothness weight, i.e. orange block in the corresponding matrix.

**Effect of data quality.** The quality of the detected feature lines in the input has a direct impact on template deformations. We analyze this effect by comparing the performance of our algorithm on synthetic data using ground truth feature lines (Figure 5.8a) and a MVS reconstruction obtained from the rendered images of the model (Figure 5.9). We observe two main sources of error that potentially influence our results. First, due to the challenges in correspondence search, MVS reconstructions cause a *general* degradation in data quality which might lead to failure in capturing fine details. Even though our algorithm selects the same templates as in the ground truth case, it fails to capture the subtle variation in the width of the two instances of triangle-top windows (Figure 5.9a). Second, factors such as large occlusions affecting specific regions of the input result in *local* degradation in data quality. Thus, when we place a large tree model in front of the house, our algorithm cannot recover the correct template assignments for the windows



occluded by this tree (Figure 5.9b).

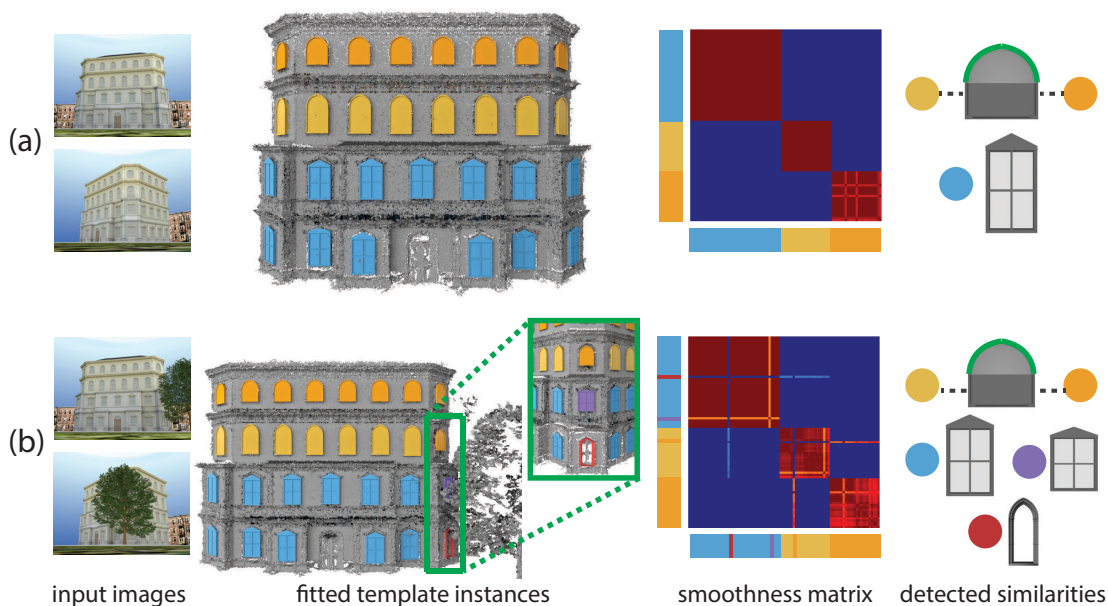


Figure 5.9: We evaluate our algorithm on MVS reconstructions obtained from rendered images of a synthetic model. Due to loss of fine details, we cannot recover the subtle variation in width of the triangle-top windows in blue (a) and the occlusion by a large tree results in wrong template assignments for some elements (b).

**Comparison to naive clustering.** Due to noise and partial data, an independent analysis of each element often results in the selection of different templates for elements that are derived from the same base geometry (Figure 5.10a, left). Even if we annotate the correct template selection, elements that are exact replicas are mapped to scattered points in the template deformation space. Thus, standard clustering algorithms such as *k-means* fail to identify the correct element clusters (Figure 5.10b, left). Our coupled analysis, however, progressively consolidates data across elements via the detected similarities. This coupling brings similar elements closer in the template deformation space and thus reveals distinct clusters (Figure 5.10b, right). For this dataset, compared to ground truth clustering by visual inspection, the clustering of our algorithm achieves a mutual information score [109] of 0.876 with one mislabeled element due to lack of sufficient 3D line features (see Figure 5.11), whereas the clustering based on independent analysis has a score of 0.468.

**Performance on real data.** We evaluate our algorithm on various real datasets with different architecture style and varying complexity (Figure 5.5, 5.11, and 5.12). Table 5.1 shows the statistics of our algorithm on each dataset. We now summarize our main findings, while we refer the reader to Appendix C for a more complete set of results.

We introduce a general definition of similarity by discovering patterns in the deformation modes of the template instances fitting a set of elements. Our algorithm not only handles elements

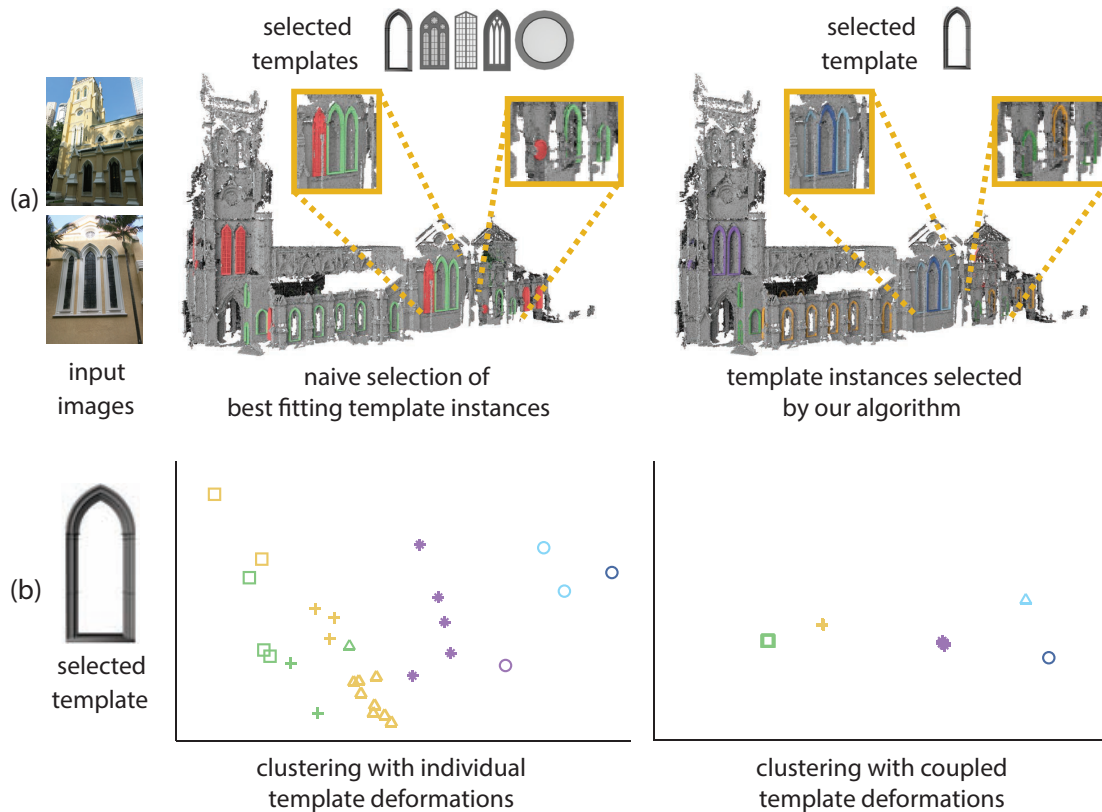


Figure 5.10: (a) Individual template fitting for a set of elements results in the selection of 5 different templates whereas our algorithm assigns the elements to 5 different instances of the same template. (b) Given a template selection, we visualize each element in the low-dimensional deformation space of the template (elements that are exact replicas are shown in same color) using the deformation parameters obtained by individual fitting vs. our algorithm. The clusters generated by the k-means ( $k=5$ ) algorithm are indicated by different symbols.

that are replicas of the same geometry but also detects similarities across elements that exhibit partial variations. Figure 5.11 and 5.12 illustrate many partial similarities detected among the elements which would have been difficult to capture otherwise. For example, for *Dataset 1*, our algorithm discovers 5 instances of the same template for 30 window elements (Figure 5.11) whereas individual template fitting for each element results in the selection of 5 different templates (Figure 5.10).

We perform coupled template matching and similarity detection solely based on the deformation parameters of the templates. We make no assumption about the presence of any specific type of regularity or spatial arrangement such as 2-dimensional grids. Yet our analysis successfully detects similarity relations between elements that are rotationally symmetric (Figure 5.12, *Dataset 7*), arranged as 1-dimensional grids (Figure 5.12, *Dataset 6*), and located across different facades of a building (Figure 5.12, *Dataset 5*). Even though noise in MVS reconstructions makes it difficult to initialize transform domain grid fitting as proposed by Pauly et al. [84], matching



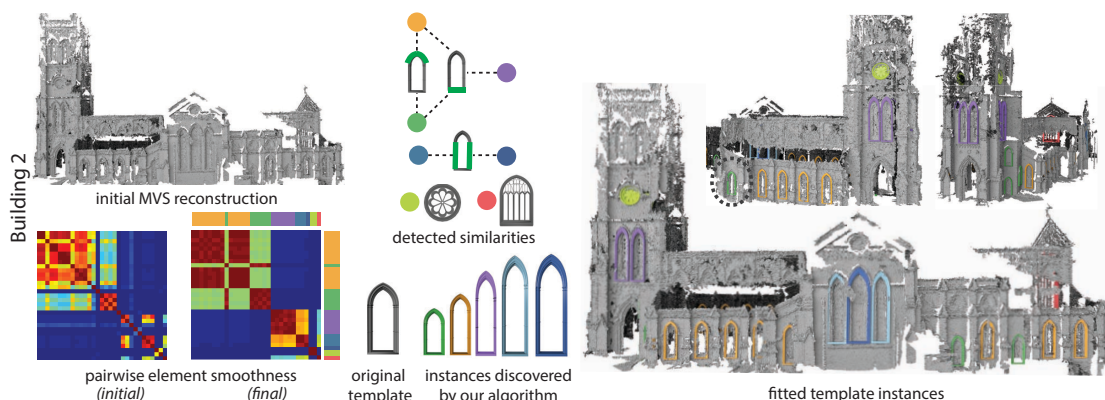


Figure 5.11: Color bars at the sides of the final smoothness matrix denote the color of the fitting template instances of the corresponding block of identical elements. We show the partial similarities detected across such element blocks in the accompanying graph by indicating the identical parts of the matching template instances in green.

template instances as detected by our algorithm enables the discovery of grid-like spatial arrangements between the elements. We use such relations to spatially snap the template instances in our results.

	$N_i$	$N_s$	$N_e$	$T_d$	$T_c$	$T_i$
Building 1	60	3	39	10	2	3
Building 2	120	7	32	8	3	7
Building 3	160	8	32	12	4	10
Building 4	299	10	99	22	7	9
Building 5	70	13	25	7	5	9
Building 6	129	6	57	13	3	4
Building 7	126	7	17	7	4	8

Table 5.1: The table shows the number of input images ( $N_i$ ), the number of user selected elements ( $N_s$ ), the total number of detected elements ( $N_e$ ), the numbers of templates selected by the independent analysis ( $T_d$ ) and with the coupled analysis ( $T_c$ ), and the total number of template instances discovered ( $T_i$ ).

As discussed previously, MVS reconstructions often suffer from general degradation in data quality which might lead to failure in capturing fine details, e.g. in the substructures of the elements. As this degradation is typically *consistent* across different elements, our coupled analysis is still able to recover the correct similarity relations among the elements. However, missing details or the lack of a more suitable template might result in selection of a template different than a user-intended one. For example, for *Dataset 4*, the closest available template has been selected to capture the two-arch structure of the windows shown in green (see Figure 5.12). As the similarity relations among the elements have been correctly identified at this stage, it is straightforward to introduce user interaction possibilities such as providing a new template. In this case the user-provided template will be deformed to the given elements while preserving the

## Chapter 5. Understanding Structured Variations

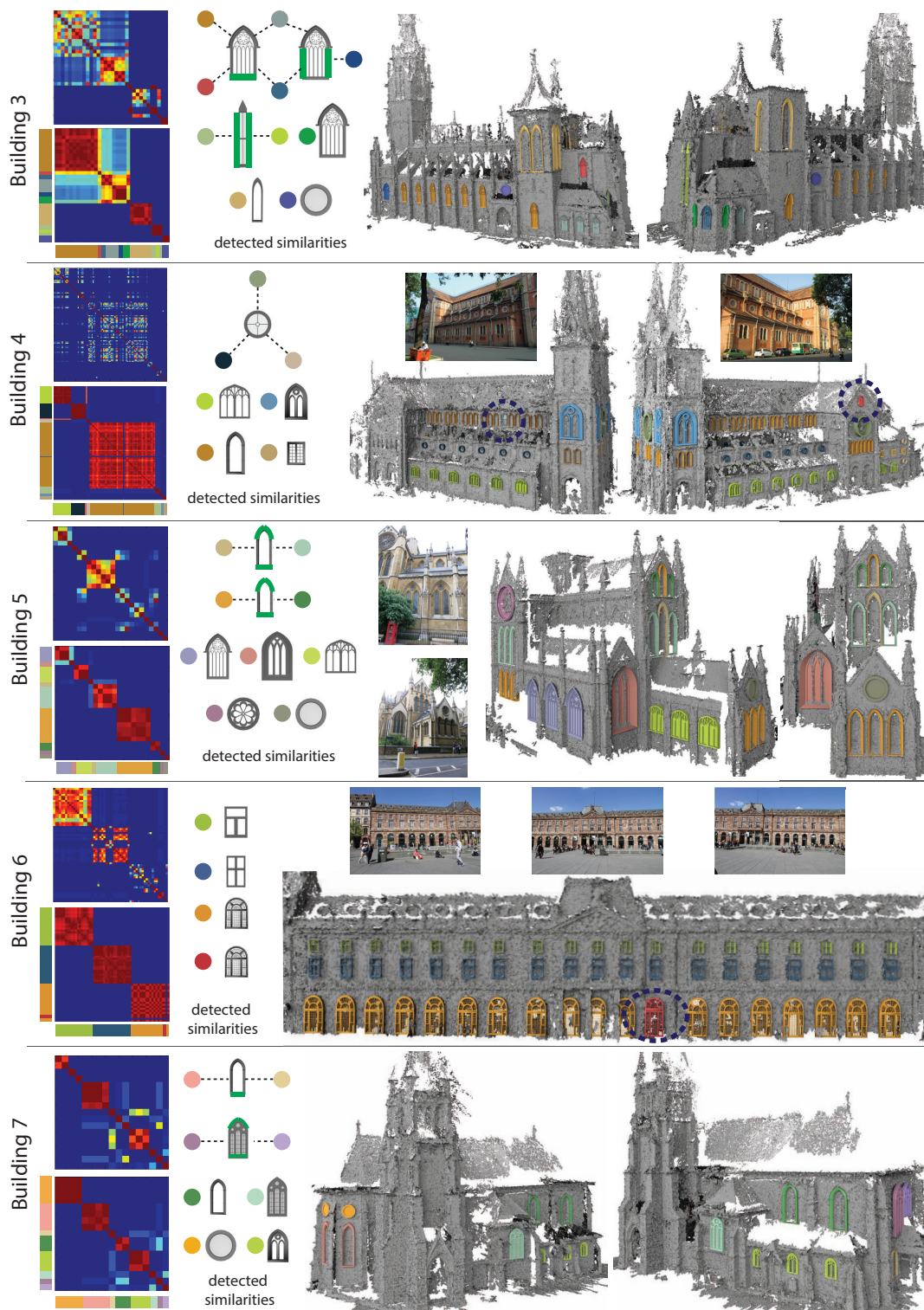


Figure 5.12: For each example, we show the smoothness matrices in the first (top left) and final (bottom left) iterations of our algorithm. Color bars at the sides of the matrices denote the color of the matching template instances of the corresponding block of identical elements. Partial similarities detected across element blocks are shown on the corresponding templates in green. We denote the elements matched to wrong template instances with dotted circles.

already detected similarity relations between them.

Even though our coupled analysis successfully recovers the full and partial element similarities in most cases, severe local degradations in data quality, e.g. due to large occlusions, may result in failures. In our examples, we indicate such erroneous elements by dotted ellipses (Figure 5.11 and 5.12). Such errors occurred due to insufficient 3D lines, which in turn prevented reliable template deformation. In Figure 5.13, we demonstrate two challenging cases, where almost half of the indicated windows are occluded. Even though our algorithm identifies the correct template, it discovers the wrong (shorter) instance. It is possible to augment our analysis with additional priors to resolve such failures. Possible solutions include incorporating smoothness constraints among elements arranged in a grid (Figure 5.13, top) or down weighting the distance from the template to the element when computing the fitting error (Figure 5.13, bottom). Automatic detection of such large occlusions, e.g. by analyzing the point distribution in the neighborhood of elements is an interesting future direction.

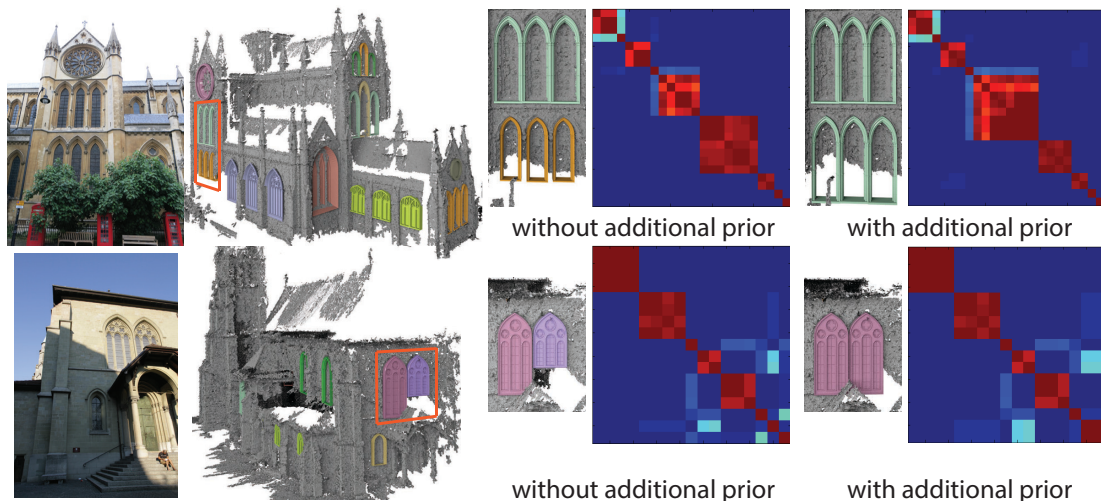


Figure 5.13: Due to the lack of sufficient 3D lines resulting from large occlusions, our algorithm fails to match the indicated windows (in orange) to the correct template instances. It is possible to augment our analysis with additional priors to resolve such failures. We show the element smoothness matrices with and without use of such priors.

## 5.4 Closing Remarks

We have presented an approach to capture patterns among a set of elements by using deformable template models. Central to our approach is a coupled template matching and deformation analysis. This analysis simultaneously identifies the best fitting template instance of each element and detects element similarities by detecting patterns in the deformation modes of the matching template instances. Even though this analysis is flexible enough to incorporate additional priors, we have assumed no additional information to demonstrate the effectiveness of the approach. Our analysis is independent of the choice of the deformation model and thus can be adopted to other

## Chapter 5. Understanding Structured Variations

---

problem settings. Defining other context-specific templates, for example for biological datasets, and performing a similar coupled analysis is an interesting area for future research.

When using templates for shape analysis, we are faced with a fundamental tradeoff. On the one end of the spectrum, static templates that can only be aligned rigidly with the data, simplify the matching process, but require a potentially prohibitive number of examples to adequately sample the space of variations observed in acquired data. On the other end of the spectrum, templates based on generic or low-level deformation models [83] have a larger number of parameters (e.g. each vertex position can be an unknown) and are prone to overfitting.

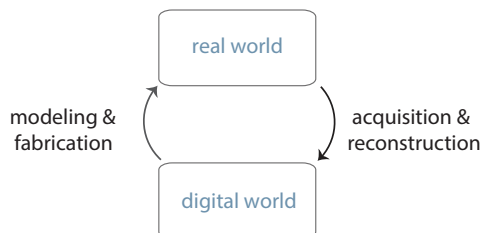
Our approach lies in the middle and employs a high-level, structure-preserving deformation model that directly encodes geometric properties of the variations observed in the data. The self-similarities detected by our coupled analysis provides valuable semantic information that can be useful for various structure aware editing tasks.

We believe that extending our coupled analysis to discover geometric similarities between the elements of different buildings can be considered as a first step in automatic detection of architectural style. The geometry of the individual elements of a building is one of the key aspects that make up its style and our analysis reveals geometric similarities between the elements of a building. It is worth exploring other geometric properties that are characteristic of certain architecture styles in the deformation patterns of the templates. In fact, the notion of architecture style can also be considered when constructing the template database. In case target data sets are expected to follow a unique architecture style, using template models revealing this style can increase the robustness of our coupled analysis. The notion of style can further be explored to define deformation priors for template models. For example, certain architecture styles impose constraints on the ratio of the width and height of its windows which can be incorporated in the template deformation process. While we have only focused on geometric similarities, enriching the templates with annotations, e.g. depicting their architecture style or material properties, and incorporating this information during template matching is also an interesting opportunity.

Finally, In our evaluations, we have utilized templates downloaded from online shape repositories which we grouped based on their geometric similarities. Considering a more sophisticated organization of the template set, e.g. a hierarchical grouping, is an interesting future direction. Furthermore, exploring the style of the templates in template organization can be helpful for specific analysis tasks.



## 6 Designing Functional Models



In a major part of this dissertation, we have concentrated on shape analysis and processing algorithms for the purpose of digitizing the physical world with a specific focus on 3D reconstruction of urban environments. Concurrent to the efforts in 3D geometry analysis and reconstruction, we are recently witnessing a revolution in digital manufacturing with the advent of accessible 3D fabrication techniques such as 3D printing, laser cutting, and milling. These recent advances in rapid prototyping technologies together with the advanced digitization techniques have given rise to a cyclic pipeline linking the physical and the digital worlds. On one hand, we strive to create accurate digital replicas of real-world objects; on the other hand, there is a growing user-base in demand of manufacturing the existing digital content. Among the various applications of 3D fabrication, we can list rapid prototyping for research purposes, patient-specific implant and medical device production, manufacturing custom-fit shoes for athletes, as well as domestic and hobbyist use.

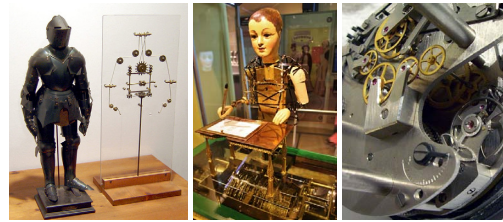
In the previous chapters, we have emphasized the importance of shape understanding in digitization of the physical world both for improving the accuracy of the reconstruction process and providing semantic information related to the acquired digital content. In this chapter, we extend our findings to underline the benefits of shape analysis for the task of recreating the physical world from the available digital data. Each manufacturing device comes with technology-specific limitations and thus imposes various constraints on the digital models that can be fabricated. Thus, a good level of shape understanding is necessary to optimize the digital content for fabrication. In the context of related shape analysis, Xu et al. [121] analyze input models based on local slippage analysis to assign appropriate joint types and support joint aware deformations. Mitra et al. [73] analyze static 3D meshes of mechanical assemblies to understand their part interactions and resulting inter-part motion possibilities. Several research efforts we have recently seen towards extending these analysis tools to design physical prototypes include optimizing geometry

## Chapter 6. Designing Functional Models

---

towards creating functionally valid furniture [107]; improving stability and robustness of printed models with thin structures [104]; creating complex assemblies of planar intersecting pieces [97]; creating assembly-free printable articulated models [13]; and decomposing large models into smaller parts that can be printed and assembled [65]. These efforts focus on generating static models. In contrast, we showcase the importance of shape understanding in the specific problem domain of creating moving mechanical figures that recreate a target motion [15]. In particular, our goal is to design and fabricate mechanical humanoid figures that mimic actions such as walking and dancing.

There are various reasons for focusing on the design of moving mechanical figures. To begin with, mechanical automata which are machines that use a combination of interconnected mechanical parts (e.g. gears, pulleys) have attracted attention of humans since antiquity due to their ability to convert a driving force into a specific target motion. A wide variety of such machines have been created



for many different purposes (e.g., clocks, music boxes, fountains). Automata designed to look like human figures performing every-day actions like walking, waving, etc. are especially popular. Famous historical examples include Leonardo Da Vinci's life-sized *armor-clad robot* from 1495 that could sit, stand and move its arms, as well as the *Draughtsman-Writer* of Henri Maillardet from the late 18th century, which was the primary inspiration for the automaton in Brian Selznick's book, *The Invention of Hugo Cabaret*. Today, wind-up toys in the form of characters are the most common types of mechanical figures. Our longstanding fascination with humanoid automata likely stems from their ability to produce surprisingly complex, lifelike motions from simple input forces through purely mechanical means.

