New Trends in 3D Video

Half-day Tutorial – EUROGRAPHICS 2007

Tutorial Organizers

Christian Theobalt
Stanford University and
Max-Planck Center for Visual Computing
theobalt@cs.stanford.edu

Stephan Würmlin
Computer Graphics Laboratory
ETH Zürich
and
CEO of LiberoVision AG
Zürich, Switzerland
wuermlin@liberovision.com

Lecturers

Edilson de Aguiar
MPI Informatik
Saarbrücken, Germany
edeaguia@mpi-if.mpg.de

Christoph Niederberger
of LiberoVision AG
Zürich, Switzerland
niederberger@liberovision.com

Table of Contents

Course Abstract

3D Video is an emerging and challenging research discipline that lives on the boundary between computer vision and computer graphics. The goal of researchers working in the field is to extract spatio-temporal models of dynamic scenes from multi-video footage in order to display them from user-controlled synthetic perspectives. 3D Video technology has the potential to lay the algorithmic foundations for a variety of intriguing new applications. This includes stunning novel visual effects for movies and computer games, as well as, facilitating the entire movie production pipeline by enabling virtual rearranging of cameras and lighting during post-processing. Furthermore, 3D Video processing will revolutionize visual media by enabling 3D TV and movies with interactive viewpoint control, or by enabling virtual fly-arounds during sports-broadcasts.

To achieve this purpose, several challenging problems from vision and graphics have to be solved simultaneously. The speakers in this course will explain the foundations of dynamic scene acquisition, dynamic scene reconstruction and dynamic scene rendering based on their own seminal work, as well as related approaches from the literature. They will explain in more detail three important categories of algorithms for dynamic shape and appearance reconstruction, namely silhouette-based, stereo-based, and model- based approaches. Alternative methods, such as data-driven approaches, will also be reviewed. The tutorial will focus on latest 3D Video techniques that were not yet covered in a tutorial, including algorithms for free-viewpoint video relighting, model-based deformable mesh tracking, as well as high-quality scene reconstruction with camera/projector setups. The course keeps a balance between the explanation of theoretical foundations, engineering problems and emerging applications of 3D Video technology. We therefore believe that the course will be a valuable and entertaining source of information for students, researchers and practitioners alike.

Syllabus

1. **Introduction (15 min)** - Speaker: Christian Theobalt

    - 3D Video - Why bother?

2. **Silhouette-based Methods (25 min)** - Speaker: Stephan Würmlin

    - Silhouette-based Methods - Foundations
    - Point Primitives for 3D Video
    - Real-time Applications in tele-presence systems (the blue-c)

3. **Stereo-based Methods (25 min)** - Speaker: Stephan Würmlin

    - Stereo-based Methods - Foundations
    - Using Camera Systems and Structured Light for High-quality 3D Video
    - Postprocessing Methods

4. **Model-based 3D Video I (25 min)** - Speaker: Christian Theobalt

    - Foundations
    - Marker-less Tracking and Dynamic Scene Reconstruction
    - Model-based 3D Video Rendering

5. **Break**

6. **Model-based 3D Video II (25 min)** - Speaker: Edilson de Aguiar

    - Alternative Model-based Approaches
    - Deformable Mesh Tracking for 3D Video

7. **Free-Viewpoint Video Relighting (25 min)** - Speaker: Christian Theobalt

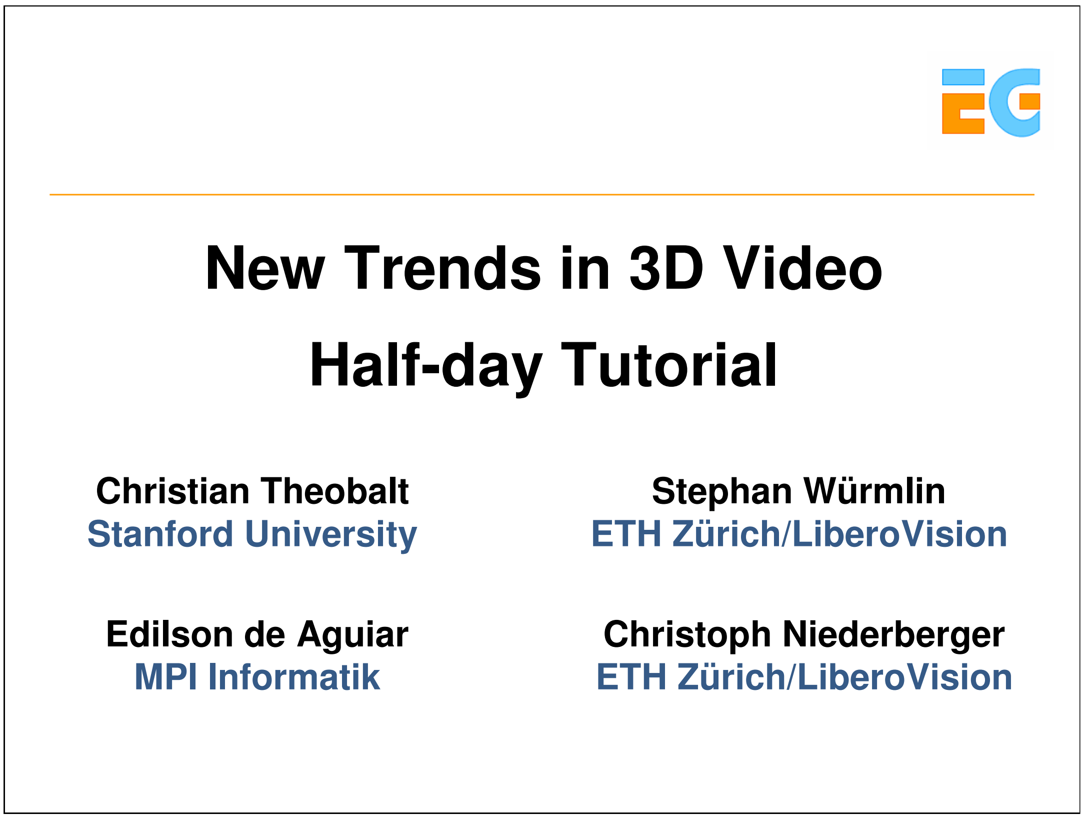
    - Data-driven Dynamic Scene Relighting
    - Model-based Free-Viewpoint Video Relighting
    -