Despite their widespread appeal, the design of automata is currently restricted to a very small group of experts. Consider the challenges involved in creating a mechanical figure that performs a specific target motion. First, a designer must choose a set of mechanical parts and decide how to connect the parts together to approximate the prescribed motion, traditionally called the *conceptual design* phase. Creating this conceptual design typically requires extensive knowledge of different part types and how part interactions transfer movement through the assembly. Next, in the *dimension synthesis* phase, the designer must determine the appropriate part parameters (e.g., gear radius) to best match the target motion. This step may require simplifying the target motion so that it falls within the range of achievable motions for the set of parts. The final step is to arrange the parts spatially to create a valid physical realization of the automaton. Only a few people have the necessary skills and expertise to perform all of these design tasks.

The attractiveness of these mechanical figures and the challenging design process creates the need for computational tools to automate the design process to make them accessible for casual users. Previously, there have been research efforts focusing on automating different phases of the mechanism design process. To automate the conceptual design phase, previous work attempts

to decompose specific functional goals into smaller subgoals and match them to a database of common building block mechanisms ([19, 93, 44] and references therein). As for the dimension synthesis stage, a common approach is to identify the relevant spatial constraints between parts and satisfy these constraints to compute part positions and orientations [4, 51]. Encoding the mating and alignment relations between different parts of an assembly using a graph of geometric constraints [85, 43] is also a common practice.

Only in the last few years, we have seen approaches that focus on the mechanism design process as a whole. In recent work, Zhu et al. [127] present a system for producing tethered mechanical assemblies that approximate user-specified input motions. In such assemblies, characters or objects (represented as rigid feature components) are driven from below by mechanisms hidden inside a box. These tethered designs drive each rigid feature component with one or two mechanical parts, allowing the component to perform a simple rotation, translation or certain combinations of two rotations/translations along orthogonal planes or axes. Concurrently to our research efforts, Coros et al. [21] present an interactive system for designing animated mechanical characters which supports complex target motions that are specified by sketching planar motion curves. In this system, certain rigid connections between different parts of the mechanism are required to be provided as input, however, to propagate the motion through the whole assembly.

Development of similar computational tools to aid users in mechanism design requires a thorough understanding of the characteristics of desired target motions. Further the requirements and limitations of mechanical components need to be carefully analyzed. Thus, in this chapter, we demonstrate how shape understanding and geometry processing can be utilized to build a *fully* automated approach for generating mechanical figures that approximate a given target motion.

## 6.1 Overview

Our goal is to present an automated approach for generating mechanical figures that realize target motions. Before providing the details of our proposed approach, we first describe our analysis results on the characteristics of the target motions. We have made specific design decisions considering these characteristic features of the target motions.

As input we consider an animation sequence for an articulated 3D humanoid figure (e.g., from a motion-capture database) that specifies the desired motion. Since mechanical automata typically perform repetitive actions, we assume that the input motions are roughly periodic (e.g., a walk cycle, an arm waving back and forth). Each bone of a human figure performs an oscillating motion, i.e. a rotation that repetitively changes direction, during such a periodic motion cycle (see Figure 6.1). Further, the bones of a limb of the figure are connected via joints forming a kinematic chain. In other words, the motion of a specific bone is affected by the motion of its parent bone. An important observation we make is that the oscillation of each bone in a kinematic chain possibly has different phases and frequencies. For example, in running motions, the oscillations of the upper and lower arms are typically out-of-phase, i.e. they are expected to



Figure 6.1: When performing a periodic motion such as walking, each bone of a humanoid figure undergoes an oscillation, possibly with different phase and frequency.

change rotation direction at different time frames and rotate with different angular speeds.

Our designs represent the bones of the input figure with rigid links connected by revolute joints. Mechanical figures are often driven from a single input crank such as a constant speed motor to facilitate practical use. Thus, the driving force of the mechanism is a constant angular-speed rotation. The automaton is required to convert the crank rotation to oscillatory movements that are propagated to the appropriate links in the figure. More importantly, this propagation should generate oscillations with different phases and frequencies for different body parts, which greatly expands the expressive range of the designs.

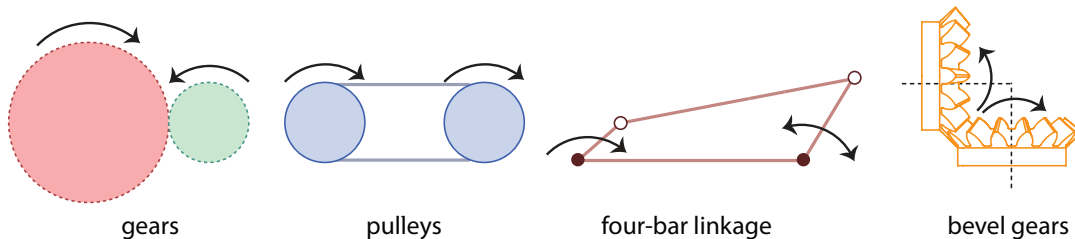


Figure 6.2: The basic components of our mechanical designs include gears, pulleys, and four-bar linkages.

Standard mechanical components most of us are familiar with include gears and pulleys which are capable of propagating uniform rotations only. In order to realize the conversion from an input rotation to an oscillation, we use another common mechanical component called the four-bar linkage (see Figure 6.2). As the input crank of the linkage is rotating uniformly in one direction, the output crank oscillates with a non-linear angular speed given certain constraints relating the bar lengths are satisfied [68]. The four-bar linkages generate oscillating motions with continuous acceleration rather than abrupt changes in direction that can put significant stress on the components.

We introduce a mechanical module composed of these standard components, i.e., gears, pulleys, and four-bar linkages, and is capable of converting a uni-directional rotation to oscillation. We



further ensure such modules can create out-of-phase oscillations when linked along a kinematic chain. Since all of these components are planar mechanisms, they can only generate planar motions. This planarity constraint, however, enables us to fabricate these components with a laser cutter. Our goal is to generate freestanding automata where the mechanisms that produce motion are attached directly to the links of the figure. While this property enables the generated automata to move more freely, it also imposes additional constraints on the layout of the mechanical components. Each link is required to be long enough to accommodate the components attached to it. In addition, since each link represents a bone of the input figure, the relative lengths of the rigid links should preserve the proportion of the corresponding bone lengths.

We propose a two-stage strategy to automatically design mechanical automata that satisfies all the requirements listed above. First, in a *motion approximation* stage, we convert the input animation sequence into a mechanically realizable motion based on the constraints imposed by the parts (gears, pulleys and four-bar linkages) in our designs. Then, in the *layout* stage, we solve for the parameters (e.g., tooth counts, link lengths) and spatial layout of the mechanical parts to produce the desired motion of each link.

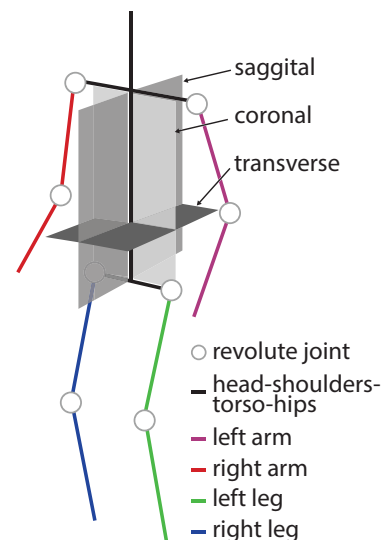
## 6.2 Algorithm Details

### 6.2.1 Mechanical Design

The input to our system is an animation sequence of an articulated 3D human figure. From this animation, our system automatically produces a humanoid automaton design that approximates the input motion. Our design represents the bones of the input figure with rigid links.

Since many human motions are characterized primarily by the movement of the limbs, we connect the arm and leg links with revolute joints to create points of articulation at the shoulders, elbows, hips and knees of the automaton. We orient these joints such that the motion of all the links within a limb lies in a single plane that is parallel to the sagittal, coronal or transverse plane of the figure (see the inset figure).

Constraining limb motions to these orthogonal planes limits the range of movements that our mechanisms can achieve. On the other hand, this design makes it possible to produce motion using standard planar mechanisms, i.e. gears, pulleys, four-bar linkages, and transfer the rotation of a single input motor to the motion plane of each limb using standard bevel gears, which convert rotation across orthogonal planes. Furthermore, there are many human movements in which the motion of the



## Chapter 6. Designing Functional Models

limbs is mostly parallel to one of these planes (e.g., walking, jumping jacks). Since we focus primarily on the motion of the limbs, we combine the head, shoulder, torso and hips to form a single rigid component.

**Oscillation Module.** To realize the oscillating motion of each link in the mechanical automaton, we introduce an oscillation module. This module uses gears, a pulley, and a four-bar linkage to convert unidirectional rotation into an oscillating motion (see Figure 6.6).

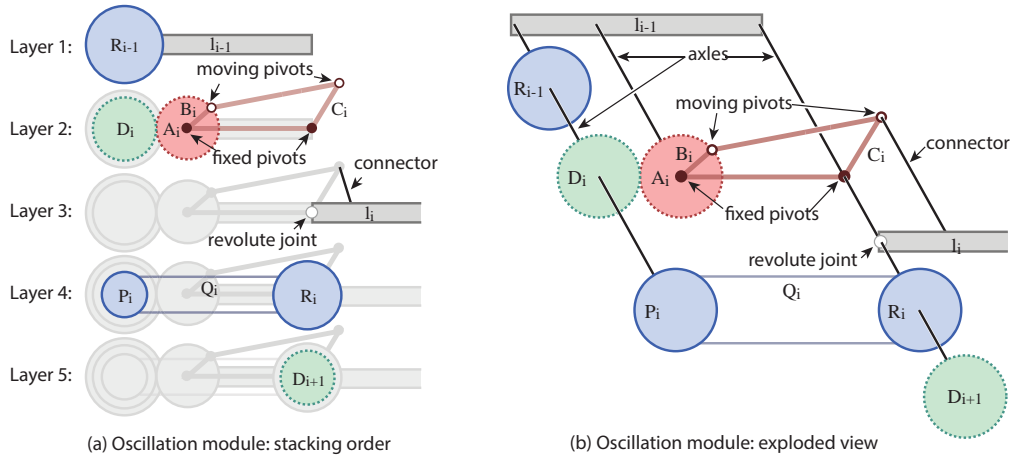
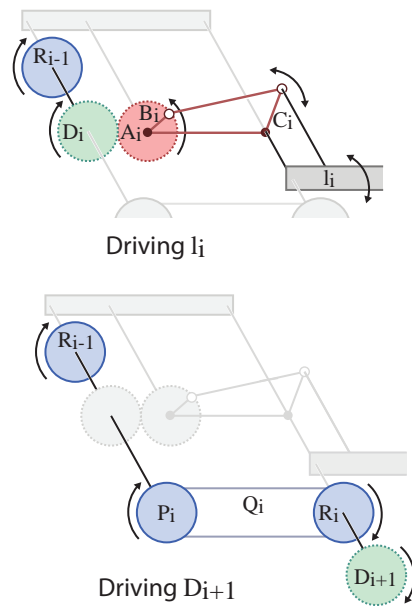


Figure 6.3: We drive the rigid links of the automaton with oscillation modules that consist of gears, pulleys, and four-bar linkages. These components are stacked onto axles that are connected to each link. We show the stacking order of the parts starting from the bottom layer (a), as well as an exploded view (b).

The input to a module  $M_i$  is the gear  $D_i$ . Applying a unidirectional rotation to  $D_i$  causes the meshing gear  $A_i$  to rotate. The rotation of  $A_i$  drives the input crank  $B_i$  of a four-bar linkage. The linkage converts this input rotation into an oscillating rotation of its output crank  $C_i$ . The axles of  $D_i$ ,  $A_i$  and the fixed pivot of  $C_i$  are all attached to the previous link  $l_{i-1}$  while the moving pivot of  $C_i$  is connected rigidly to link  $l_i$ . Thus, the oscillation of  $C_i$  causes  $l_i$  to oscillate as well. Rotating  $D_i$  also drives the pulley wheel  $P_i$ , which is attached rigidly to the same axle as  $D_i$ . The pulley belt  $Q_i$  transfers the rotation of  $P_i$  to the other pulley wheel  $R_i$ . To prevent slipping between  $P_i$ ,  $Q_i$  and  $R_i$ , we use pulley wheels and belts that have interlocking teeth. Connecting  $R_i$  rigidly to the same axle as the input gear  $D_{i+1}$  of the next module  $M_{i+1}$  propagates the motion to the next link in the chain. Thus, we can move an entire limb of



the automaton by attaching a chain of oscillation modules to the rigid links and driving the root module  $M_1$  (which must be attached to a stationary link of the figure) with a uni-directional rotation. We should emphasize that the module that drives the last link in the chain does not have pulley parts because no further rotation propagation is required. To simplify the motion and layout parameter computation, we restrict  $D_i$  and  $A_i$  to be of the same size.

**Motion Parameters.** There are several parameters that define the mechanical motion of a chain of oscillation modules. Before outlining an automated procedure to design such modules, we first provide an overview of these parameters. The motion of a single module is defined by the rotation speed  $\dot{\alpha}_i$  of its input gear  $D_i$ , the lengths of its four-bar linkage  $b_i$ ,  $c_i$ , and  $g_i$ , and the initial angles  $\theta_i(0)$  and  $\phi_i(0)$  of the input and output cranks in the linkage (see Figure 6.4). The length  $h_i$  of the bar connecting the ends of the two cranks  $B_i$  and  $C_i$  is fully determined by the other linkage parameters, i.e. the lengths of the remaining bars and the initial orientation of the input and output bars. Following the book of McCarthy [68], we can express the angle  $\phi_i(t)$  of the output crank  $C_i$  at time  $t$  as:

$$\begin{aligned} \phi_i(t) &= \arctan\left(\frac{S_i(t)}{T_i(t)}\right) + \arccos\left(\frac{U_i(t) - V_i(t)}{W_i(t)}\right) \\ S_i(t) &= 2b_i c_i \cos(\theta_i(t)) - 2g_i c_i \\ T_i(t) &= 2b_i c_i \sin(\theta_i(t)) \\ U_i(t) &= 2b_i g_i (\cos(\theta_i(0)) - \cos(\theta_i(t))) \\ V_i(t) &= 2c_i (g_i \cos(\phi_i(0)) - b_i \cos(\theta_i(0) - \phi_i(0))) \\ W_i(t) &= 2c_i \sqrt{g_i^2 + b_i^2 - 2b_i g_i \cos(\theta_i(0))}, \end{aligned} \quad (6.1)$$

where  $\theta_i(t)$  denotes the rotation of the input crank  $B_i$  at time  $t$ . If  $D_i$  is rotated by an angular speed of  $\dot{\alpha}_i$ , both  $A_i$  and  $B_i$  will have an angular speed of  $-\dot{\alpha}_i$ . Thus, we can conclude that  $\theta_i(t) = \theta_i(0) - t\dot{\alpha}_i$ .

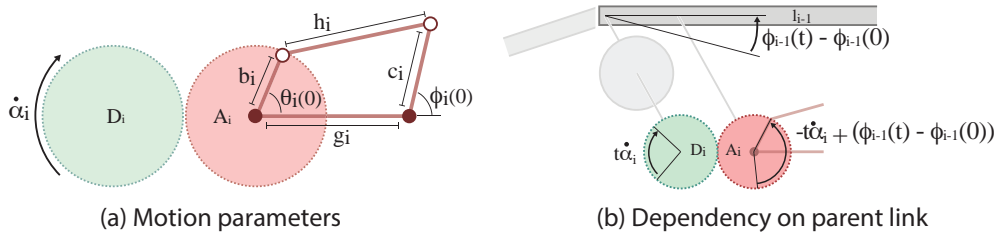


Figure 6.4: The output of an oscillation module is defined by a set of motion parameters including the lengths and the initial orientation of the linkage bars (a). The motion of a one module in a chain of modules depends on the motion of its parent link (b).

When we connect a chain of oscillation modules together, we must account for the fact that the input crank rotation  $\theta_i(t)$  for each module depends on the motion of the parent link  $l_{i-1}$ . Since

## Chapter 6. Designing Functional Models

---

the axle of  $A_i$  is connected to  $l_{i-1}$ ,  $A_i$  rolls around  $D_i$  as  $l_{i-1}$  rotates, which affects the total rotation of  $A_i$  (see Figure 6.4). Specifically, if we apply the rotation speed  $\dot{\alpha}_i$  to  $D_i$ , then the angular speed of  $A_i$  is a combination of this input rotation and the rotation of  $l_{i-1}$ . As a result, the definition for  $\theta_i(t)$  in a chain of modules becomes

$$\theta_i(t) = \theta_i(0) - t\dot{\alpha}_i + (\phi_{i-1}(t) - \phi_{i-1}(0)), \quad (6.2)$$

where  $(\phi_{i-1}(t) - \phi_{i-1}(0))$  denotes the total rotation of  $C_{i-1}$  (and thus the attached link  $l_{i-1}$ ) at time  $t$  with respect to its initial angle. Since the root module  $M_1$  that drives the first link  $l_1$  in the chain is attached to the main support structure of the automaton rather than a moving link, we define  $\phi_0(t)$  to be 0 for all  $t$ .

We have to incorporate several constraints on the motion parameter values to ensure the oscillation modules are physically valid. To ensure that the input crank  $B_i$  of the linkage can rotate fully when driven by  $D_i$ , we constrain the relative lengths of the linkage bars as follows [68]:

$$\begin{aligned} g_i + b_i - h_i - c_i &< 0 \\ (h_i - c_i)^2 - (g_i - b_i)^2 &< 0 \end{aligned}$$

Further, since the argument of the arccos function in Equation 6.1 must be in the range  $[-1, 1]$ , we enforce the following constraint:

$$\begin{aligned} 2g_i(b_i \cos(\theta_i(0)) - c_i \cos(\phi_i(0))) + \\ 2b_i c_i \cos(\theta_i(0) - \phi_i(0)) - b_i^2 - c_i^2 - g_i^2 \leq 0 \end{aligned}$$

The equations above define the space of all possible motions for a chain of rigid links using our design based on the oscillation module. We will later use these relationships to compute motion parameters that best approximate an input animation sequence.

**Layout Parameters.** In addition to the motion parameters that partially specify the configuration of the oscillation module components, we need several additional parameters that fully define the layout of the module.

The parameters  $b_i, c_i, g_i$  determine the lengths of the linkage bars, but since scaling the entire

linkage uniformly does not change its motion, we treat the overall scale  $\omega_i$  as a free layout parameter. The other layout parameters include the radii of the gears and pulley wheels ( $r_{D_i}, r_{A_i}, r_{P_i}, r_{R_i}$ ) as well as the pulley belt length  $s_{Q_i}$  and link length  $s_{l_i}$ . As mentioned earlier, we restrict  $D_i$  and  $A_i$  to be the same size, so we set  $r_{A_i} = r_{D_i}$  and eliminate  $r_{A_i}$  as a free parameter. Given a fixed tooth size, the tooth count of the gears, pulley wheels, and pulley belt determines the radius and the length of these components respectively. Thus, we express  $r_{D_i}, r_{P_i}, r_{R_i}, s_{Q_i}$  in terms of integer tooth counts  $k_{D_i}, k_{P_i}, k_{R_i}, k_{Q_i}$ .

Not all assignments of layout parameters result in physically valid layouts. There are three relevant constraints to ensure physical validity:

1. To prevent  $A_i$  from colliding with the axle of the next link  $l_{i+1}$ , we must ensure that  $r_{A_i} < \omega_i g_i$ .
2. Similarly, to avoid collisions between the pulley wheel  $R_i$  of the module  $M_i$  and the axle of the gear  $A_{i+1}$  of the next module  $M_{i+1}$ , the relation  $r_{R_i} < 2r_{A_{i+1}}$  must hold.
3. The length of each link must match the size of the components placed on the link, so that  $s_{l_i} = 2r_{A_{i+1}} + \omega_{i+1} g_{i+1}$ . The link length also influences the pulley belt length  $s_{Q_i}$ . However, since belts with a small amount of slack do not prevent the pulley from functioning, we represent this relationship as an energy term in our layout optimization rather than a hard constraint.

We also incorporate a few constraints to reflect the limitations of the fabrication process.

1. We manufacture the gears  $D_i$  and  $A_i$  with a laser cutter, which limits the minimum and maximum sizes (i.e., tooth counts) of these components. In our designs, we constrain the tooth count to be between 14 and 80.
2. Similarly, we constrain the lengths of the linkage bars based on the minimum bar length (8mm) that we can cut.
3. Finally, since it is difficult to create robust pulley wheels and belts with a laser cutter, we use stock pre-made pulley parts that are only available in a discrete set of wheel and belt sizes. To encode this constraint in our layout parameterization, we introduce a set of binary indicator variables  $\{u_1, u_2, \dots, u_N\}$  to select between the  $N$  available part sizes  $\{v_1, v_2, \dots, v_N\}$ , and we represent the size of each pulley part as  $\sum_{i=1}^N u_i v_i$  with the constraint  $\sum_{i=1}^N u_i = 1$ .

This set of layout parameters and constraints fully specifies the physical layout of a chain of oscillation modules in our design. We next describe how we optimize for the motion parameters of a chain of oscillation modules to best approximate a given motion and compute the layout that matches the optimized motion parameters.

### 6.2.2 Motion and Layout Optimization

In the previous section, we have introduced a set of *motion parameters* that define the motion of a chain of oscillation modules and a set of *layout parameters* that fully define the layout of these modules. Given an input motion sequence, our goal is to optimize for these parameters to generate a physically valid automaton that approximates the motion as close as possible. We begin our discussion with the motion approximation step. Given a set of optimized motion parameters, we then optimize for the final layout of the oscillation modules.

**Motion Approximation.** Given a mocap sequence as input, we first simplify the animation by projecting the motion of the bones within each limb onto a plane. We then run an optimization to find the motion parameters that yield the best approximation of the planar motion that can be generated with a chain of oscillation modules.

*Planar approximation:* For each kinematic chain, we start by tracing the path  $X_i = \{x_i(1), x_i(2), \dots, x_i(n)\}$  of the endpoint of the  $i$ th bone at each frame of the input animation sequence with respect to its parent joint where  $n$  is the total number of frames. We then compute the least-squares plane of the traced path and project each point onto this plane to obtain a planar path  $Y_i = \{y_i(1), y_i(2), \dots, y_i(n)\}$ . We then snap the fitted plane to one of the saggital, coronal, or transverse planes and update the planar path accordingly. Finally, for every frame  $j$ , we convert the motion of each bone to a rotation  $\Phi_i(j)$  around the normal of the fitted plane as:

$$\cos(\Phi_i(j)) = \cos(\Phi_i(j-1)) + \frac{y_i(j)}{\|y_i(j)\|} \cdot \frac{y_i(j-1)}{\|y_i(j-1)\|},$$

where  $\cos(\Phi_i(1))$  denotes the initial orientation of the bone with respect to its parent and  $\cos(\Phi_i(0)) = 0$ .

Converting the rotation of a single input crank to motion in different planes for each limb requires the use of bevel gears. Even though bevel gears can be designed to work for other angles, they are often mounted on shafts 90 degrees apart allowing to change the rotation axis across orthogonal planes. Thus, we snap the plane of motion for each limb to one of the saggital, coronal, or transverse plane of the figure.

*Motion parameter optimization:* Given the planar bone rotation angles  $\Phi_i(j)$  for each chain, our goal is to compute motion parameters that approximate the mocap bone motion with the corresponding rigid links in the automaton. Since each link  $l_i$  is driven by the output crank  $C_i$ , we aim to produce a mechanical motion where changes in the crank angle  $\phi_i(t)$  match changes in the bone angles  $\Phi_i(j)$ . Thus we define the following energy term for each chain:

$$E_{\Phi_i} = \sum_i \sum_j \left( \sin \frac{\Delta\Phi_i(j) - \Delta\phi_i(j)}{2} \right)^2, \quad (6.3)$$

where  $i$  indexes the bones,  $j$  indexes the frames of the animation sequence,  $\Delta\Phi_i(j) = \Phi_i(j+1) - \Phi_i(j)$  denotes the change in the bone angle between frames  $j$  and  $j+1$  and  $\Delta\phi_i(j) = \phi_i(j+1) - \phi_i(j)$  denotes the corresponding change in angle of the output crank  $C_i$ .

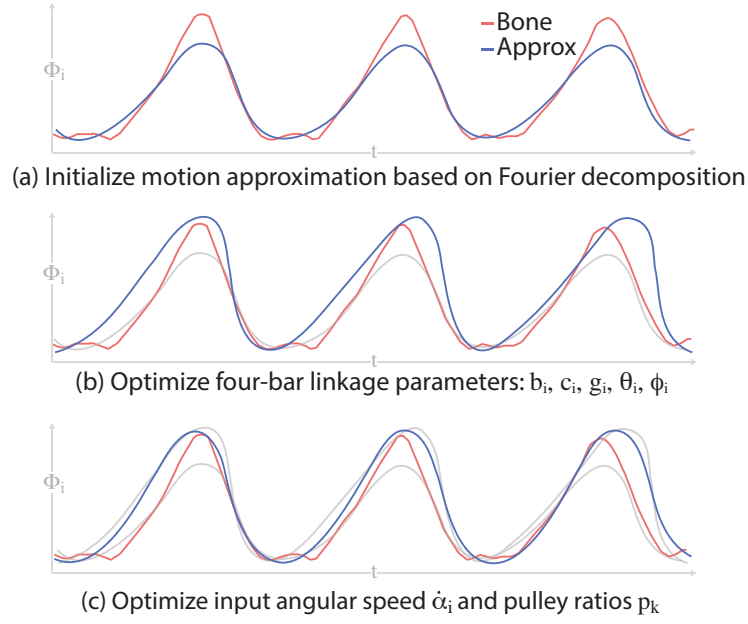


Figure 6.5: We apply a two-step optimization procedure to optimize for the motion parameters of the oscillation modules used in our designs.

Recall that,  $\phi_i(t)$  depends on the four-bar linkage parameters  $b_i, c_i, g_i, \theta_i(0), \phi_i(0)$ , and the input angular speed of each module  $\dot{\alpha}_i$ . Our goal is to minimize  $E_{\Phi_i}$  by optimizing these parameters. Instead of directly optimizing for  $\dot{\alpha}_i$ , we represent  $\dot{\alpha}_i$  in terms of the pulley ratios for all the preceding modules in the chain. In particular, for all but the root module  $M_1$ , the input angular speed of  $M_i$  is defined as  $\dot{\alpha}_i = \prod_{k=0}^{i-1} p_k$  where  $p_k = k_{P_k} / k_{R_k}$  is the pulley ratio of module  $M_k$ . Based on this relationship, we optimize for the linkage parameters, the angular speed  $\dot{\alpha}_1$  of the root module, and the pulley ratios  $p_i$  of each remaining module in the chain.

We introduce a two-step optimization procedure to minimize  $E_{\Phi_i}$ . In the first step, given  $\dot{\alpha}_1$  and  $p_i$ , we solve for the linkage parameters  $b_i, c_i, g_i, \theta_i(0), \phi_i(0)$  of each module. In the second step, we use the computed linkage parameters to update  $\dot{\alpha}_1$  and  $p_i$  across all the bones in the chain. In both of these steps, we minimize the non-linear error function given in Equation 6.3 with respect to the constraints involving the linkage parameters described in the previous section. We use the *SQP* method implemented in the *MATLAB Optimization Toolbox* to solve these optimization

problems, which typically converge in 5-10 iterations.

The non-linearity of the objective function defined in Equation 6.3 mandates a good initialization of the optimization variables to obtain a valid minimum. Since we are considering periodic motions, we employ a frequency-space analysis for this purpose. More precisely, we initialize the motion parameters using a Fourier decomposition on the rotation angles of each bone in the chain. The motion of each bone is then approximated using the Fourier component of highest magnitude as

$$\Phi(j) = \mu * \cos(2\pi f(j/n) + \rho),$$

where  $\mu$  represents the magnitude,  $f$  represents the frequency, and  $\rho$  represents the phase of the Fourier component. This cosine function represents the change in rotation of the output crank of the four-bar linkage as the input crank completes one full rotation cycle in  $n/f$  frames. Assuming a fixed initial orientation  $\theta_i(0)$  of the input of the linkage, each sampled value of  $j$  corresponds to a desired pair of input and output crank rotation angles  $(\theta_i(0) + 2\pi j(f/n), \Phi(j))$ . We sample 3 such angle pairs which uniquely define a set of linkage bar lengths that interpolate the sample points [28]. We initialize  $\alpha_1$  as  $2\pi(f_1/n)$  where  $f_1$  is the frequency of the Fourier component approximating the rotation of the first bone in the chain. We assume the initial values for the pulley size ratios are  $p_i = (2\pi(f_i/n))/(2\pi(f_{i-1}/n))$ .

Having too short or too long linkage bars make the physical assembly process difficult. To prevent this case from happening upon scaling of the four-bar linkage, we add an additional energy term to penalize large ratios between the bar lengths. Specifically, we define the following relations between the bar lengths:

$$\kappa_i = \frac{b_i g_i \cos(\theta_i(0)) - g_i c_i \cos(\phi_i(0)) + b_i c_i \cos(\theta_i(0) - \phi_i(0))}{(b_i c_i)}$$

$$\lambda_i = g_i / b_i$$

$$\mu_i = g_i / c_i$$

We add the energy term below to the first step of the optimization:

$$E_{bar}^i = \left(\kappa_i - \frac{1}{\kappa_i}\right)^2 + (\lambda_i - 1)^2 + \left(\frac{1}{\lambda_i} - 1\right)^2 + (\mu_i - 1)^2 + \left(\frac{1}{\mu_i} - 1\right)^2$$

A mechanical automaton performing a periodic motion should return to its starting configura-



tion at the end of each motion cycle. However, any residual error introduced during motion approximation might lead to drifts in the motion of the links over time. To reduce such drifts, we explicitly define a periodicity constraint in the second step of the optimization. Specifically, assuming the provided motion sequence contains one cycle of the desired motion, the period of the overall motion is  $n$  frames. Each oscillation module should return back to its initial state at the end of the motion cycle. Thus, the periodicity of each bone in the chain is an integer divisor of  $n$ , and the input crank of the corresponding four bar linkage should be rotated by an integer multiple of  $2\pi$ , i.e.  $2\pi m_i$  during the given motion cycle. In addition to the propagated input rotation parameters, we also solve for these discrete multipliers  $m_i$ . We treat  $m_i$  as continuous variables and adopt a branch and bound strategy to round them to integer values. Specifically, starting at the root module, at each iteration we round  $m_i$  to  $\lfloor m_i \rfloor$  and  $\lceil m_i \rceil$  and choose the value resulting in smaller optimization error. In practice, we have observed that duplicating the input motion several times (10 in our experiments) and running the motion approximation to this longer cycle improves the approximation results while still satisfying the periodicity constraints.

*Layout Optimization.* At the end of the motion parameter optimization step, we compute the motion related parameters of each oscillation module driving the limbs that generate an approximation of the input motion. Given this simplified motion, the next step is to compute the remaining layout parameters for the oscillation modules to generate the final layout of the mechanical automaton. Our goal is to optimize for the layout parameters that match both the target motion parameters computed in the previous step and relative bone lengths of the input figure as closely as possible.

The layout of an oscillation module is defined by continuous parameters  $(\omega_i, s_i)$ , discrete parameters  $(k_{D_i}, k_{P_i}, k_{R_i}, k_{Q_i})$ , and the binary indicator variables for the pulley part sizes. Thus, we use mixed-integer programming to search over the space of possible solutions. We formulate the layout problem by defining an energy function that encodes several desired properties for the layout of each kinematic chain. We now introduce these properties in detail:

- **Propagated rotation:** At the end of the motion parameter optimization step, we have computed a desired angular speed  $\dot{\alpha}_1$  driving the root module  $M_1$  of each chain and the ratios of the pulley wheel radii,  $p_i$ , that specify how the input rotation should be propagated to the remaining modules in the chain. The actual values of  $\dot{\alpha}_1$  and  $p_i$  in the final automaton depend on several layout parameters. In our designs, the input crank of the automaton drives an input gear whose rotation is propagated to the driving gear  $D_1$  of each root module  $M_1$ . Thus, the angular speed driving  $M_1$  can be expressed as  $\dot{\alpha}_1 = \alpha_I k_I / k_{D_1}$  where  $\alpha_I$  is the constant angular speed of the automaton input crank,  $k_I$  is the tooth count of the input gear, and  $k_{D_1}$  is the tooth count of  $D_1$ . The pulley ratios  $p_i$  can simply be expressed as  $k_{P_i} / k_{R_i}$  (i.e., the ratios of the corresponding pulley wheel tooth counts). Based on these expressions, we define the following energy term to penalize deviations from the desired

angular speed and pulley ratios:

$$E_{rot_i} = \begin{cases} (\dot{\alpha}_I k_I / k_{D_1} - \dot{\alpha}_1)^2 & \text{if } i = 1 \\ (k_{P_i} / k_{R_i} - p_i)^2 & \text{if } i > 1 \end{cases} \quad (6.4)$$

- Pulley belt length: In our oscillation module design, the wheels of each pulley,  $P_i$  and  $R_i$ , are attached to the axles at either end of link  $l_{i-1}$  (see Figure 6.6). Thus, we can express the tooth count (i.e., length) of the pulley belt  $k_{Q_i}$  in terms of the tooth counts of the pulley wheels,  $k_{P_i}$  and  $k_{R_i}$ , and the length  $s_{l_{i-1}}$  of link  $l_{i-1}$ :

$$k_{Q_i} z_p = z_p (k_{P_i} + k_{R_i}) / 2 + 2s_{l_{i-1}},$$

where  $z_p$  is a constant defining the distance between two consecutive teeth of the pulley wheels or belt.

The pulley wheels  $P_1$  and  $R_1$  that drive the second link  $l_2$  of each chain are part of the root module  $M_1$ . Since  $M_1$  is attached to the main support structure of the automaton rather than a link in the chain, the tooth count  $k_{Q_1}$  of the pulley belt  $Q_1$  is not constrained by the length of a link. Instead,  $k_{Q_1}$  depends on the tooth counts of the pulley wheels,  $P_1$  and  $R_1$ , the input gear  $D_1$ , and the distance  $\omega_1 g_1$  between the fixed pivots of  $B_1$  and  $C_1$ :

$$k_{Q_1} z_p = z_p (k_{P_1} + k_{R_1}) / 2 + 2(z_p k_{D_1} + \omega_1 g_1).$$

We recall that we provide the pulley wheels and belts from retail stores where only a limited set of options are available. In practice, a small amount of slack in the pulley belt does not prevent the mechanism from functioning properly. Therefore, instead of treating the above equalities as hard constraints, we define an energy term to penalize the difference between the belt length  $k_{Q_i}$  chosen from the discrete set of available options and the ideal belt length defined by the layout of the mechanism:

$$E_{pul_i} = \begin{cases} (k_{Q_i} z_p - z_p \frac{k_{P_i} + k_{R_i}}{2} + 2(m_p k_{D_{i-1}} + \omega_{i-1} g_{i-1}) - \epsilon)^2 & \text{if } i = 1 \\ (k_{Q_i} z_p - z_p (k_{P_i} + k_{R_i}) / 2 + 2s_{l_{i-2}} - \epsilon)^2 & \text{if } i > 1 \end{cases} \quad (6.5)$$

where we set  $\epsilon = 1$  to allow for the slack. The pulley part sizes referenced in this formulation are represented by the corresponding binary indicator variables as described before.

- Link lengths: The oscillation module driving a link in a kinematic chain is attached to the previous link in the chain. Thus, each link should be long enough to accommodate the parts attached to itself. In order to preserve the shape of the input figure, the length  $s_{l_i}$  of

each link should also match as closely as possible the corresponding relative bone length  $s_{b_i}$ . The dimensions of the bones are defined up to a scale using a free scale variable  $\xi$ . Thus, the desired length of each rigid link is  $s_{l_i} = \xi s_{b_i}$ . To penalize the deviations from the original bone lengths, we add the following term to our energy function for each link:

$$E_{s_{l_i}} = (s_{l_i} - \xi s_{b_i})^2 \quad (6.6)$$

We further desire symmetric bone pairs (e.g., left and right upper leg bones) to have similar lengths. For each such symmetric bone pair, we penalize the differences between the corresponding link lengths as,

$$E_{s_{ij}} = (s_{l_i} - s_{l_j})^2. \quad (6.7)$$

We combine all the energy terms for each link  $l_i$  in each kinematic chain  $c_j$  to obtain the final energy function:

$$E = \sum_{c_j} \sum_{l_i \in c_j} w_{rot_i} E_{rot_i} + w_{pul_i} E_{pul_i} + w_{s_{l_i}} E_{s_{l_i}} + \sum_{i,j \in S} w_{s_{ij}} E_{s_{ij}}, \quad (6.8)$$

where  $S$  denotes symmetric bone tuples and  $w_{rot_i}$ ,  $w_{pul_i}$ ,  $w_{s_{l_i}}$ , and  $w_{s_{ij}}$  are the weights for each energy term. In our experiments, we set  $w_{rot_i}$  and  $w_{pul_i}$  to 5 and set  $w_{s_{l_i}}$  and  $w_{s_{ij}}$  to 1.

Given the energy function  $E$  and all the linear hard constraints described previously, we formulate the optimization as a mixed integer programming problem and solve it using a branch-and-bound technique. More specifically, we use the constrained mixed-integer solver Gurobi [42]. During the optimization process, the solver treats the discrete unknowns as continuous variables and generates a solution where these variables have fractional values. The solver then constructs a search tree where each branch represents the result of constraining a discrete variable to either the floor or ceiling of its fractional value from the continuous solution.

In our experiments, this optimization process takes around 100 seconds to find a feasible solution for a problem with 360 binary, 10 integer, and 10 continuous variables. The output of this optimization is a set of layout parameters that fully define the size of the mechanical parts in each oscillation module. Since each module has a predefined structure, the optimized parameters also define the final layout of the modules.

### 6.2.3 Unified Layout Design

Once the motion and layout parameters of the oscillation modules driving each limb of the input figure are optimized, the final step is to compose all the kinematic chains into a unified design that can be driven by a single input crank. This process involves choosing the orientation and position of the crank and then generating mechanisms that propagate the crank rotation to the root module of each moving chain in the automaton. Since we assume each limb is moving in one of the principal orthogonal planes, we define a set of rules considering the direction of motion to generate the unified design automatically.

In our designs, we choose to attach the input crank to the fixed torso link of the figure. We use pulleys and bevel gears (if necessary) to propagate the rotation of the input crank to the rest of the automaton. We arrange these pulleys and bevel gears on the stationary torso, head and (in some cases) shoulder links based on the orientations of the motion planes for the moving limbs.

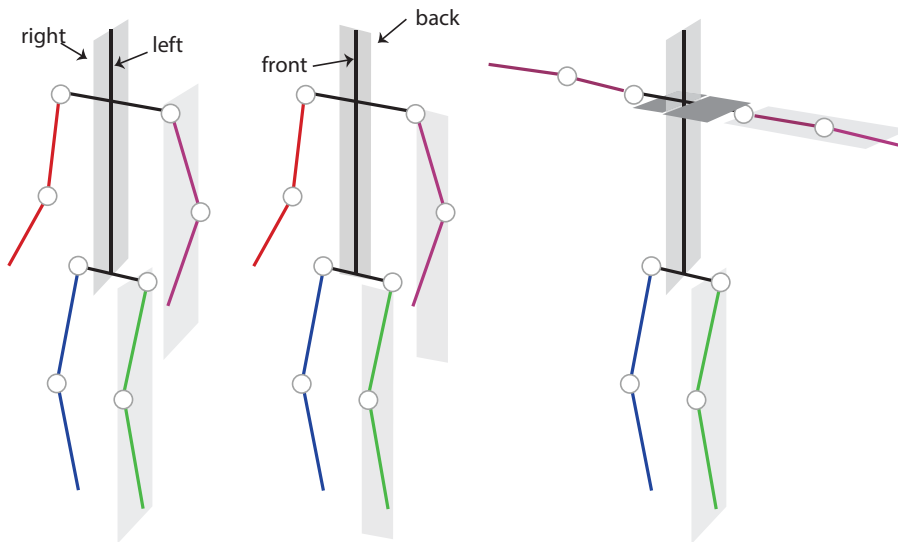


Figure 6.6: The main support structure of the mechanical automaton, the torso, is oriented based on the motion planes of the moving limbs. Both sides of this support is used to accommodate the pulleys propagating the input crank rotation to the limbs. In special cases, the shoulders are used as support structures as well.

If the motion planes of the moving limbs are all parallel to either the saggital or coronal planes of the figure, we orient the input crank, torso and head links parallel to the motion plane (see Figure 6.6). We recall that the root module of each chain must be placed on one of the stationary links. We position the root modules of the arms on the head link and the root modules of the legs on the torso link. Since pulleys are better suited to propagate motion along a longer distance, we use pulleys to connect the input crank to the driving gear of each root module. If the limbs move parallel to the saggital plane, we position the pulleys on the left and right sides of the torso/head links, and for coronal plane motion, we put the pulleys on the front and back sides of the torso/head links.

If the motion planes of the moving limbs are all parallel to the transverse plane, or if the motion planes are not all parallel to each other, we incorporate bevel gears into our unified design (see Figure 6.6). If all of the non-transverse motion planes are parallel to the coronal plane, then we orient the input crank, torso and head links parallel to the coronal plane as well. Otherwise, we orient these components parallel to the saggital plane. We orient the root module of each chain so that its rotation plane is parallel to the rotation plane of the input crank. We place the root modules for the arms and legs on the head and torso links, respectively. For chains whose motion plane is orthogonal to the rotation plane of the input crank, we use bevel gears to convert the rotation of the output crank  $C_1$  and pulley wheel  $R_1$  of the root module to the appropriate motion plane.

We observe a special case when the motion plane of an arm is parallel to the coronal or transverse plane. We then use the shoulder link as an additional support structure to accommodate the root module of the arm. We orient the shoulder to be parallel to the motion plane of the arm, and if the input crank rotates in a different plane, we add a bevel gear configuration to convert the input crank rotation to the appropriate motion plane.

At the end of unified layout design stage, our automata are ready to be operated from a single input motor or crank.

## 6.3 Evaluation

Automatic mechanism design is a very challenging task. In order to simplify this task and generate mechanisms using standard mechanical components, we have made several assumptions. As one of the main assumptions, we have assigned a planar motion to each of the moving limbs of the input figure. In order to evaluate how well this planar motion simplification algorithm approximates various human motions, we analyze a large number of mocap sequences from existing motion databases.

We first trace the path  $X_i = \{x_i(1), x_i(2), \dots, x_i(n)\}$  of the endpoint of the  $i$ -th bone of a kinematic chain  $c_j$  at each frame of the input motion. We compute the common least squares plane  $P_j(\mathbf{n}, d)$  of all such paths for  $c_j$ . We then measure the fitting error as the deviation from a planar motion over  $n$  frames as:

$$E_j := \sum_{b_i \in c_j} \sum_n (\mathbf{n} \cdot x_i(n) + d). \quad (6.9)$$

In order to account for differences in the lengths of the paths traced by each bone, we normalize the error by the total length of the paths, measured as  $m_j := \sum_{b_i \in c_j} \sum_n \|x_i(n+1) - x_i(n)\|$ . The

final planar fitting score of the motion is then computed over all the chains as:

$$M_p = \min_j m_j / (E_j + \epsilon), \quad (6.10)$$

with  $\epsilon = 0.01$  used as a regularizer. Essentially,  $M_p$  measures how well a non-trivial motion can be approximated using a set of planes, one for each chain.

We evaluate this planarity measure for a total of 80 mocap sequences from the CMU motion database [14]. We observe the following motion types (about 40% of the database) to be well suited for our algorithm: walking, running, jogging, exercising, waving, drinking as they consistently get high scores ( $M_p > 10$ ). We call such motions *good motion* types. On the other hand, sequences like swimming, swordplay, fishing ( $M_p < 2$ ) contain large out-of-plane motions, and hence our algorithm cannot replicate them faithfully.

Our system produces a mechanical automaton design for a given input motion. We simulate the mechanism using forward kinematics to qualitatively validate how faithfully the sequences recreate the input motion. Specifically, we build a mechanism graph  $G = (V, E)$ , where each node  $v_i \in V$  represents a part (e.g., a gear or a pulley) and each edge  $e_{ij} \in E$  connecting the vertices  $v_i$  and  $v_j$  represents the relation between the corresponding parts. In our setup, typical relations include the coaxial or parallel-axis properties of the gears, and the motion chains comprising of pulleys and four-bar linkages. At every frame of the motion sequence, we assume that the input crank rotates clockwise with a constant speed. We propagate this rotation along the edges of the mechanism graph to compute the position and orientation of each component in the mechanism. These graphs are free of any loops by construction.

In Figure 6.7, we present representative realizations for some of the good motion types. For each motion type, several snapshots of the original animation and the corresponding simulation frames are provided. In all of our example automata, the torso of the figure has been used as the main support structure. A video demonstrating the generated automata in operation can be accessed from the project page.<sup>1</sup>

**Fabrication.** In order to demonstrate the physical validity of the generated automata, we have fabricated two examples, one for each of the major types of layout configurations as shown in Figure 6.8: the *walking* sequence consists of parallel motion planes and does not require any bevel gears; the *dancing* sequence requires to convert motion to two different planes. In this example, in addition to the torso of the figure, the shoulders have also been used as support structure since the arms move parallel to the transverse plane of the figure. For both examples, to propagate the rotation of the driver gear to each kinematic chain, pulleys are used to fill up the space between the driver gear and the root of the chain along the support structure.

---

<sup>1</sup><http://www.duygu-ceylan.com/duygu-ceylan/mechAuto.html>

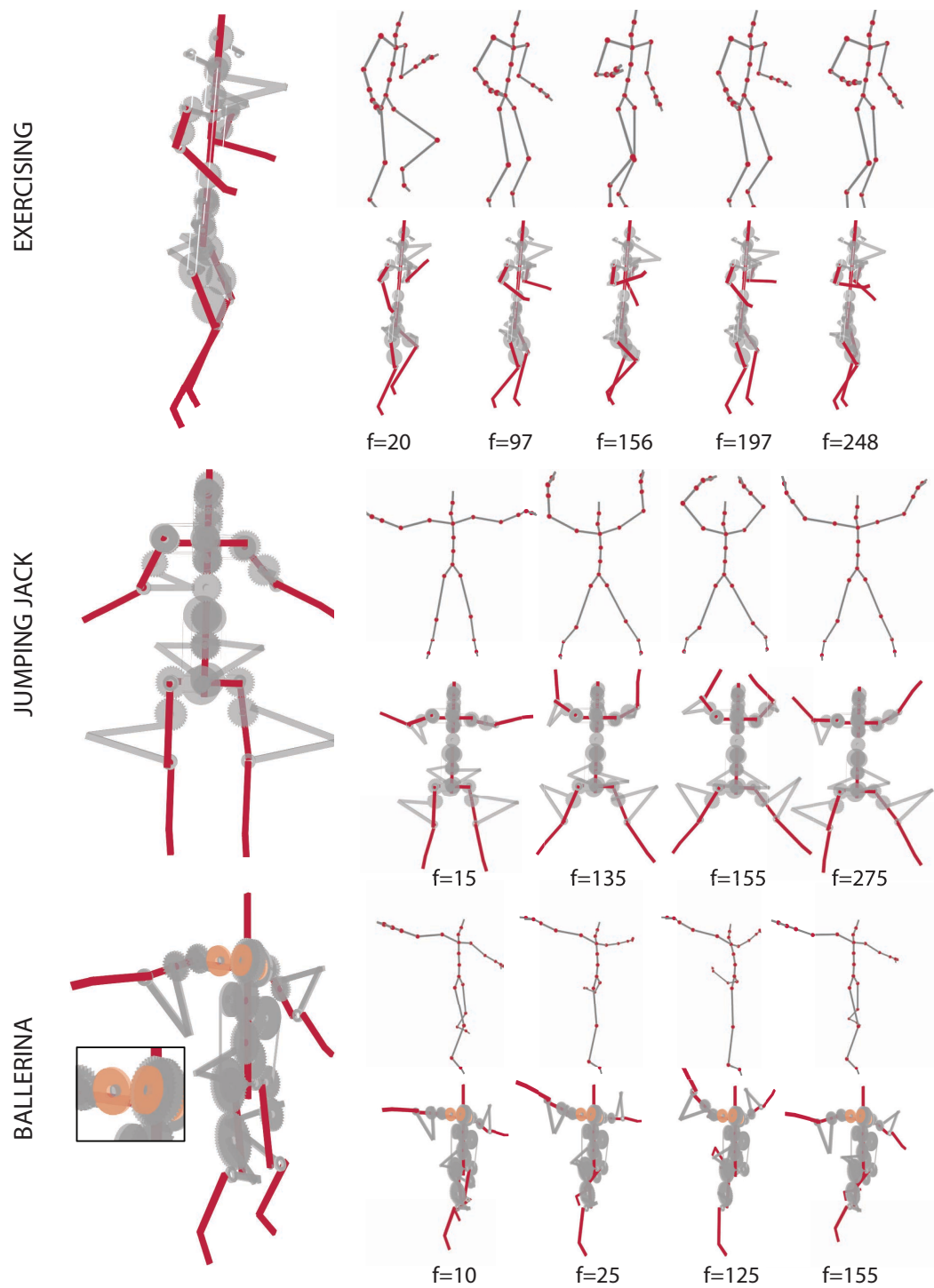


Figure 6.7: For each of the example automata generate by our system, snapshots of the original motion sequence and the corresponding simulation result of the automata are given for various time frames.

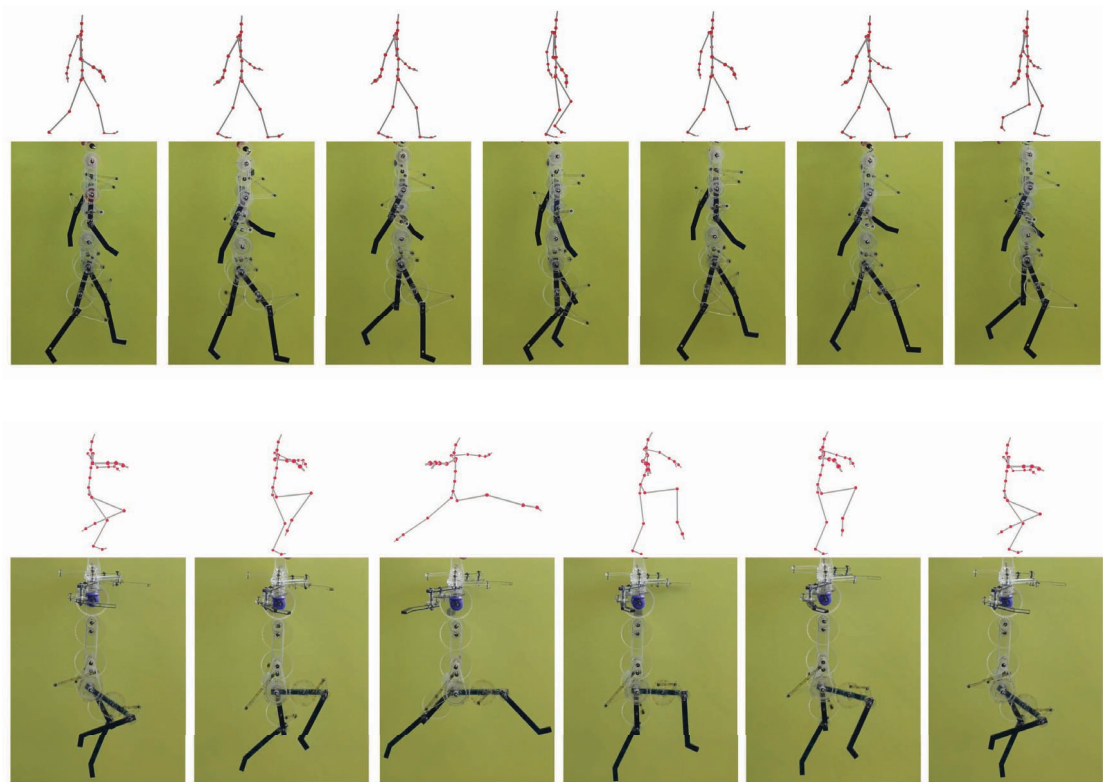


Figure 6.8: Physical prototypes of the *walking* (top) and the *dancing* examples show several snapshots of the input figure performing the desired motion and the corresponding mechanical automata.

In these designs, we use off-the-shelf pieces for the bevel gears, pulleys, and pins. The linkages and gears are laser cut based on the dimensions as prescribed by our optimization. The most time consuming part is often assembling the pieces, which is done manually. However, given that our design consists of similar modular components (i.e., the oscillation modules) the assembly process is relatively straightforward (albeit somewhat tedious). We use a single speed motor to drive the resulting automata.

**User control.** Although we propose a fully automatic system, we point out various ways the user can control the final design.

(i) Depending on the target fabrication method, the user can provide additional constraints. Based on the specification of our laser cutter, we used minimum and maximum gear tooth counts of 14 and 80 (given a 1mm tooth size). These constraints have a direct effect on the realization of the desired motion, specifically for links that require a large change in input angular speed.

(ii) Our system restricts the motion planes of the limbs to be parallel to the saggital, coronal or transverse plane of the figure. While we automatically generated the motion planes in our experiments, the user can also manually specify the motion plane orientations.



(iii) Finally, the user can set a threshold for the minimum angular range for the motion of the bones (0.35 radians in our results). We set bones oscillating with an angular range less than this threshold to be fixed, thus reducing the complexity of the generated mechanisms.

## 6.4 Closing Remarks

In this chapter, we have presented an algorithm to automatically realize a mechanical automaton starting from an input mocap sequence. We have first analyzed the characteristics of typical target motions and the requirements of the mechanical components. We have then designed an oscillation module that can generate kinematic chain motions with links that oscillate at different phases and frequencies. We propose a motion approximation algorithm that converts an input sequence to a mechanically realizable motion through the use of such modules. We automatically determine the parameters and spatial layout of the mechanical parts that are necessary to realize the target motions while taking into account physical and fabrication constraints.

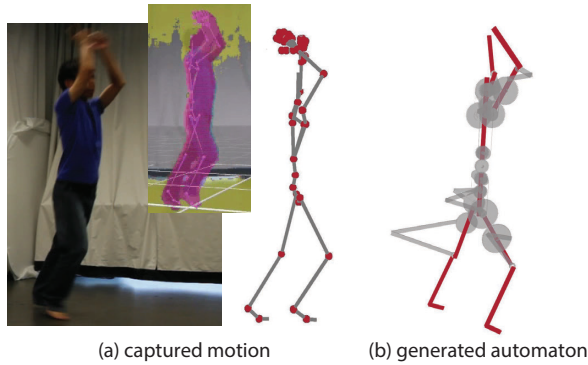
We demonstrate examples with non-trivial human motions with multiple moving limbs. We enforce the use of standard planar mechanical components to simplify the fabrication and assembly processes. This restriction results in various aspects of certain motions that our method fails at reproducing accurately. We believe that further investigations about other mechanism types, such as five- or six-bar linkages, that can potentially increase the motion approximation quality are valuable. In addition, our current method optimizes for the motion parameters of the mechanism first, and then solves for the layout of the parts. It is possible that a joint (or perhaps iterative) optimization of these parameters could produce better motion approximations in the final automaton designs.

In the motion approximation part of our method, we use an objective function based on the rotation angles of each joint. Our intuition is that the changes in joint angles have a significant effect on how we perceive a motion since they are visually well exposed in the static body posture. Developing better perceptual metrics for comparing human motions, however, is an interesting avenue for future work.

We assume periodic motions as input to allow driving the automaton indefinitely with a constant-speed input rotation. While we did not observe noticeable drift for extended operation of our assemblies, machining imprecisions or mechanical wear could potentially cause temporal deviations that over time reduce the accuracy of the motion approximation. At a certain point, the automaton might have to be partially disassembled to re-align the parts to the original configuration.

Currently, we have focused only on the kinematics of the generated automata, trying to realize the input motions as close as possible. It will be interesting to consider the stability aspect of these mechanisms and investigate the design of self-standing mechanical figures that remain balanced while they move. Generating such automata requires a method that accounts for both the stability

of the design as well as the target motion. One challenge here is that the placement of mechanical components (e.g., gears, linkages) affects the weight distribution and thus the stability of the automaton.



In our examples, we have primarily used motions selected from the existing motion databases as input. However, recent advances in commercial motion sensing input devices such as the Microsoft Kinect enable easy tracking of the human skeleton and thus provide an interface to directly capture input motions. Being a fully automatic method, we believe that our system has potential to be applicable to such captured input motions. We demonstrate a first result along this direc-

tion with the kendo sequence shown in the inset figure. For this example, we have used the Kinect to record a performance and the commercial tool ipi Soft to extract the human skeleton with the commercial tool ipi Soft [2013]. We have supplied the extracted skeleton motion as input to our system.

We believe that integrating our system with a real-time motion capturing interface brings up many exciting future applications. Specifically, we envision that providing the users with real-time feedback about the quality of the motion approximation and the complexity of the generated automata will be invaluable. This feedback might enable the users to make slight changes in the desired motions while improving the quality of the physical recreations of the target motions. Providing such feedback obviously requires further investigations on the nature of the typical human motions and capabilities of various mechanical components. We believe that such advanced analysis will play a crucial role in bringing the digital and the physical worlds further closer.

Finally, we believe that mechanism design is a process that can benefit from a data-driven approach. Specific combinations of mechanical building blocks reoccur in realizations of similar motions. For example, a slider-crank mechanism, commonly used in piston engines, is a building block used to convert rotational motion into reciprocating translation. This observation is almost similar to the analysis presented in the previous chapter to detect geometric similarities between the elements of a building. Mechanical building blocks can be considered as the *templates* we desire to fit to the given input motions. Based on this analogy, *functional similarity* between motions can be defined as the similarity between the type and the parameters of the mechanical buildings blocks used to create them. We believe that investigating the concept of functional similarity is an exciting research direction.

# 7 Conclusion and Future Work

This dissertation has presented shape analysis algorithms aimed at facilitating the 3D reconstruction and modeling tasks while simultaneously performing structure discovery in the underlying content.

We have focused our investigations mainly in the domain of image-based reconstruction of urban data sets. Special emphasis was put on exploring *symmetry* as the main analysis tool due to its ubiquity in such data sets. We have presented coupled approaches for simultaneous 3D reconstruction of buildings and detection of symmetry priors.

In the latter part of this dissertation, we have extended our analysis methods to a different problem domain, namely automated design and fabrication of mechanical figures capable of producing motions such as walking and dancing. We have demonstrated that shape analysis is a crucial component for addressing this problem as well. Specifically, we have made design decisions by analyzing the characteristics of the target motions and the requirements of the physical fabrication process.

This chapter summarizes our main findings and suggests several directions for future explorations.

## 7.1 Summary and Take-Home Messages

**Coupling Reconstruction and Analysis:** Many digital applications require accurate reconstruction of the physical world together with a high-level shape analysis of the underlying content. On the one hand, obtaining an accurate and high-quality representation of the physical world is necessary to create precise replicas of real-world objects for purposes such as prototyping, reverse engineering, or industrial simulations. On the other hand, automatic extraction of semantic knowledge from the digital content similar to a human being is crucial. To illustrate, given a 3D reconstruction of a building, discovering patterns in the geometry and the spatial arrangement of its elements facilitates many editing tasks.

## Chapter 7. Conclusion and Future Work

---

A central theme in the algorithms we have presented is to handle these accurate reconstruction and high-level shape analysis tasks in a coupled approach. Our main motivation stems from the fact that these tasks are complementary for each other. In the context of 3D urban reconstruction, this coupling also enables to break the problem of cyclic dependency between reconstruction and structure detection: While consolidating observations across symmetric and repeating elements enables us to remove noise and fill holes in 3D reconstructions, accurate and clean data measurements are necessary to facilitate detection of these structures.

Our algorithms focus on simultaneously performing structure discovery and 3D reconstruction. We demonstrate that combining early estimates about the structural priors and intermediate 3D reconstructions into a unified framework helps to refine both of these initial measurements.

**Use of Suitable Priors:** A feature common to all the algorithms we have presented is to use specific priors suitable for the target problem domain.

For the purpose of 3D reconstruction of urban data sets, we have utilized mainly two types of priors. While we have explored line and plane features as dominant geometric priors in such data sets, we have explored structural priors in the form of repetitions of the individual building elements. Geometric priors enable to capture characteristics of the underlying data such as sharp facade corners. Symmetry priors, on the other hand, provide non-local coupling between multiple observations of the same geometry.

Similarly, for the automated mechanism design problem, we have decomposed target motions into a set of planar motions. This planarity assumption enables us to realize the desired motion with standard planar mechanisms and thus facilitates the fabrication and assembly processes.

At this point, we would like to emphasize the trade-off between the generality and the performance of the proposed algorithms. While priors are effective in tackling challenging tasks, they typically impose limitations on the applicability of the proposed approaches. For example, we demonstrate that the reconstruction algorithms we have proposed in Chapters 3 and 4 outperform general purpose dense [33] and sparse [103] reconstruction methods. On the other hand, these algorithms explore the presence of dominant repetitions in the captured scenes and thus are as effective as the underlying repetition patterns.

Despite the limitations imposed, however, we encourage the use of suitable priors for many challenging problem domains. A good set of priors often provides natural assumptions and generalizes elegantly to many practical use cases. Similar to the geometric and structural priors we have explored in the context of 3D reconstruction of buildings, cuboid proxies for editing of images of man-made environments [124] or smoothness and visual hull priors for shape completion [57] are used as such priors. We believe that suitable prior information effectively improves the quality of the processing results for many ill-posed problems.

**Reformulation in Alternative Problem Domains:** For some challenging geometry analysis problems, it is possible to map the problem into a different space where specific relations become

more apparent. This mapping between the problem domains can be considered similar to the Fourier transform which has been a cornerstone in signal processing applications to transform signals between the time and frequency domain.

In context of shape and geometry analysis, one elegant examples of such a mapping is the structural regularity detection framework of Pauly et al. [84]. Detection of repetition grids in 3D geometry is performed by mapping the problem to the space of transformations and detecting resulting characteristic grids in this transformation space.

We employ a similar approach for detecting structured variations between the elements of complex architectural data sets (see Chapter 5). It is challenging to detect such variations by comparing the geometry of the individual elements, especially in case of noisy and partial data measurements. Instead, we reformulate this problem as detecting similarities in the deformation modes of a set of template models. The advantages of mapping this problem to the deformation space of template models are twofold. First, deformable template models enable us to capture structured variations of a base geometry which are otherwise difficult to characterize. Second, we can accumulate observations from different templates in the deformation space making our approach more robust to noise and outliers.

We believe that similar problem reformulations and mappings might be possible for other shape analysis tasks and thus encourage the reader to perform investigations along this direction.

**Combination of Multiple Data Representations:** With the advances in acquisition technologies we witness an increase in variety of digital data sources. In addition to 2D data representations such as images and videos, 3D data acquisition methods provide digital data in the form of polygonal meshes or point clouds. In another thread, motion capture is becoming more convenient with the advent of commercial depth sensing devices such as the Microsoft Kinect.

In most of our algorithms, we have used images and intermediate 3D reconstructions obtained from these images in an interleaving manner. We believe that each of these data representations have specific advantages. Images are often high-resolution and thus better capture fine details of the underlying geometry. However, features extracted from the images often also contain noise and outliers and lack depth information. 3D data representations, on the other hand, provide depth information and act as a bridge to link multiple images. By accumulating observations across multiple images, they enable to distinguish between the actual features and noise.

We have demonstrated an elegant approach for combining 2D and 3D line features in Chapter 3 for reliable repeating element detection. We show that accumulating 2D edges via a 3D line reconstruction step acts as a filter to prune out noisy features. On the other hand, features missed due to aggressive pruning are later recovered by leveraging the 2D edges via smart hypotheses obtained from the 3D data. We believe that combining multiple data representations in the context of many geometry capture problems allows to benefit from their individual advantages.

### 7.2 Open Questions and Future Work

Accurate digitization of the physical world and extracting semantic information from the digital content remains a challenging and open problem. Therefore, we expect to see more future work in the domain of shape reconstruction and analysis.

We believe that there are certain immediate extensions to the algorithms we have presented in this dissertation. To illustrate, in our image-based reconstruction frameworks we have mainly explored planar repetitions arranged in 1- or 2-dimensional grids. Incorporating other types of repetition patterns such as rotational symmetries as found in dome-like structures is a possible extension.

A unique feature of the image-based reconstruction algorithms we have proposed is to explore the coupling between data acquisition and structure detection. Even though we have showcased results in the domain of urban reconstruction, we believe that applying similar principals for indoor reconstruction is a promising research direction. Similar to buildings, indoor environments are rich both in geometric and structural priors. For example an office environment consists of dominant plane features corresponding to the floor, walls, or table tops. In addition, certain objects such as chairs or bookshelves occur multiple times. We have seen some research efforts that explore these geometric [32] and repetition [53] priors. However, these approaches do not explore the coupling between the reconstruction and structure detection tasks. With advances in commercial depth sensor devices such as the Microsoft Kinect, we envision to perform detection of geometric and structural priors during the scanning process where the extracted priors are immediately used to improve the reconstruction quality. A recent effort along these directions is the SLAM++ system of Moreno et al. [94]. This system searches for objects that are scanned in a pre-processing stage in the incoming capture data. However, we believe that there is still a lot of room for improvement for such a system. For example, we predict that exploring geometric abstractions such as planes and cuboid proxies during an online scanning session provides compact representations enabling longer capture sessions. In addition, these geometric priors are expected to help registration of the incoming data frames more robustly thus reducing possible drifts in long captures.

We have presented interesting image-space interaction metaphors in Chapter 4 where certain image edits are automatically propagated to a collection of images using the extracted 3D scene information. We believe that collective editing of images is becoming an interesting and challenging open problem with the growing number of images taken at a specific event or environment. Developing such interaction possibilities requires analyzing large image collections and extracting depth information relating them. For images of man-made environments, however, use of cuboid like proxies to compute sparse depth information might be sufficient. This sparse depth information can later be used to link the input images and propagate edits performed in one image to the rest.

We have proposed an algorithm to detect structured variations among the elements of complex

architectural data sets. At the core of this algorithm is an approach that utilizes template proxies to analyze the geometric structure of the elements. We believe that such an analysis has potential to be extended towards the ambitious goal of automatic architecture style depiction. Geometries of the individual elements of a building and patterns observed in the spatial arrangement of these elements are among the key ingredients to specify an architecture style. Thus, we believe that detecting suitable matches from a database of template models and observing common deformation patterns provide important cues about the style of a building. However, there are certainly other aspects such as material properties or method of construction that are crucial for characterizing architecture style. Thus it is critical to extend shape analysis algorithms to capture such properties as well.

We believe that the notion of *style* is not restricted to architecture and in fact the proposed analysis algorithms can be extended to other data sets. Investigating this problem for depicting the style of furniture or even textiles is an interesting research direction.

Finally, the advances in sensing technologies indicate that we will be witnessing a revolution in how we capture and process data. Google has recently announced the *Project Tango* [39] that brings the possibility of having depth sensors in our mobile devices. In other words, in near future the images we take might be augmented with the additional depth information. We believe that this will create a new perspective for all the image processing applications facilitating many challenging tasks such as image segmentation or object detection.

Concurrent to the advances in data acquisition technologies, we are also experiencing rapid improvements in digital manufacturing. The ability to manufacture customized objects creates a demand for fabricating not only static objects but also functional models. We have demonstrated an early effort in automating the process of designing functional models focusing on the specific problem of mechanical automata design. We are expecting a growing demand for designing other types of functional models. In fact, we have recently seen research efforts in analyzing shapes from a perspective of functionality [125, 52]. Extending these analysis methods, we believe that adding functionality to static objects is an interesting research direction. A possible approach to tackle this problem is to adopt a data-driven approach. To illustrate, knowledge about how to open and close the doors of a car model might be useful to place joints to add functionality to the door of a plane model as well.





## A SfM Comparisons

In the following, we compare our symmetry-based SfM method to Bundler [103] and the method of Zach et al. [122]. For several data sets provided in Chapter 4, we include example input images and the user template (shown in orange) provided to guide the repetition detection. In order to make this comparison more clear, we demonstrate the dense reconstructions produced by PMVS [33], a state-of-the-art multiview stereo algorithm, using the camera parameters computed by each method. We also show the final 3D grid and its projection in several example images.

# Appendix A. SfM Comparisons

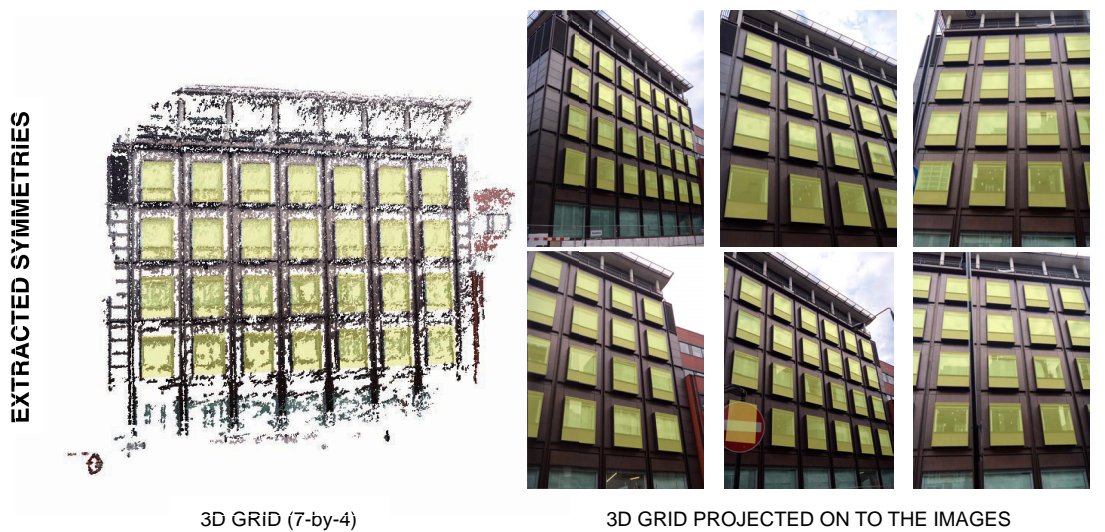
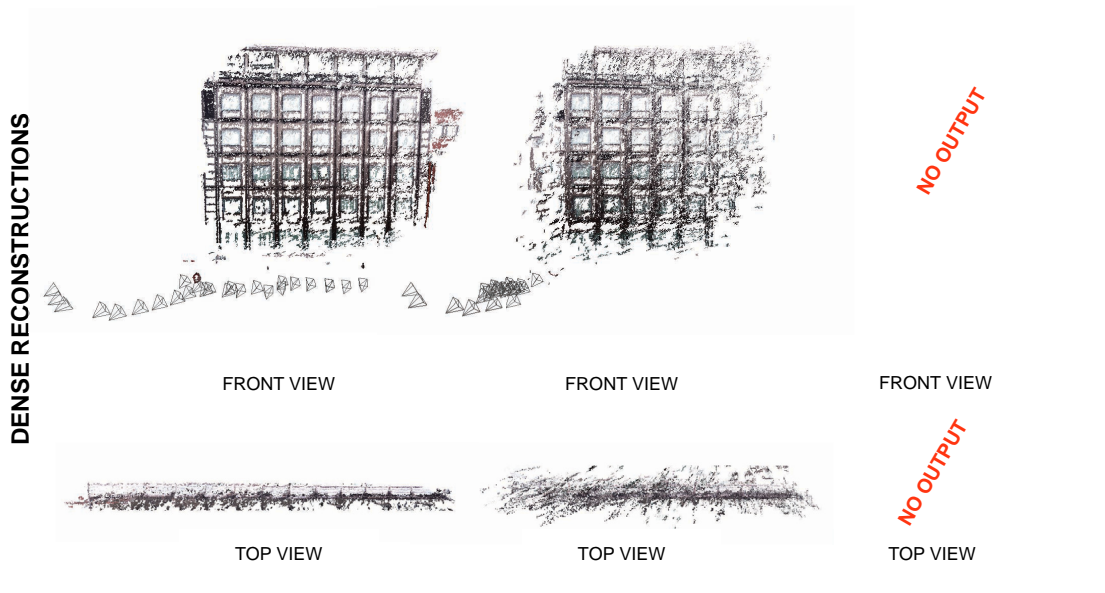
## BUILDING 1



OUR RESULT

BUNDLER

ZACH ET AL.'10

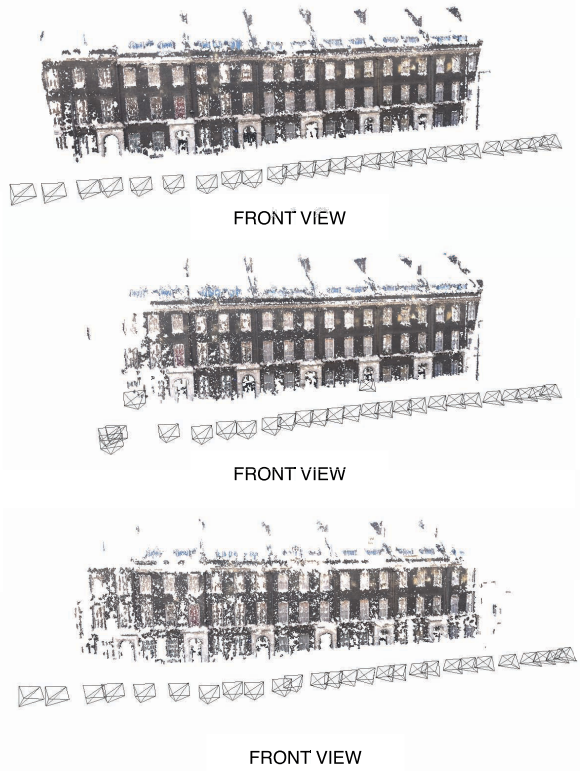


**BUILDING 2**

**INPUT IMAGES**



**DENSE RECONSTRUCTIONS**



**OUR RESULT**



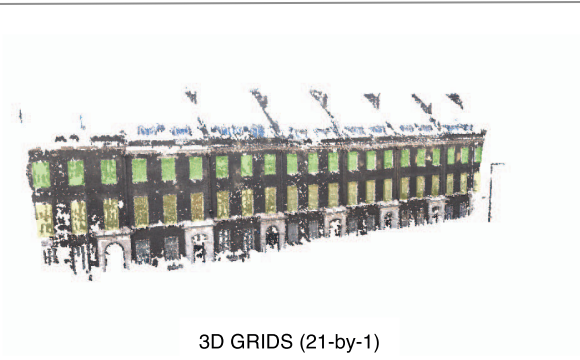
**BUNDLER**



**ZACH ET AL. '10**



**EXTRACTED SYMMETRIES**



3D GRIDS (21-by-1)



3D GRIDS PROJECTED ON TO THE IMAGES



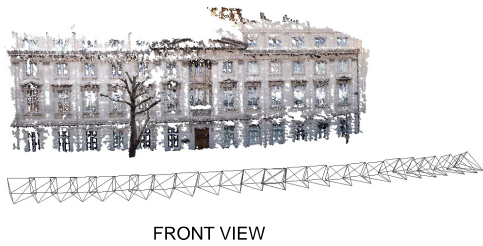
# Appendix A. SfM Comparisons

## BUILDING 3

INPUT IMAGES



DENSE RECONSTRUCTIONS

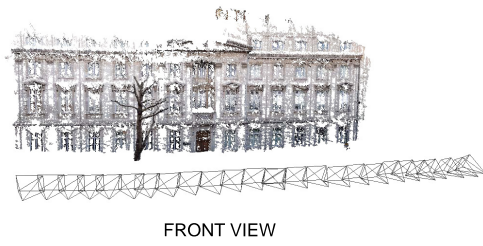


FRONT VIEW



TOP VIEW

OUR RESULT

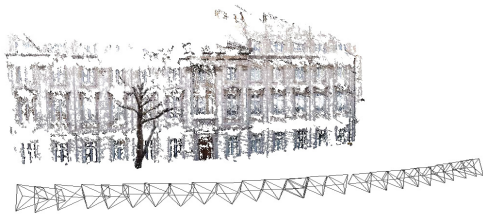


FRONT VIEW



TOP VIEW

BUNDLER



FRONT VIEW



TOP VIEW

CVPR10

EXTRACTED SYMMETRIES



3D GRIDS (5-by-1)



3D GRID PROJECTED ON TO THE IMAGES

**BUILDING 4**

**INPUT IMAGES**

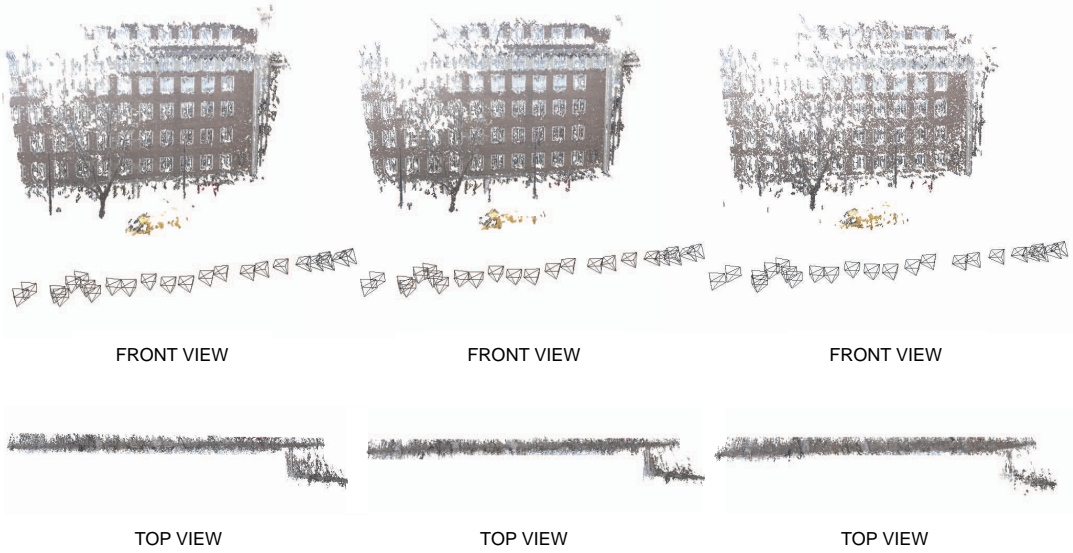


**DENSE RECONSTRUCTIONS**

**OUR RESULT**

**BUNDLER**

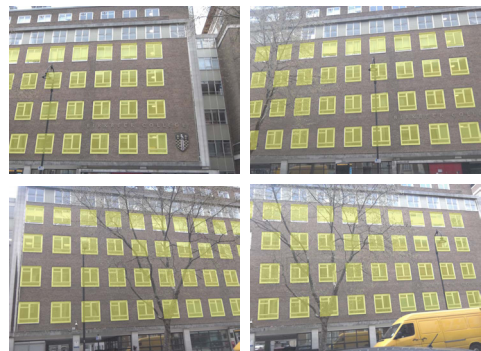
**ZACH ET AL.'10**



**SYMMETRY DETECTION**



**3D GRID (4-by-11)**



**3D GRID PROJECTED ON TO THE IMAGES**



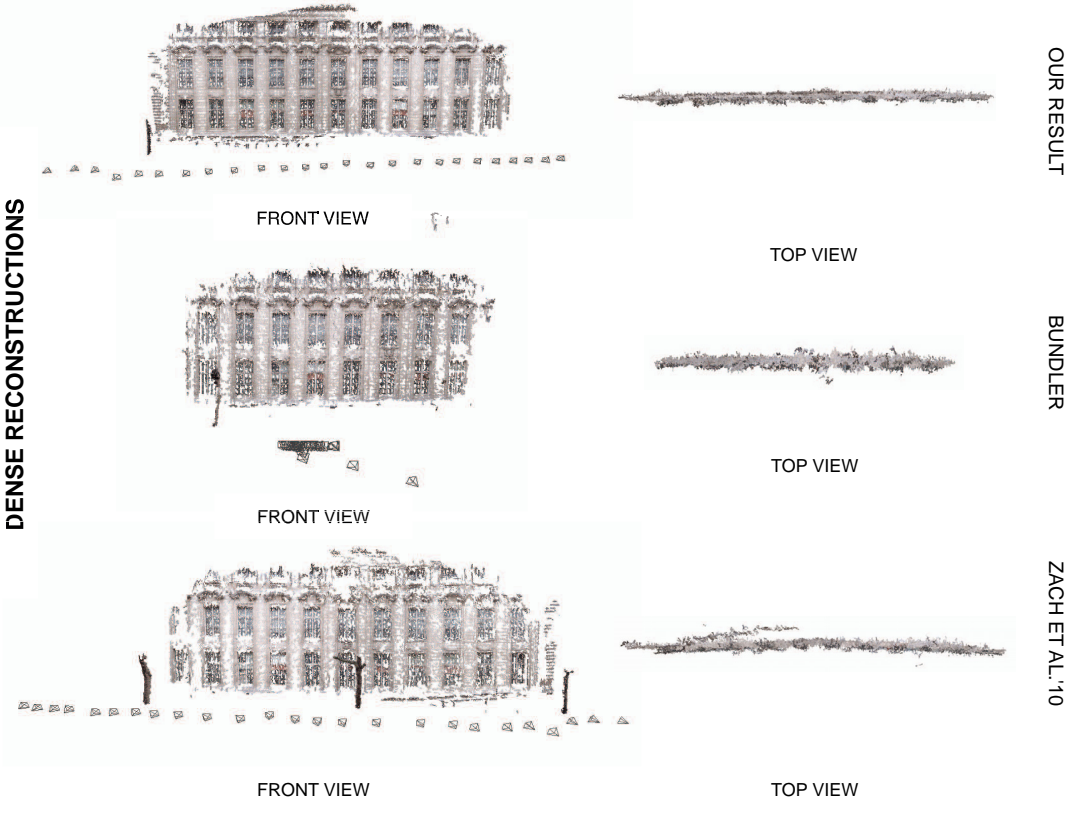
Appendix A. SfM Comparisons

BUILDING 5

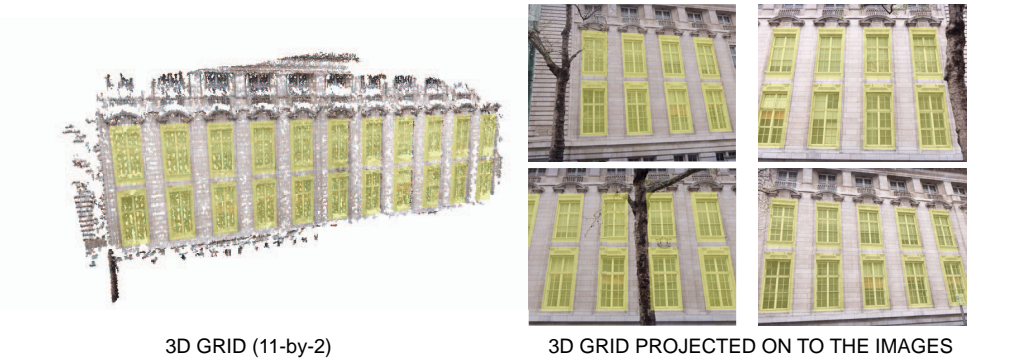
INPUT IMAGES



DENSE RECONSTRUCTIONS

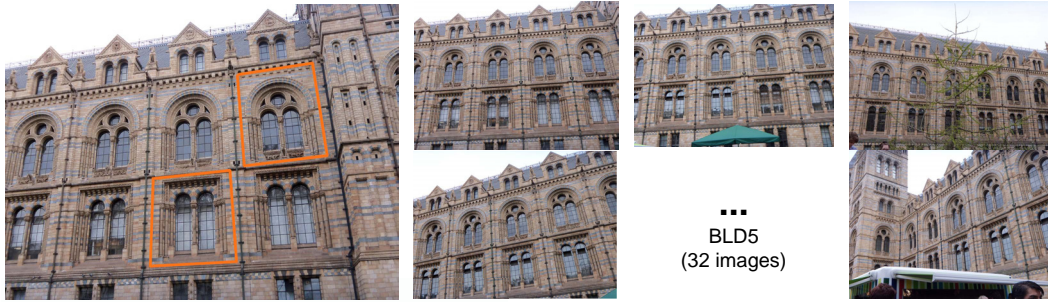


EXTRACTED SYMMETRIES

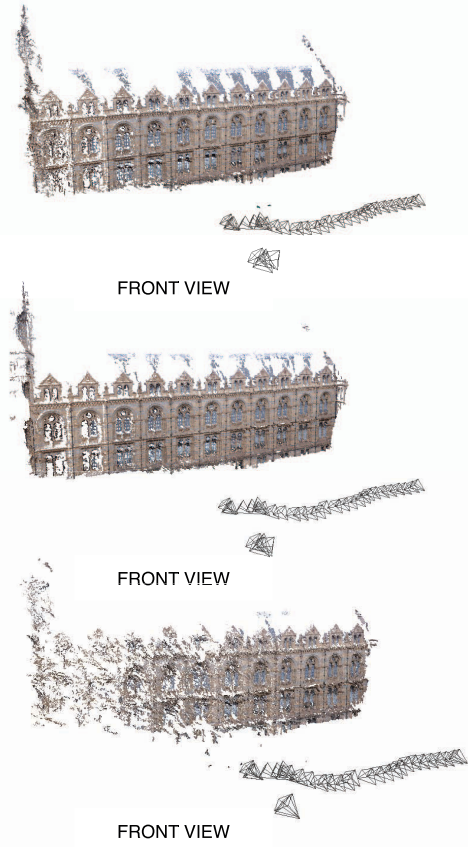


**BUILDING 6**

**INPUT IMAGES**



**DENSE RECONSTRUCTIONS**



TOP VIEW



TOP VIEW



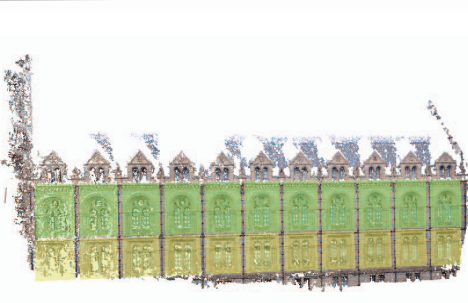
TOP VIEW

**OUR RESULT**

**BUNDLER**

**ZACH ET AL. '10**

**EXTRACTED SYMMETRIES**



3D GRID (11-by-2)



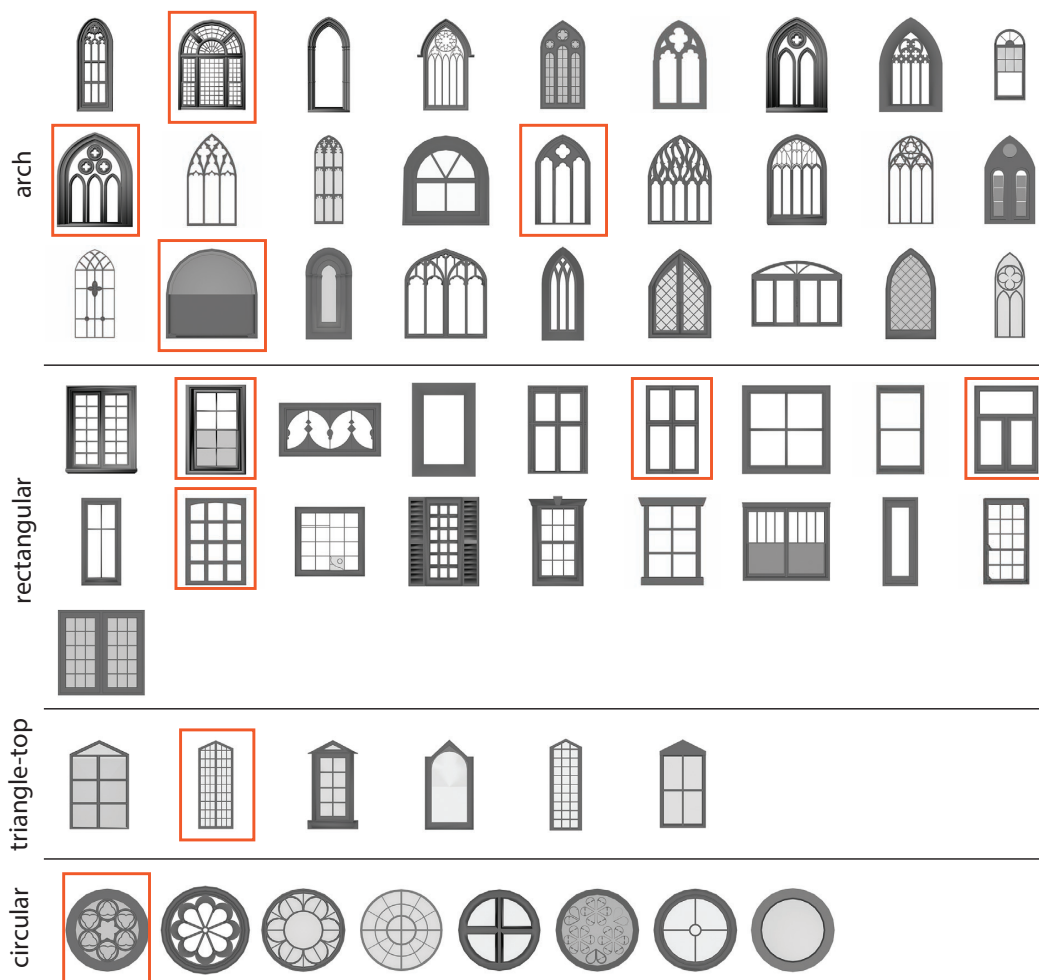
3D GRID PROJECTED ON TO THE IMAGES





# B Template Database

For the results presented in Chapter 5 of this dissertation, we have used the following window template models. Templates used for the evaluation in Figure 5.8b are shown in orange.





## **C Chapter 5 Detailed Results**

In the following, for each dataset we provide a selection of the input images, 2D edges detected in the images, and views of the computed 3D line features. We provide element smoothness matrices computed in the first and final iterations of our algorithm together with the detected element similarities. We further include close-up views of the templates instances matched to the elements.

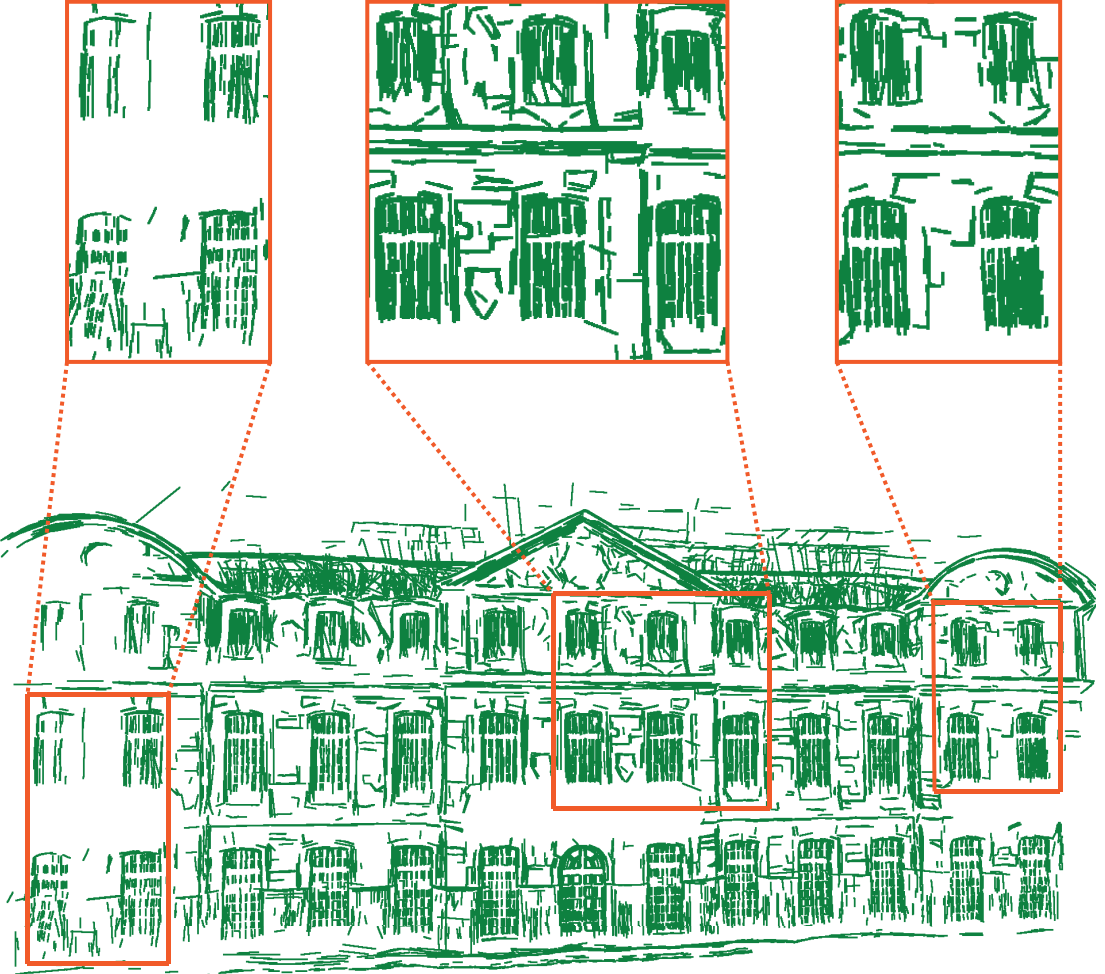
Building 1

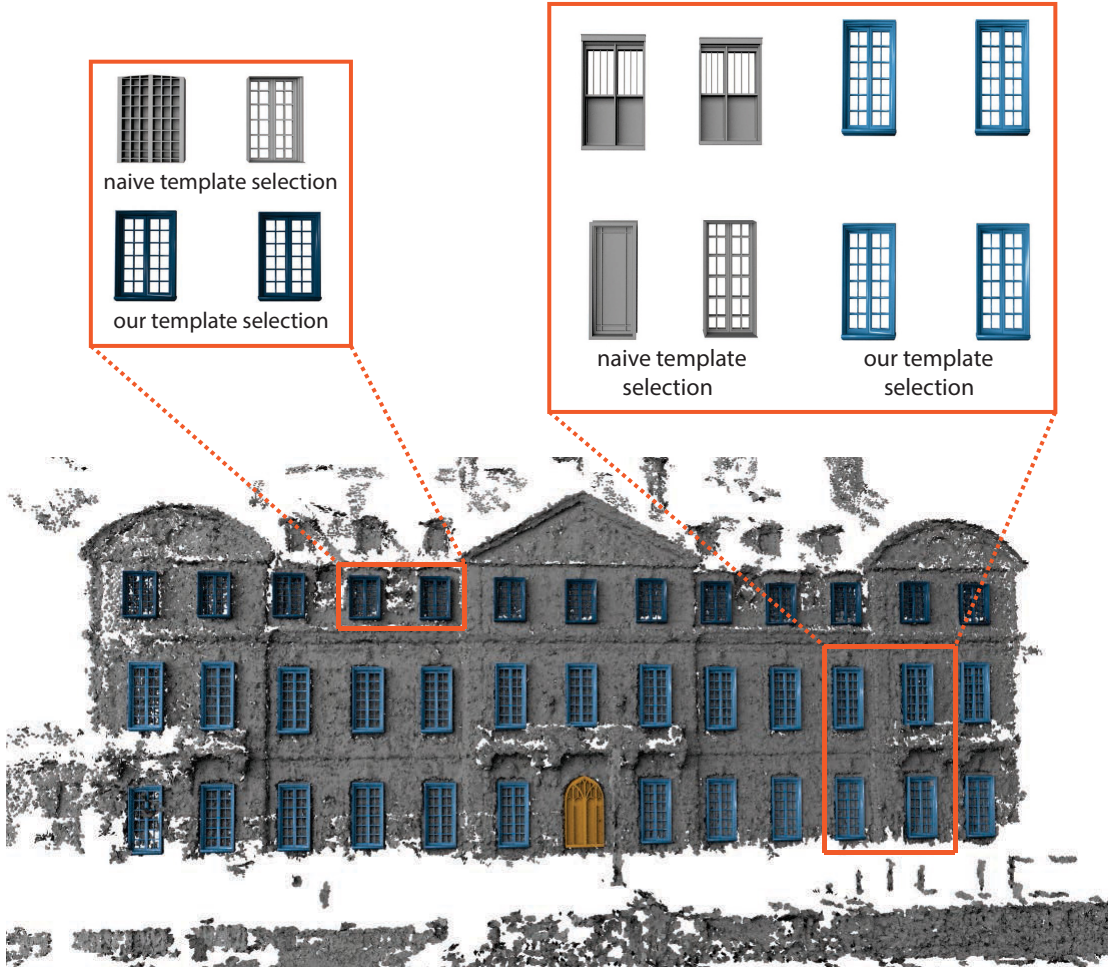
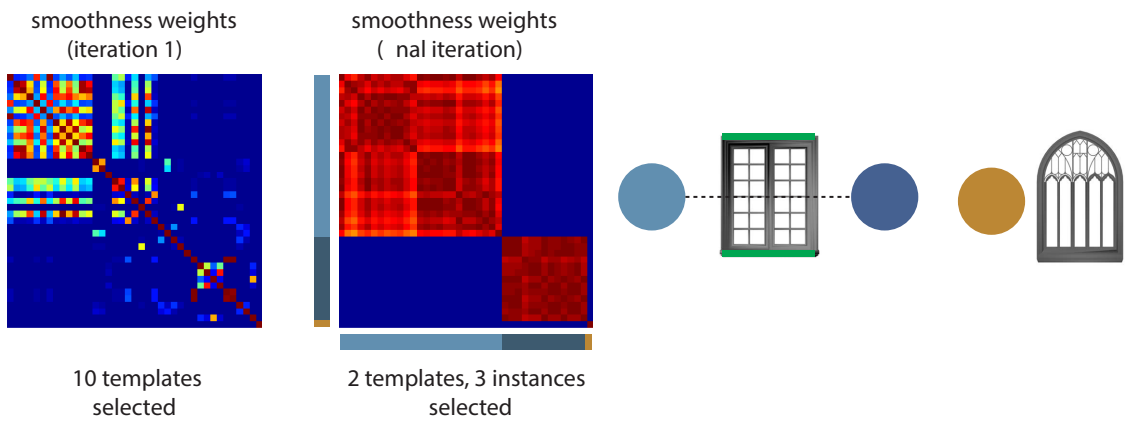


image edges



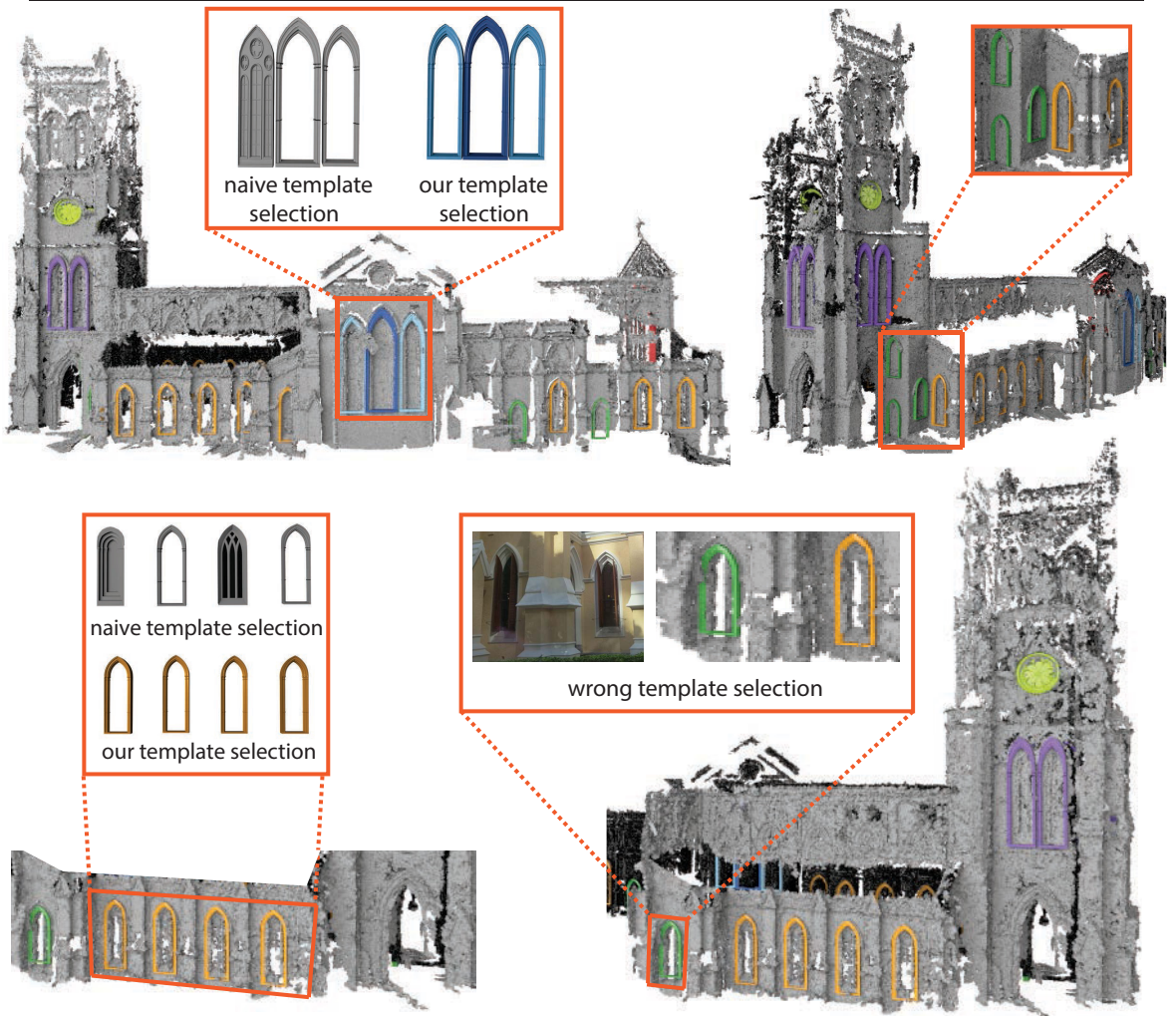
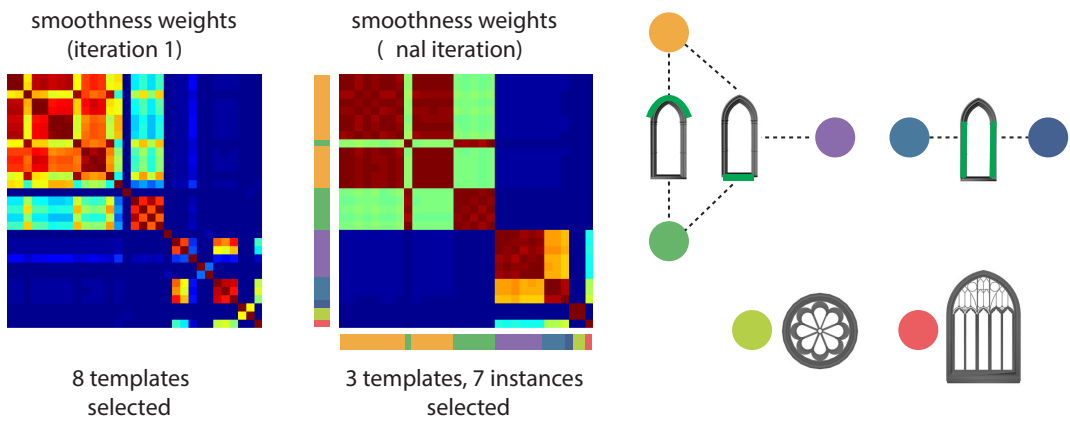
3D lines



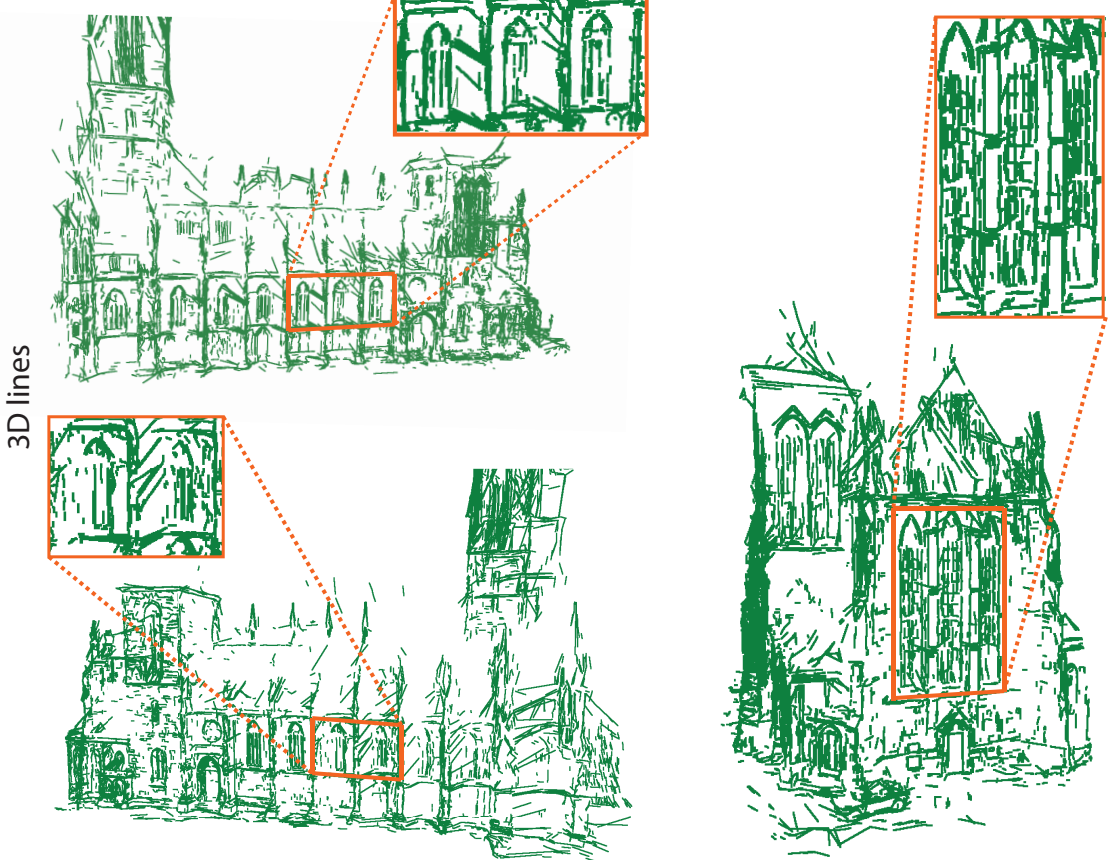




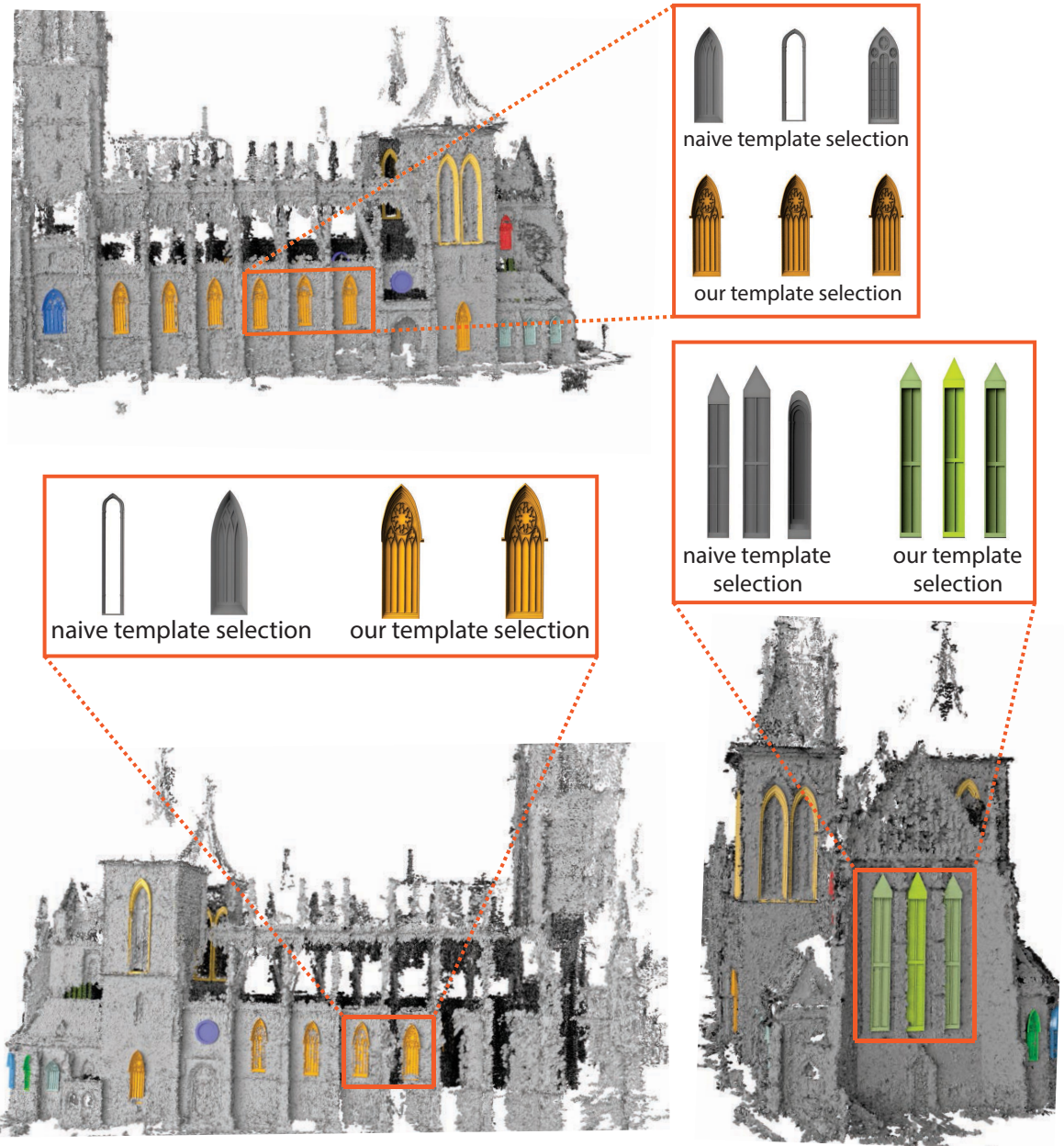
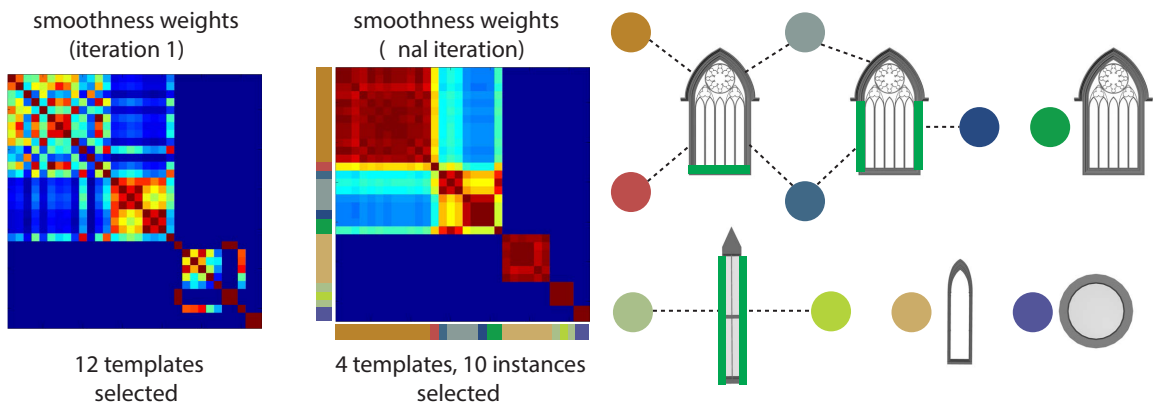












Building 4



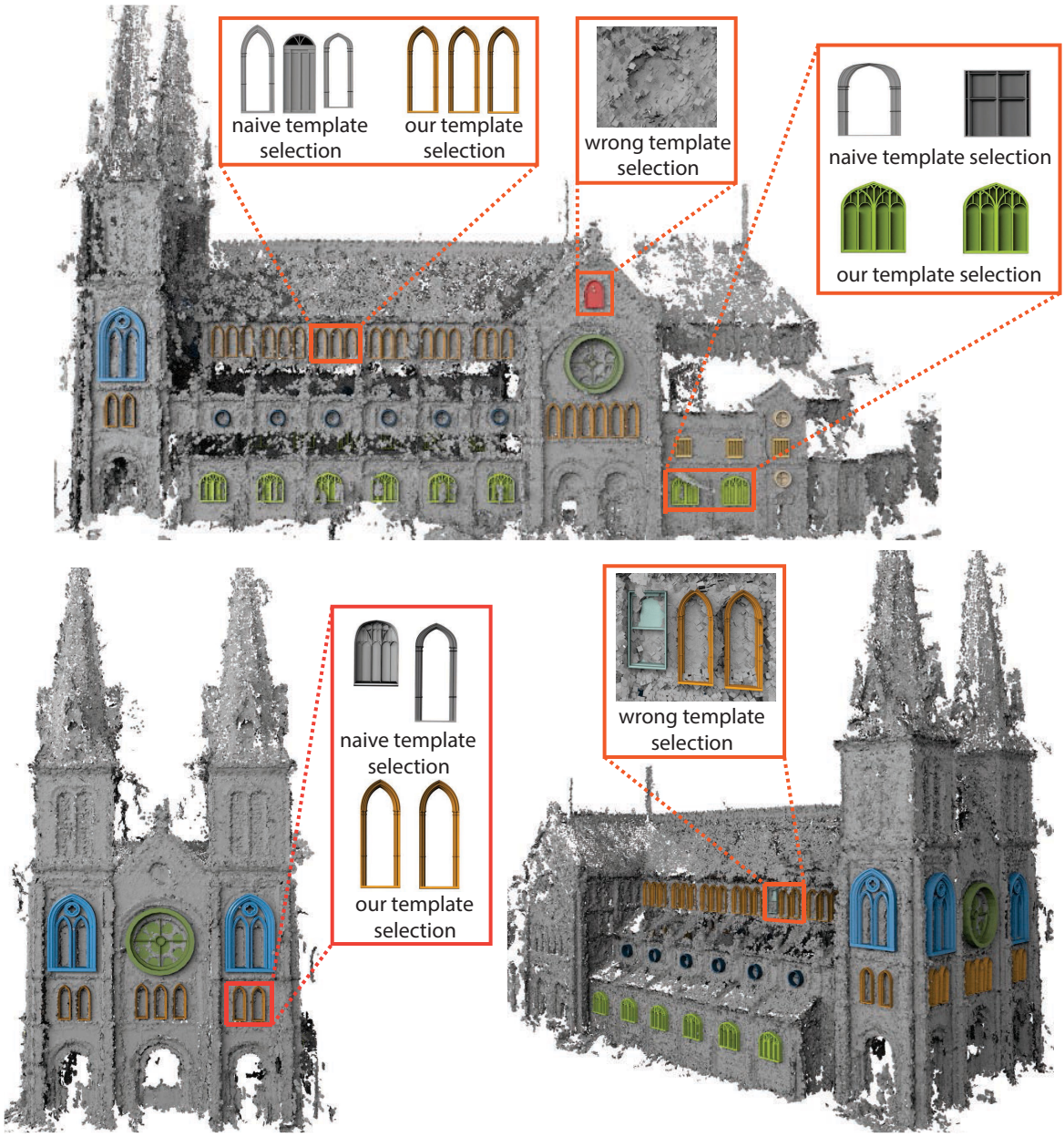
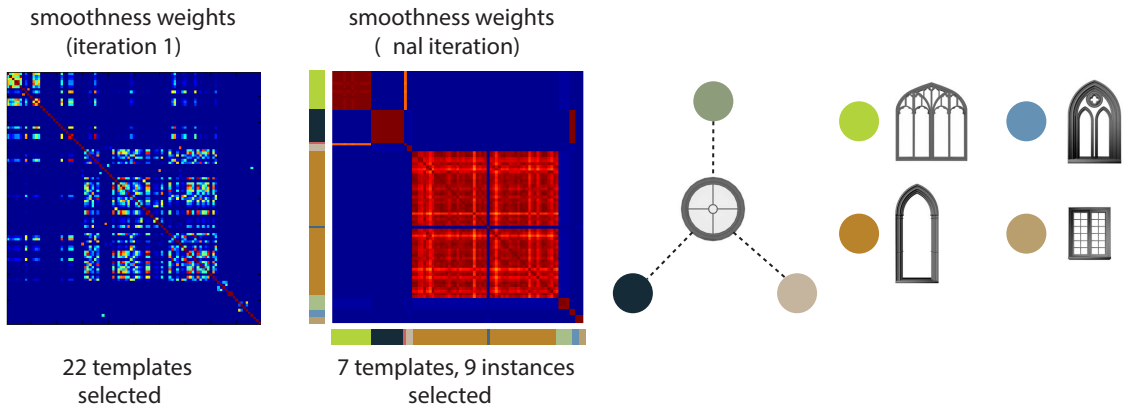
image edges



3D lines







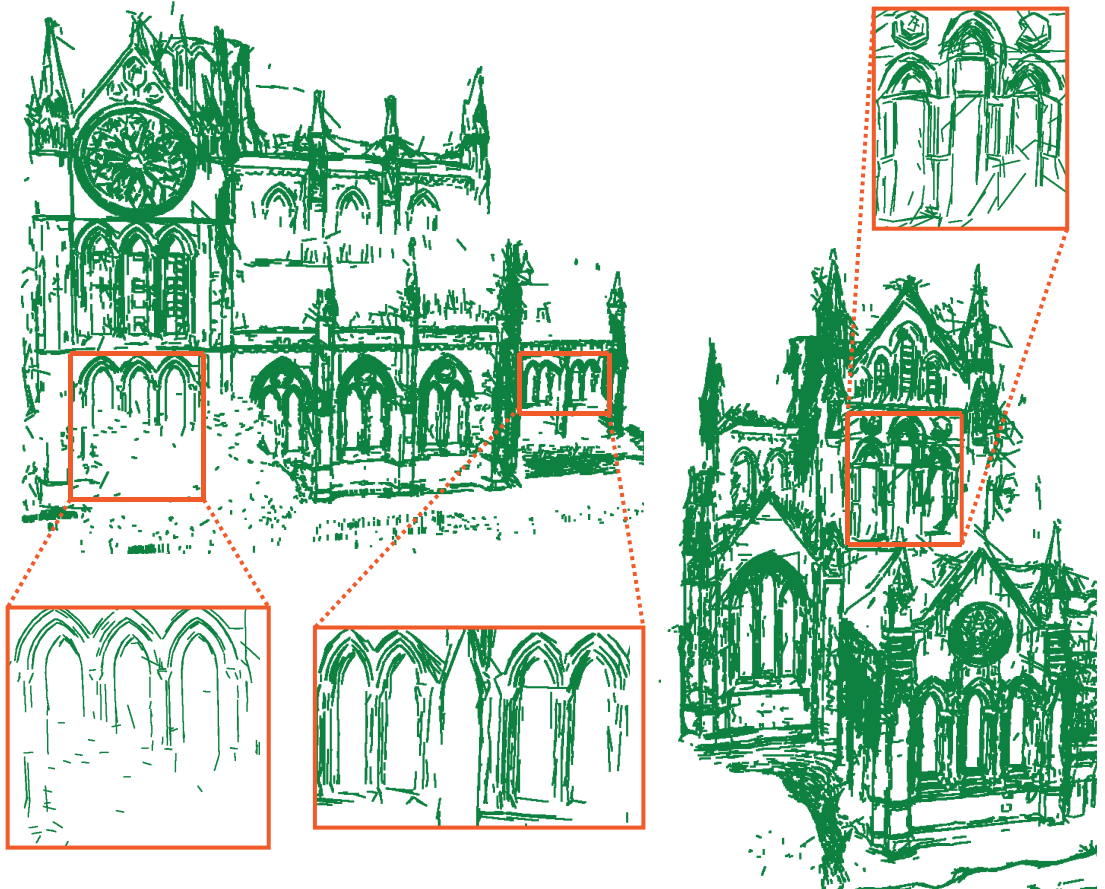
Building 5



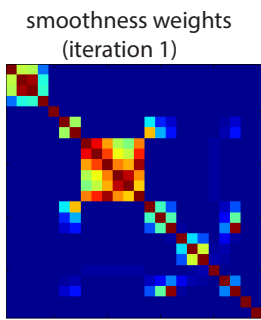
image edges



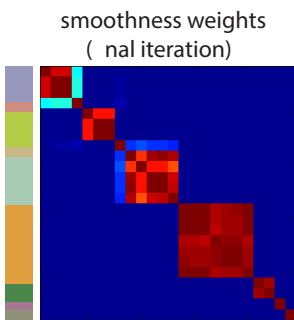
3D lines



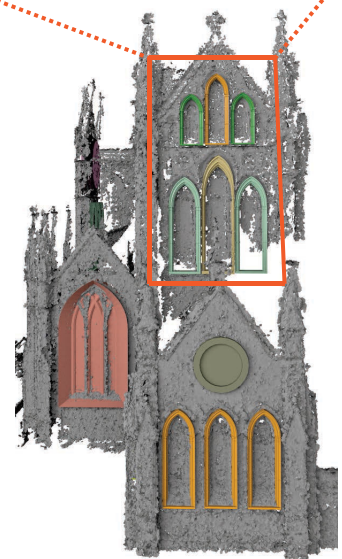
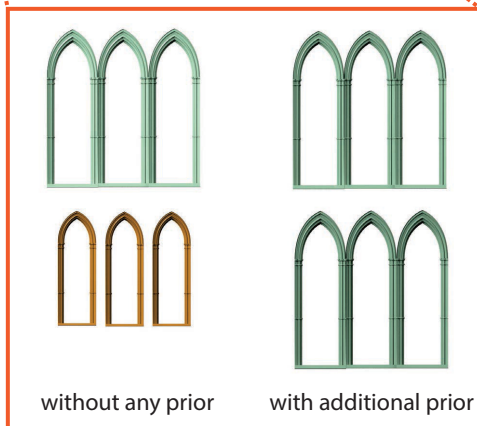
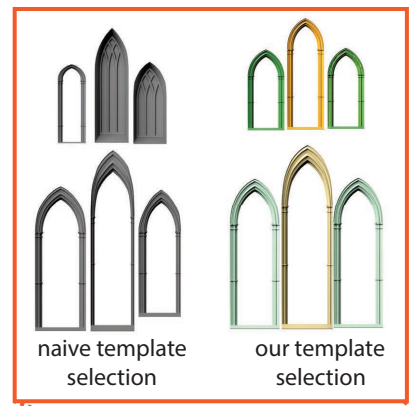
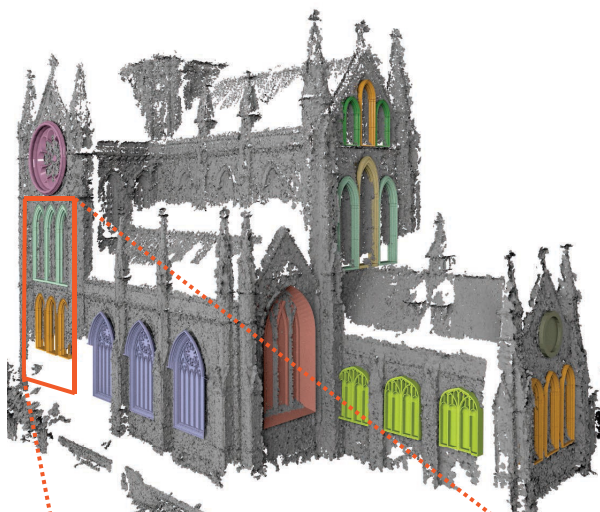
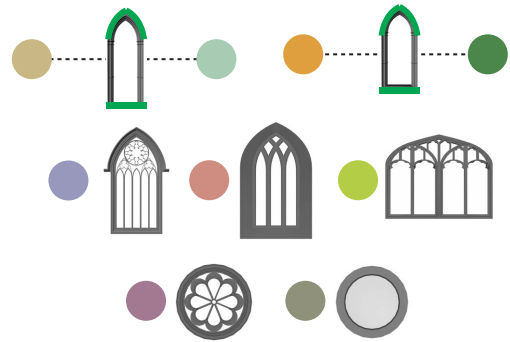




7 templates  
selected



5 templates, 9 instances  
selected



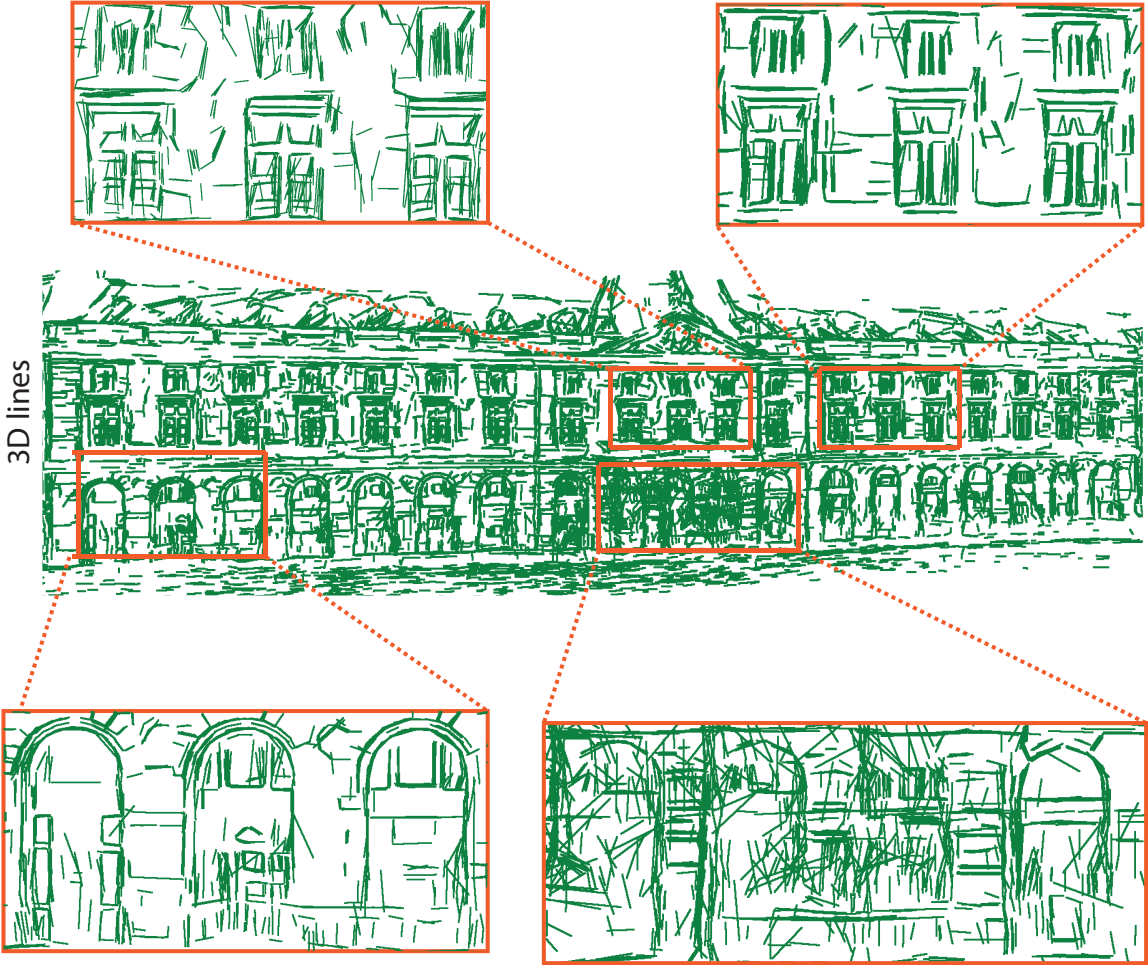
Building 6

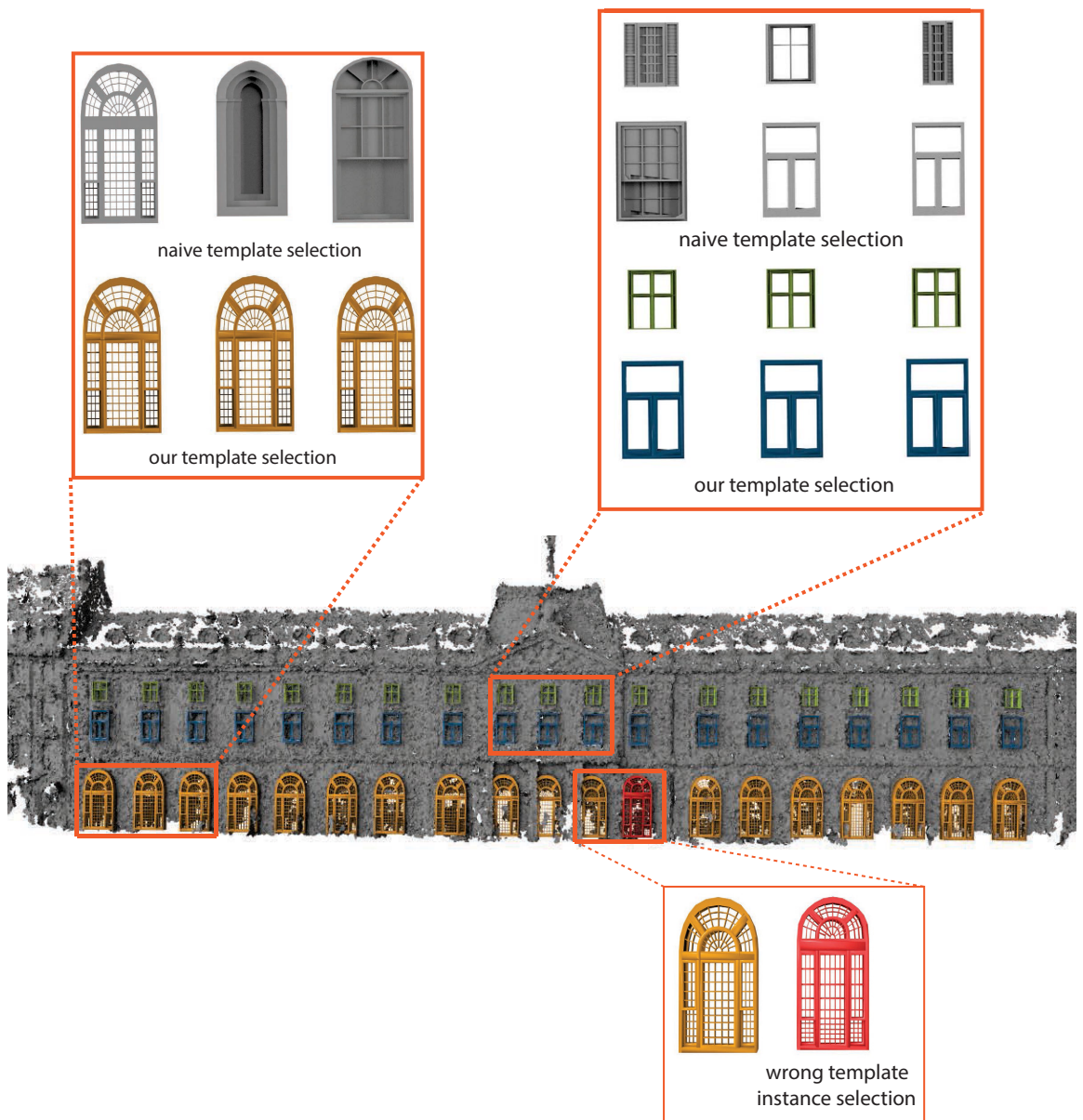
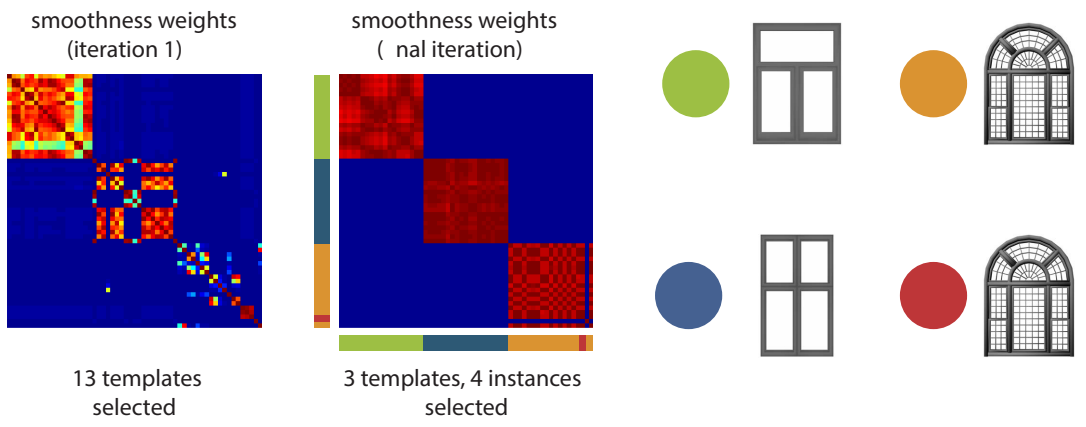


image edges



3D lines



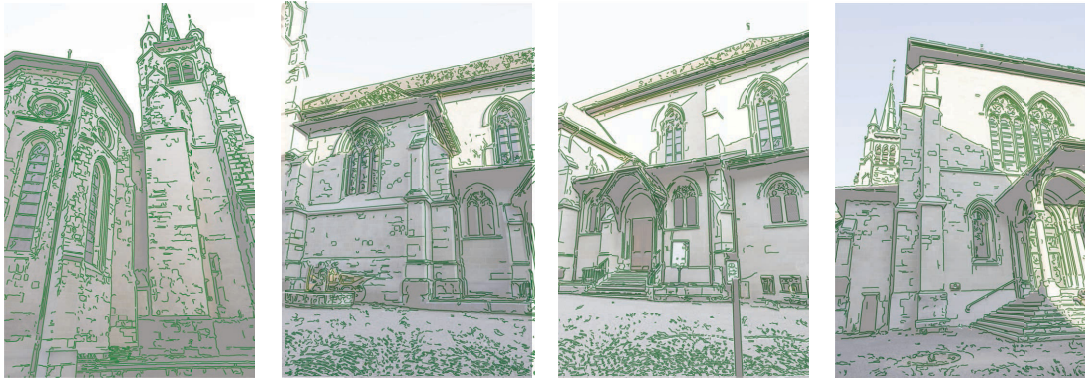




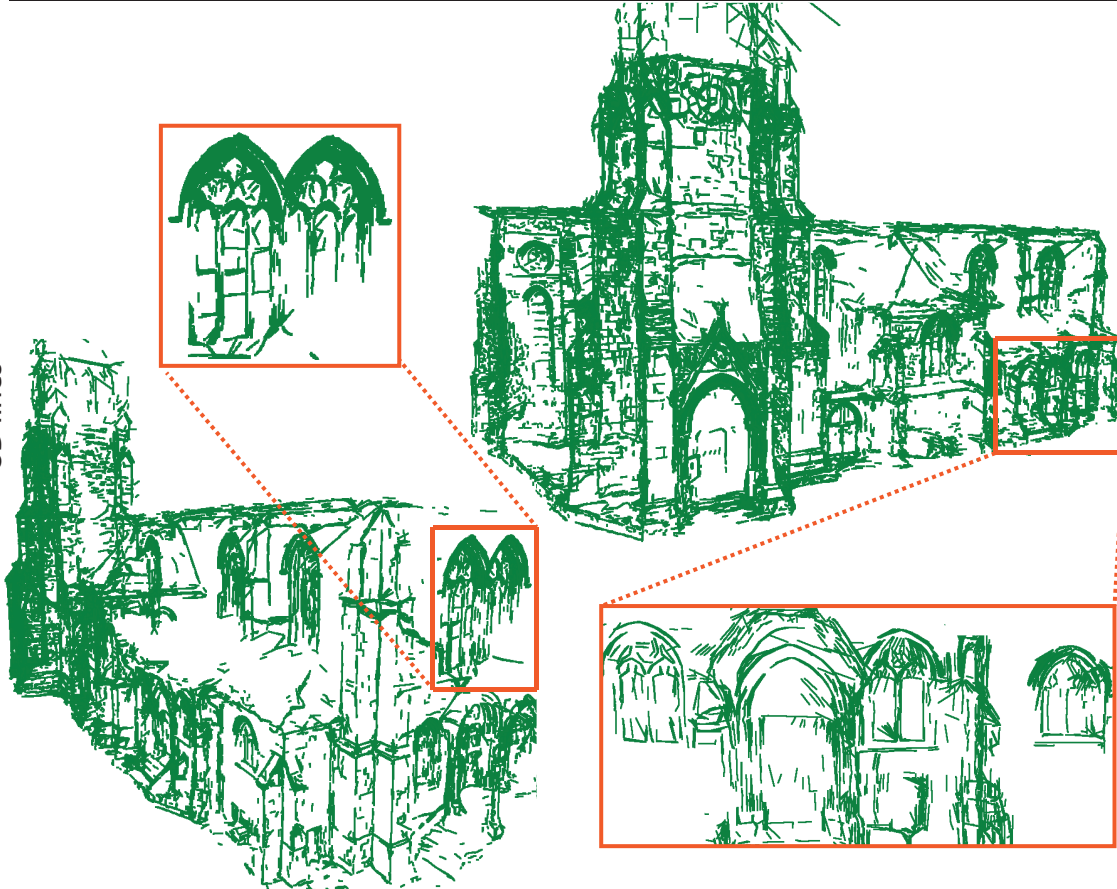
Building 7

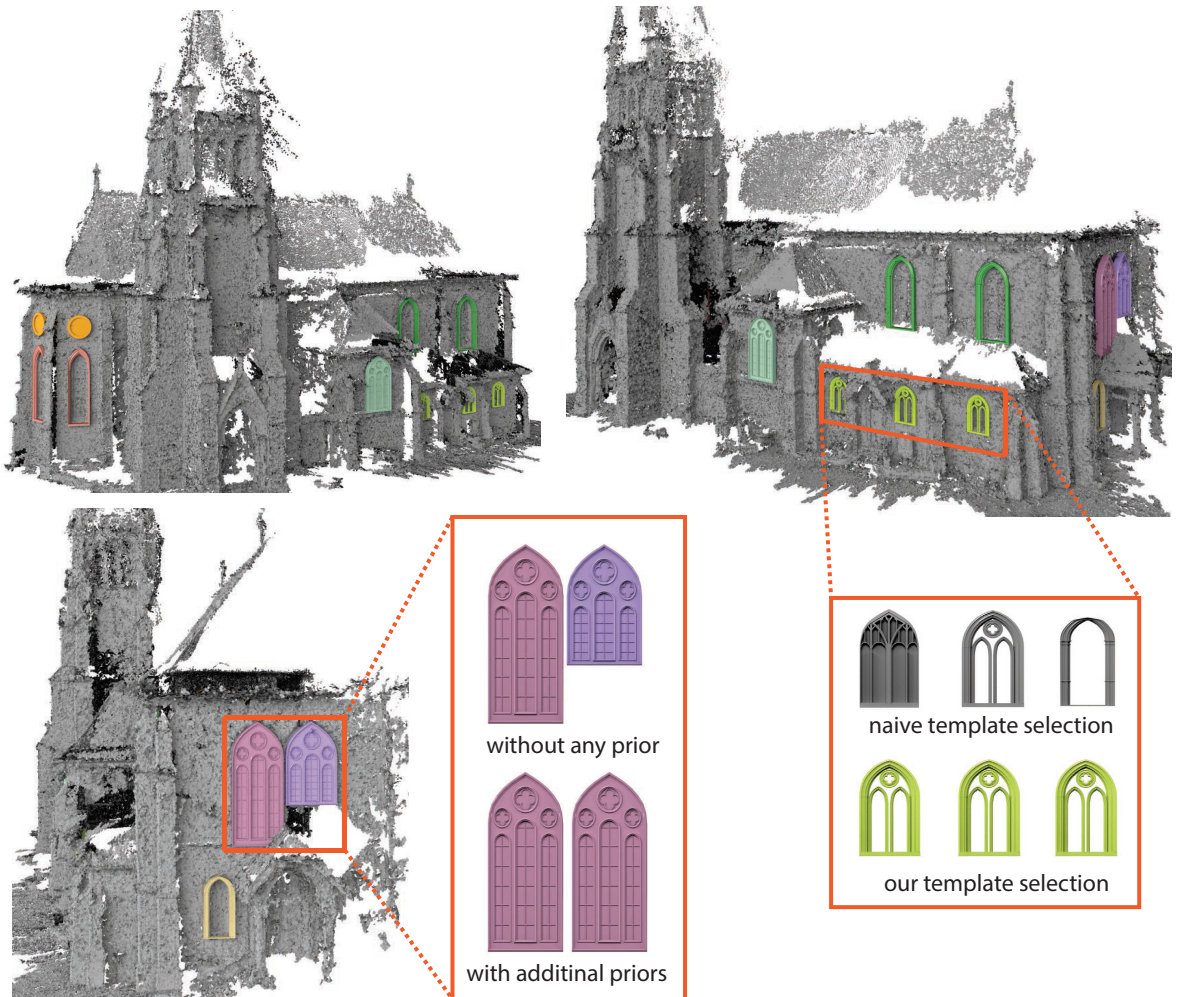
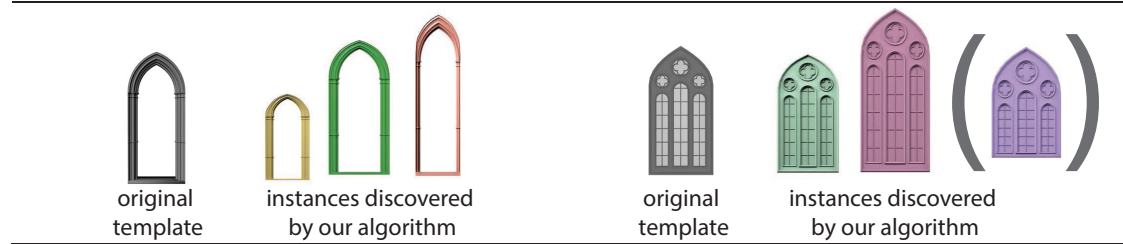
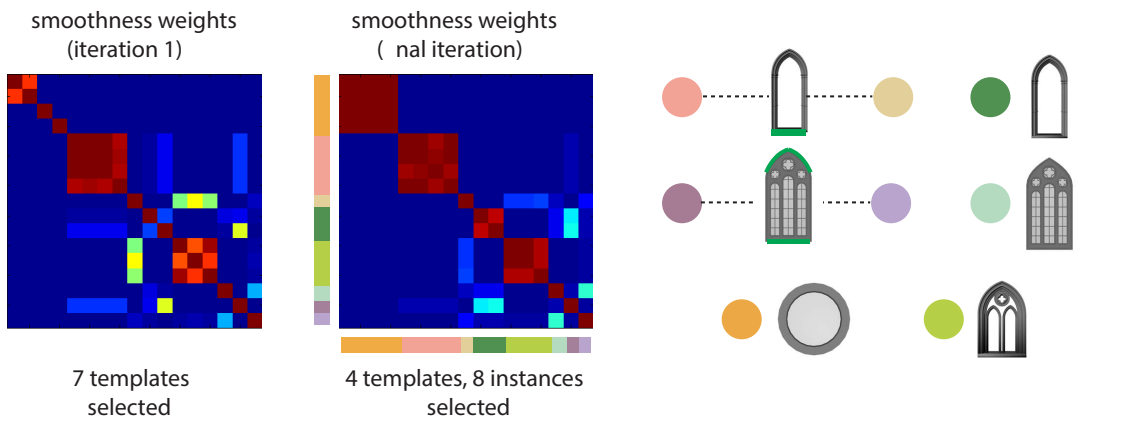


image edges



3D lines







# Bibliography

- [1] AGARWAL, S., FURUKAWA, Y., SNAVELY, N., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. Reconstructing Rome. *IEEE Computer* (2010), 40–47.
- [2] AGARWAL, S., FURUKAWA, Y., SNAVELY, N., SIMON, I., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. Building rome in a day. *Commun. ACM* 54, 10 (2011), 105–112.
- [3] ALIAGA, D., ROSEN, P., AND BEKINS, D. Style grammars for interactive visualization of architecture. *IEEE TVCG* 13, 4 (2007), 786–797.
- [4] ANANTHA, R., KRAMER, G. A., AND CRAWFORD, R. H. Assembly modelling by geometric constraint satisfaction. *Computer-Aided Design* 28, 9 (1996), 707 – 722.
- [5] BAATZ, G., KÖSER, K., CHEN, D., GRZESZCZUK, R., AND POLLEFEYS, M. Handling urban location recognition as a 2d homothetic problem. In *Proc. ECCV* (2010), pp. 266–279.
- [6] BAILLARD, C., SCHMID, C., ZISSERMAN, A., AND FITZGIBBON, A. W. Automatic line matching and 3D reconstruction of buildings from multiple views. In *ISPRS Conf. on Automatic Extraction of GIS Objects from Digital Imagery* (1999), pp. 69–80.
- [7] BAO, S. Y., CHANDRAKER, M., LIN, Y., AND SAVARESE, S. Dense object reconstruction with semantic priors. In *Proc. IEEE CVPR* (2013).
- [8] BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. PatchMatch: A randomized correspondence algorithm for structural image editing. In *ACM TOG (SIGGRAPH)* (2009), vol. 28, pp. 24:1–24:11.
- [9] BOKELOH, M., BERNER, A., WAND, M., SEIDEL, H.-P., AND SCHILLING, A. Symmetry detection using line features. In *CGF (EUROGRAPHICS)* (2009), vol. 28, pp. 697–706.
- [10] BOKELOH, M., WAND, M., AND SEIDEL, H.-P. A connection between partial symmetry and inverse procedural modeling. In *ACM TOG (SIGGRAPH)* (2010), ACM, pp. 104:1–104:10.
- [11] BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND UNO LEVY, B. *Polygon Mesh Processing*. AK Peters, 2010.

## Bibliography

---

- [12] BOYKOV, Y., VEKSLER, O., AND ZABIH, R. Fast approximate energy minimization via graph cuts. *IEEE PAMI* 23, 11 (Nov. 2001), 1222–1239.
- [13] CALÌ, J., CALIAN, D. A., AMATI, C., KLEINBERGER, R., STEED, A., KAUTZ, J., AND WEYRICH, T. 3d-printing of non-assembly, articulated models. *ACM TOG (SIGGRAPH Asia)* 31, 6 (2012), 130:1–130:8.
- [14] CARNEGIE MELLON UNIVERSITY. Cmu graphics lab motion capture database, 2003.
- [15] CEYLAN, D., LI, W., MITRA, N. J., AGRAWALA, M., AND PAULY, M. Designing and fabricating mechanical automata from mocap sequences. In *ACM TOG (SIGGRAPH Asia)* (Nov. 2013), vol. 32, ACM, pp. 186:1–186:11.
- [16] CEYLAN, D., MITRA, N. J., LI, H., WEISE, T., AND PAULY, M. Factored facade acquisition using symmetric line arrangements. In *CGF (EUROGRAPHICS)* (2012), pp. 671–680.
- [17] CEYLAN, D., MITRA, N. J., ZHENG, Y., AND PAULY, M. Coupled structure-from-motion and 3d symmetry detection for urban facades. *ACM Trans. Graph.* 33, 1 (Feb. 2014), 2:1–2:15.
- [18] CHEN, X., KANG, S. B., XU, Y.-Q., DORSEY, J., AND SHUM, H.-Y. Sketching reality: Realistic interpretation of architectural designs. *ACM TOG* 27 (2008), 11:1–11:15.
- [19] CHIOU, S.-J., AND SRIDHAR, K. Automated conceptual design of mechanisms. *Mechanism and Machine Theory* 34, 3 (1999), 467 – 495.
- [20] COHEN, A., ZACH, C., SINHA, S., AND POLLEFEYS, M. Discovering and exploiting 3d symmetries in structure from motion. In *Proc. IEEE CVPR* (2012), pp. 1514 –1521.
- [21] COROS, S., THOMASZEWSKI, B., NORIS, G., SUEDA, S., FORBERG, M., SUMNER, R. W., MATUSIK, W., AND BICKEL, B. Computational design of mechanical characters. *ACM TOG (SIGGRAPH)* 32, 4 (July 2013), 83:1–83:12.
- [22] CVX RESEARCH, I. CVX: Matlab software for disciplined convex programming, version 2.0 beta. <http://cvxr.com/cvx>, 2012.
- [23] DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proc. of SIGGRAPH* (1996).
- [24] DELONG, A., OSOKIN, A., ISACK, H., AND BOYKOV, Y. Fast approximate energy minimization with label costs. In *Proc. IEEE CVPR* (June 2010), pp. 2173–2180.
- [25] DICK, A. R., TORR, P. H. S., RUFFLE, S. J., AND CIPOLLA, R. Combining single view recognition and multiple view stereo for architectural scenes. In *Proc. IEEE ICCV* (2001).

- 
- [26] DONG, X., FROSSARD, P., VANDERGHEYNST, P., AND NEFEDOV, N. Clustering on multi-layer graphs via subspace analysis on grassmann manifolds. *IEEE Transactions on Signal Processing* 62 (2014), 905–918.
- [27] DU SAUTOY, M. *Symmetry: A Journey into the Patterns of Nature*. Harper, 2008.
- [28] FREUDENSTEIN, F. Approximate synthesis of four-bar linkages. *Resonance* 15, 8 (2010), 740–767.
- [29] FUHRMANN, S., AND GOESELE, M. Fusion of depth maps with multiple scales. In *ACM TOG (SIGGRAPH Asia)* (2011), pp. 148:1–148:8.
- [30] FURUKAWA, Y., CURLESS, B., SEITZ, S., AND SZELISKI, R. Manhattan-world stereo. In *Proc. IEEE CVPR* (2009).
- [31] FURUKAWA, Y., CURLESS, B., SEITZ, S., AND SZELISKI, R. Towards internet-scale multi-view stereo. In *Proc. IEEE CVPR* (June 2010), pp. 1434–1441.
- [32] FURUKAWA, Y., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. Reconstructing building interiors from images. In *Proc. IEEE ICCV* (2009).
- [33] FURUKAWA, Y., AND PONCE, J. Accurate, dense, and robust multiview stereopsis. *IEEE PAMI* 32 (2009), 1362–1376.
- [34] GAL, R., SORKINE, O., MITRA, N. J., AND COHEN-OR, D. iWIRES: An analyze-and-edit approach to shape manipulation. In *ACM TOG (SIGGRAPH)* (2009), vol. 28, pp. 33:1–33:10.
- [35] GIL, A., REINOSO, O., MOZOS, O., STACHNISSI, C., AND BURGARD, W. Improving data association in vision-based slam. In *Intelligent Robots and Systems* (2006), pp. 2076–2081.
- [36] GOESELE, M., CURLESS, B., AND SEITZ, S. Multi-view stereo revisited. In *Proc. IEEE CVPR* (2006), vol. 2, pp. 2402–2409.
- [37] GOESELE, M., SNAVELY, N., CURLESS, B., HOPPE, H., AND SEITZ, S. M. Multi-view stereo for community photo collections. In *Proc. IEEE ICCV* (2007).
- [38] GOLUB, G. H., AND VAN LOAN, C. F. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
- [39] GOOGLE. Atap project tango, 2014.
- [40] GOVINDU, V. Lie-algebraic averaging for globally consistent motion estimation. In *Proc. IEEE CVPR* (2004), pp. 684–691.
- [41] GOVINDU, V. M. Robustness in motion averaging. In *Proc. ACCV* (2006), pp. 457–466.
- [42] GUROBI OPTIMIZATION, I. Gurobi optimizer reference manual, 2012.



## Bibliography

---

- [43] HALLER, K., JOHN, A. L.-S., SITHARAM, M., STREINU, I., AND WHITE, N. Body-and-cad geometric constraint systems. In *Proc. Symp. on Applied Computing* (2009), ACM, pp. 1127–1131.
- [44] HAN, Y.-H., AND LEE, K. A case-based framework for reuse of previous design concepts in conceptual synthesis of mechanisms. *Computers in Industry* 57, 4 (2006), 305 – 318.
- [45] HARTLEY, R., AND ZISSERMAN, A. *Multiple View Geometry*, 2nd ed. Cambridge University Press, 2003.
- [46] JIANG, N., TAN, P., AND CHEONG, L.-F. Symmetric architecture modeling with a single image. In *ACM TOG (SIGGRAPH Asia)* (2009), pp. 113:1–113:8.
- [47] JIANG, N., TAN, P., AND CHEONG, L.-F. Multi-view repetitive structure detection. In *IEEE International Conference on Computer Vision (ICCV)* (Barcelona, Spain, 2011), pp. 535–542.
- [48] JIANG, N., TAN, P., AND CHEONG, L.-F. Seeing double without confusion: Structure-from-motion in highly ambiguous scenes. In *Proc. IEEE CVPR* (2012), pp. 1458 –1465.
- [49] KAZHDAN, M., BOLITHO, M., AND HOPPE, H. Poisson surface reconstruction. In *Symp. on Geometry Processing* (2006).
- [50] KELLY, T., AND WONKA, P. Interactive architectural modeling with procedural extrusions. *ACM TOG* 30 (2011), 14:1–14:15.
- [51] KIM, J., KIM, K., CHOI, K., AND LEE, J. Solving 3d geometric constraints for assembly modelling. *The International Journal of Advanced Manufacturing Technology* 16 (2000), 843–849.
- [52] KIM, V. G., CHAUDHURI, S., GUIBAS, L., AND FUNKHOUSER, T. Shape2pose: Human-centric shape analysis.
- [53] KIM, Y. M., MITRA, N. J., YAN, D.-M., AND GUIBAS, L. Acquiring 3d indoor environments with variability and repetition. In *ACM TOG (SIGGRAPH Asia)* (2012), pp. 138:1–138:11.
- [54] KLOPSCHITZ, M., IRSCHARA, A., REITMAYR, G., AND SCHMALSTIEG, D. Robust incremental structure from motion. In *Proc. 3DPVT* (2010).
- [55] KOLMOGOROV, V. Convergent tree-reweighted message passing for energy minimization. *IEEE PAMI* 28 (2006), 1568–1583.
- [56] KRAEVOY, V., SHEFFER, A., SHAMIR, A., AND COHEN-OR, D. Non-homogeneous resizing of complex models. pp. 111:1–111:9.
- [57] LI, H., LUO, L., VLASIC, D., PEERS, P., POPOVIĆ, J., PAULY, M., AND RUSINKIEWICZ, S. Temporally coherent completion of dynamic shapes. *ACM TOG* 31, 1 (Feb. 2012), 2:1–2:11.



- [58] LI, Y., ZHENG, Q., SHARF, A., COHEN-OR, D., CHEN, B., AND MITRA, N. J. 2d-3d fusion for layer decomposition of urban facades. In *Proc. IEEE ICCV* (Barcelona, Spain, 2011).
- [59] LIEBOWITZ, D., CRIMINISI, A., AND ZISSERMAN, A. Creating architectural models from images. In *CGF (EUROGRAPHICS)* (1999).
- [60] LIPMAN, Y., CHEN, X., DAUBECHIES, I., AND FUNKHOUSER, T. Symmetry factored embedding and distance. In *ACM TOG (SIGGRAPH)* (2010), pp. 103:1–103:12.
- [61] LIU, L., AND STAMOS, I. A systematic approach for 2d-image to 3d-range registration in urban environments. In *Proc. IEEE ICCV* (Oct 2007), pp. 1–8.
- [62] LIU, Y., HEL-OR, H., KAPLAN, C. S., AND GOOL, L. V. Computational symmetry in computer vision and computer graphics. *Foundations and Trends® in Computer Graphics and Vision* 5, 1–2 (2010), 1–195.
- [63] LOURAKIS, M. A., AND ARGYROS, A. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software* 36, 1 (2009), 1–30.
- [64] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *IJCV* 60, 2 (2004), 91–110.
- [65] LUO, L., BARAN, I., RUSINKIEWICZ, S., AND MATUSIK, W. Chopper: partitioning models into 3d-printable parts. *ACM TOG (SIGGRAPH Asia)* 31, 6 (2012), 129:1–129:9.
- [66] LUXBURG, U. A tutorial on spectral clustering. *Statistics and Computing* 17, 4 (2007), 395–416.
- [67] MARTINEC, D., AND PAJDLA, T. Robust rotation and translation estimation in multiview reconstruction. In *Proc. IEEE CVPR* (2007), pp. 1–8.
- [68] MCCARTHY, J. *Geometric Design of Linkages*. Interdisciplinary Applied Mathematics Series. Springer Verlag, 2000.
- [69] MICUSIK, B., WILDENAUER, H., AND KOSECKA, J. Detection and matching of rectilinear structures. In *Proc. IEEE CVPR* (2008), pp. 1–7.
- [70] MITRA, N. J., GUIBAS, L., AND PAULY, M. Symmetrization. In *ACM TOG (SIGGRAPH)* (2007), ACM, pp. 63:1–63:8.
- [71] MITRA, N. J., GUIBAS, L. J., AND PAULY, M. Partial and approximate symmetry detection in 3d. In *Proc. of SIGGRAPH* (2006), pp. 560–568.
- [72] MITRA, N. J., PAULY, M., WAND, M., AND CEYLAN, D. Symmetry in 3d geometry: Extraction and applications. *Computer Graphics Forum* (2013), no–no.
- [73] MITRA, N. J., YANG, Y.-L., YAN, D.-M., LI, W., AND AGRAWALA, M. Illustrating how mechanical assemblies work. *ACM TOG (SIGGRAPH)* 29, 3 (2010).

## Bibliography

---

- [74] MIČUŠÍK, B., AND KOŠECKÁ, J. Multi-view superpixel stereo in urban environments. *IJCV* 89 (2010), 106–119.
- [75] MÜLLER, P., ZENG, G., WONKA, P., AND GOOL, L. V. Image-based procedural modeling of facades. In *ACM TOG (SIGGRAPH) (2007)*, ACM.
- [76] MUSIALSKI, P., WONKA, P., ALIAGA, D. G., WIMMER, M., VAN GOOL, L., AND PURGATHOFER, W. A survey of urban reconstruction. *Computer Graphics Forum* 32, 6 (2013), 146–177.
- [77] NAN, L., SHARF, A., ZHANG, H., COHEN-OR, D., AND CHEN, B. Smartboxes for interactive urban reconstruction. In *ACM TOG (SIGGRAPH) (2010)*, pp. 93:1–93:10.
- [78] NAN, L., XIE, K., AND SHARF, A. A search-classify approach for cluttered indoor scene understanding. In *ACM TOG (SIGGRAPH Asia) (2012)*.
- [79] NG, A. Y., JORDAN, M. I., AND WEISS, Y. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (2001)*, MIT Press, pp. 849–856.
- [80] NGUYEN, A., BEN-CHEN, M., WELNICKA, K., YE, Y., AND GUIBAS, L. An optimization approach to improving collections of shape maps. In *Symp. on Geometry Processing (2011)*, Blackwell Publishing Ltd, pp. 1481–1491.
- [81] PANOZZO, D., WEBER, O., AND SORKINE, O. Robust image retargeting via axis-aligned deformation. In *CGF (EUROGRAPHICS) (2012)*, vol. 31, pp. 229—236.
- [82] PAULY, M. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. PhD thesis, Federale Institute of Technology (ETH) of Zurich, 2003.
- [83] PAULY, M., MITRA, N. J., GIESEN, J., GROSS, M., AND GUIBAS, L. J. Example-based 3d scan completion. In *Symp. on Geometry Processing (2005)*.
- [84] PAULY, M., MITRA, N. J., WALLNER, J., POTTMANN, H., AND GUIBAS, L. Discovering structural regularity in 3D geometry. In *ACM TOG (SIGGRAPH) (2008)*, pp. 43:1–43:11.
- [85] PENG, X., LEE, K., AND CHEN, L. A geometric constraint solver for 3-d assembly modeling. *The International Journal of Advanced Manufacturing Technology* 28 (2006), 561–570.
- [86] PÉREZ, P., GANGNET, M., AND BLAKE, A. Poisson image editing. *Proc. of SIGGRAPH* 22, 3 (2003), 313–318.
- [87] POLLEFEYS, M., NILSTER, D., FRAHM, J. M., AKBARZADEH, A., MORDOHAI, P., CLIPP, B., ENGELS, C., GALLUP, D., KIM, S. J., MERREL, P., SALMI, C., SINHA, S., TALTON, B., WANG, L., YANG, Q., STEWENIUS, H., YANG, R., WELCH, G., AND TOWLESS, H. Detailed real-time urban 3d reconstruction from video. *IJCV* 78 (2008), 143–167.

- 
- [88] PONOKO. Ponoko - how it works, 2014.
- [89] PU, S., AND VOSSELMAN, G. Knowledge based reconstruction of building models from terrestrial laser scanning data. *{ISPRS} Journal of Photogrammetry and Remote Sensing* 64, 6 (2009), 575 – 584.
- [90] ROBERTS, R., SINHA, S., SZELISKI, R., AND STEEDLY, D. Structure from motion for scenes with large duplicate structures. In *Proc. IEEE CVPR* (2011), pp. 3137 –3144.
- [91] ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. “GrabCut”: interactive foreground extraction using iterated graph cuts. *Proc. of SIGGRAPH 23* (Aug. 2004), 309–314.
- [92] ROTMAN, J. *An Introduction to the Theory of Groups*. 4th edition, Springer, 1994.
- [93] ROY, U., PRAMANIK, N., SUDARSAN, R., SRIRAM, R., AND LYONS, K. Function-to-form mapping: model, representation and applications in design synthesis. *Computer-Aided Design* 33, 10 (2001), 699 – 719.
- [94] SALAS-MORENO, R. F., NEWCOMBE, R. A., STRASDAT, H., KELLY, P. H., AND DAVISON, A. J. Slam++: Simultaneous localisation and mapping at the level of objects. *Proc. IEEE CVPR 0* (2013), 1352–1359.
- [95] SCHINDLER, G., KRISHNAMURTHY, P., AND DELLAERT, F. Line-based structure from motion for urban environments. In *Proc. 3DPVT* (2006).
- [96] SCHINDLER, K. A model-based method for building reconstruction. In *In Proc. of the Int. Conf. on Computer Vision Workshop on Higher-Level Knowledge in 3D Modeling and Motion* (2003), IEEE Computer Society, pp. 74–82.
- [97] SCHWARTZBURG, Y., AND PAULY, M. Fabrication-aware Design with Intersecting Planar Pieces. *CGF (EUROGRAPHICS)* 32, 2 (2013), 317–326.
- [98] SEITZ, S. M., CURLESS, B., DIEBEL, J., SCHARSTEIN, D., AND SZELISKI, R. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. IEEE CVPR* (2006).
- [99] SHAPEWAYS. Shapeways | about - how it works, 2014.
- [100] SINHA, S. N., STEEDLY, D., AND SZELISKI, R. Piecewise planar stereo for image-based rendering. In *Proc. IEEE ICCV* (29 2009-oct. 2 2009), pp. 1881 –1888.
- [101] SINHA, S. N., STEEDLY, D., SZELISKI, R., AGRAWALA, M., AND POLLEFEYS, M. Interactive 3d architectural modeling from unordered photo collections. In *ACM TOG (SIGGRAPH Asia)* (New York, NY, USA, 2008), ACM, pp. 159:1–159:10.
- [102] SNAVELY, N. Scene reconstruction and visualization from internet photo collections: A survey. *IPSJ Transactions on Computer Vision and Applications* 3 (2011), 44–66.

## Bibliography

---

- [103] SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. Photo tourism: exploring photo collections in 3d. In *Proc. of SIGGRAPH* (New York, NY, USA, 2006), ACM, pp. 835–846.
- [104] STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. Stress relief: Improving structural strength of 3d printable objects. In *ACM TOG (SIGGRAPH)* (2012), pp. 48:1–48:11.
- [105] TALTON, J. O., LOU, Y., LESSER, S., DUKE, J., MĚCH, R., AND KOLTUN, V. Metropolis procedural modeling. *ACM Trans. Graph.* 30, 2 (2011), 11:1–11:14.
- [106] TUYTELAARS, T., VAN GOOL, L., PROESMANS, M., AND MOONS, T. The cascaded Hough transform as an aid in aerial image interpretation. In *Proc. IEEE ICCV* (1998), pp. 736–739.
- [107] UMETANI, N., IGARASHI, T., AND MITRA, N. J. Guided exploration of physically valid shapes for furniture design. *ACM TOG (SIGGRAPH)* 31, 4 (2012), 86:1–86:11.
- [108] VANEGAS, C. A., ALIAGA, D. G., AND BENES, B. Automatic extraction of manhattan-world building masses from 3d laser range scans. *IEEE Transactions on Visualization and Computer Graphics* 18, 10 (Oct. 2012), 1627–1637.
- [109] VINH, N. X., EPPS, J., AND BAILEY, J. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY, USA, 2009), ICML '09, ACM, pp. 1073–1080.
- [110] WAECHTER, A., AND BIEGLER, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* 106, 1 (2006), 25–57.
- [111] WAN, G., SNAVELY, N., COHEN-OR, D., ZHENG, Q., CHEN, B., AND LI, S. Sorting unorganized photo sets for urban reconstruction. *Graph. Models* 74, 1 (2012), 14–28.
- [112] WEYL, H. *Symmetry*. Princeton University Press, 1952.
- [113] WILSON, K., AND SNAVELY, N. Network principles for sfm: Disambiguating repeated structures with local context. In *Proc. IEEE ICCV* (2013).
- [114] WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. Instant architecture. *ACM TOG* 22 (2003), 669–677.
- [115] WU, C. Towards linear-time incremental structure from motion. In *3D Vision, 2013 International Conference on* (June 2013), pp. 127–134.
- [116] WU, C., AGARWAL, S., CURLESS, B., AND SEITZ, S. Multicore bundle adjustment. In *Proc. IEEE CVPR* (June 2011), pp. 3057–3064.

- 
- [117] WU, C., AGARWAL, S., CURLESS, B., AND SEITZ, S. M. Schematic surface reconstruction. In *Proc. IEEE CVPR* (2012), pp. 1498–1505.
- [118] WU, C., FRAHM, J.-M., AND POLLEFEYS, M. Detecting large repetitive structures with salient boundaries. In *Proc. ECCV* (2010).
- [119] WU, C., FRAHM, J.-M., AND POLLEFEYS, M. Repetition-based dense single-view reconstruction. In *Proc. IEEE CVPR* (2011).
- [120] XIAO, J., FANG, T., TAN, P., ZHAO, P., OFEK, E., AND QUAN, L. Image-based façade modeling. In *ACM TOG (SIGGRAPH Asia)* (2008), ACM, pp. 161:1–161:10.
- [121] XU, W., WANG, J., YIN, K., ZHOU, K., VAN DE PANNE, M., CHEN, F., AND GUO, B. Joint-aware manipulation of deformable models. *ACM TOG (SIGGRAPH Asia)* 28, 3 (2009), 35:1–35:9.
- [122] ZACH, C., KLOPSCHITZ, M., AND POLLEFEYS, M. Disambiguating visual relations using loop constraints. In *Proc. IEEE CVPR* (2010), pp. 1426–1433.
- [123] ZHENG, Q., SHARF, A., WAN, G., LI, Y., MITRA, N. J., COHEN-OR, D., AND CHEN, B. Non-local scan consolidation for 3D urban scenes. In *ACM TOG (SIGGRAPH)* (2010), ACM, pp. 94:1–94:9.
- [124] ZHENG, Y., CHEN, X., CHENG, M.-M., ZHOU, K., HU, S.-M., AND MITRA, N. J. Interactive images: Cuboid proxies for smart image manipulation. In *ACM TOG (SIGGRAPH)* (2012), pp. 99:1–99:11.
- [125] ZHENG, Y., COHEN-OR, D., AND MITRA, N. J. Smart variations: Functional substructures for part compatibility. In *CGF (EUROGRAPHICS)* (2013), vol. 32, pp. 195–204.
- [126] ZHOU, Q.-Y., AND NEUMANN, U. Fast and extensible building modeling from airborne lidar data. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2008), GIS '08, pp. 7:1–7:8.
- [127] ZHU, L., XU, W., SNYDER, J., LIU, Y., WANG, G., AND GUO, B. Motion-guided mechanical toy modeling. *ACM TOG (SIGGRAPH Asia)* 31, 6 (2012), 127:1–127:10.



## EDUCATION

PHD, COMPUTER SCIENCE 09/2009-ONGOING  
École Polytechnique Fédérale de Lausanne, Switzerland  
School of Computer and Communication Sciences  
Advisor: Prof. Mark Pauly

MSC, COMPUTER SCIENCE 09/2007-09/2009  
Bilkent University, Turkey  
Department of Computer Engineering  
Advisor: Asst. Prof. Tolga Capin  
CPGA: 3.90/4.0

BACHELOR'S DEGREE, COMPUTER SCIENCE 09/2003-09/2007  
Middle East Technical University, Turkey  
CGPA: 3.70/4.0

HIGH SCHOOL DIPLOMA 09/1996-09/2003  
American Collegiate Institute, Turkey

## RESEARCH EXPERIENCE

ÉCOLE POLYTECHNIQUE FÉDÉRALE de LAUSANNE 09/2009-ONGOING  
Research Assistant, Computer Graphics and Geometry Laboratory

UNIVERSITY COLLEGE LONDON 08/2013, 04/2012,  
Visiting Researcher, Virtual Environments and Computer Graphics 09/2011

BILKENT UNIVERSITY 09/2007-09/2009  
Master's Student, Computer Engineering

## TEACHING EXPERIENCE

**Teaching Assitant**

École Polytechnique Fédérale de Lausanne, Switzerland  

- Digital 3D Geometry Processing (Spring 2011, Spring 2012)
- Introduction to Computer Graphics (Fall 2010, Fall 2011, Fall 2012)
- Informatics 1 (Spring 2013)

## WORK EXPERIENCE

ADOBE RESEARCH, USA 06/2012-09/2012  
Research Intern in the Creative Technology Labs

ASELSAN INC, TURKEY 02/2007-05/2009  
Software Engineer



## THESES

### **3D Mesh Animation System Targeted for Multi-touch Environments**

Master of Science, Bilkent University, 08/2009

## PUBLICATIONS

### **Detecting Structured Variations via Coupled Template Matching** (in submission)

### **SAFE: Structure-aware Facade Editing**

Minh Dang, Duygu Ceylan, Boris Neubert, Mark Pauly

Computer Graphics Forum, Proceedings of Eurographics 2014

### **Coupled Structure-from-Motion and 3D Symmetry Detection for Urban Facades**

Duygu Ceylan, Niloy J. Mitra, Youyi Zheng, Mark Pauly

ACM Transactions on Graphics (TOG) 2014

### **Designing and Fabricating Mechanical Automata from Mocap Sequences**

Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, Mark Pauly

ACM Transactions on Graphics, Proceedings of SIGGRAPH Asia 2013

### **Symmetry in 3D Geometry Extraction and Applications**

Niloy J. Mitra, Mark Pauly, Michael Wand, Duygu Ceylan

Computer Graphics Forum (CGF) 2013

### **Symmetry in 3D Geometry Extraction and Applications**

Niloy J. Mitra, Mark Pauly, Michael Wand, Duygu Ceylan

State-of-the-art Report (STAR), Eurographics 2012

### **Factored Facade Acquisition using Symmetric Line Arrangements**

Duygu Ceylan, Niloy J. Mitra, Hao Li, Thibaut Weise, Mark Pauly

Computer Graphics Forum, Proceedings of Eurographics 2012

### **A Multi-touch Interface for 3D Mesh Animation**

Duygu Ceylan, Tolga Capin

Computer Animation and Social Agents Conference 2010, CASA 2010

## INVITED TALKS

### **Digitizing and Understanding the Real World**

University of Washington, Seattle, 02/2014

Microsoft Research, Seattle, 02/2014

Adobe Research, San Jose, 03/2014

Stanford University, Stanford, 03/2014

### **Reverse Engineering for Editing**

Industrial Light & Magic, San Francisco, 09/2012

### **Factored Facade Acquisition using Symmetric Line Arrangements**

University College London, London, 04/2012

## PROFESSIONAL ACTIVITIES

- 154 **Symposium on Geometry Processing Conference Organization**  
Lausanne, Switzerland, 07/2011

## Reviewer

Siggraph 2014, Siggraph Asia 2013, Siggraph 2013, Eurographics 2012, Computer Graphics Forum 2012

## AWARDS

- Swiss National Science Foundation Early Postdoc Mobility Fellowship, 2014 (not used to join Adobe Research)
- The Google Anita Borg Memorial Scholarship finalist, 2013
- Visionair Visualization and Interaction Technologies Project Grant, 2012
- Fellowship for first year PhD studies given by EPFL, 2009
- National scholarship for master's education given by The Scientific and Technological Council of Turkey, 2007
- Graduated third from the top of Computer Engineering Department of Middle East Technical University among 171 students, 2007
- American Collegiate Institute scholarship for high school education, 1996-2003

## TECHNICAL SKILLS

### Operating Systems

Mac OS X, Linux, Windows

### Programming

IDEs Apple Xcode, Microsoft Visual Studio, Eclipse

Languages C/C++, C#, Objective C, Java

### Professional Tools

Solidworks, Autodesk Maya, Adobe Photoshop, Adobe Illustrator, MS Office Applications (Excel, Word, Powerpoint), Matlab

## LANGUAGES

Turkish (Mother Tongue), English (Fluent), Spanish (Intermediate), French (Intermediate)

## REFERENCES

### Assoc. Prof. Dr. Mark Pauly

École Polytechnique Fédérale de Lausanne, Computer Graphics and Geometry Laboratory

e-mail: mark.pauly@epfl.ch

### Assoc. Prof. Dr. Niloy J. Mitra

University College London, Virtual Environments and Computer Graphics

e-mail: n.mitra@cs.ucl.ac.uk

### Asst. Prof. Dr. Hao Li

University of Southern California

e-mail: hao@hao-li.com

### Dr. Wilmot Li

Senior Research Scientist, Adobe Research

e-mail: wilmotli@adobe.com