8. **Applications (30 min)** - Speaker: Christoph Niederberger

    - Authoring and Editing 3D Video
    - Applications of 3D Video in Movie and TV Production
    -

9. **Outlook And Discussion (10 min)** - Speaker: Stephan Würmlin

- Conclusions
- Questions



New Trends in 3D Video

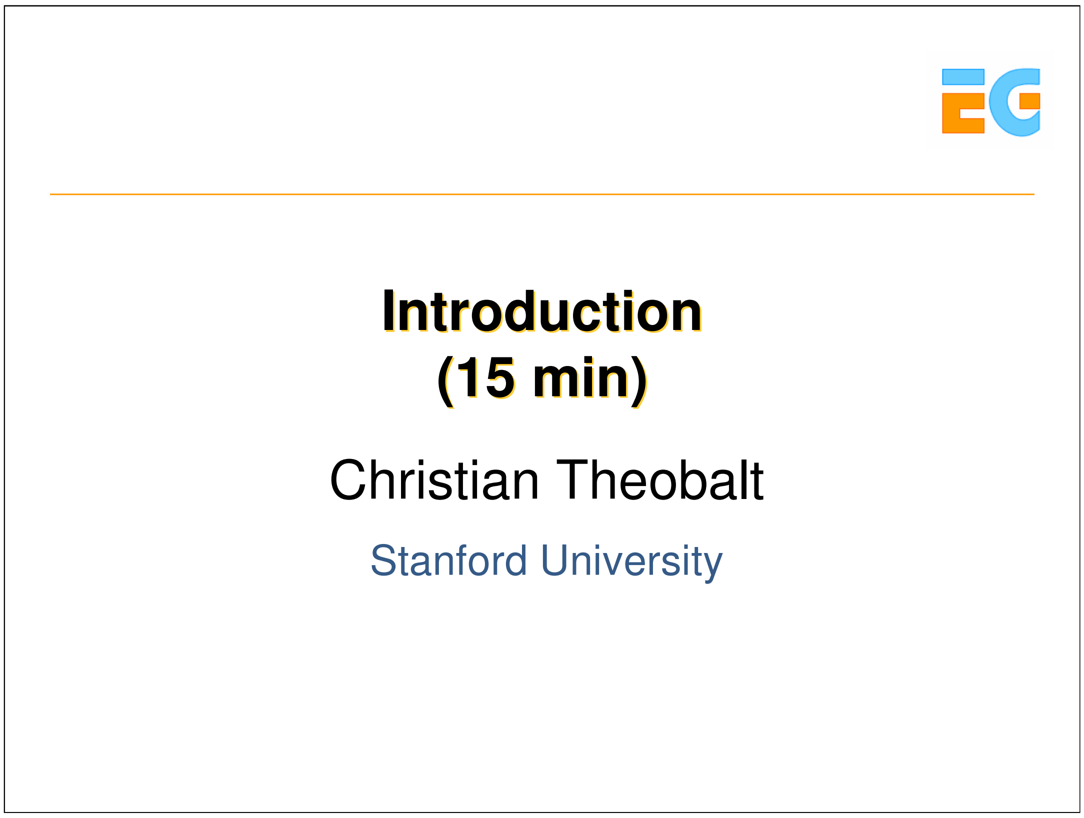
Half-day Tutorial

Christian Theobalt
Stanford University

Stephan Würmlin
ETH Zürich/LiberoVision

Edilson de Aguiar
MPI Informatik

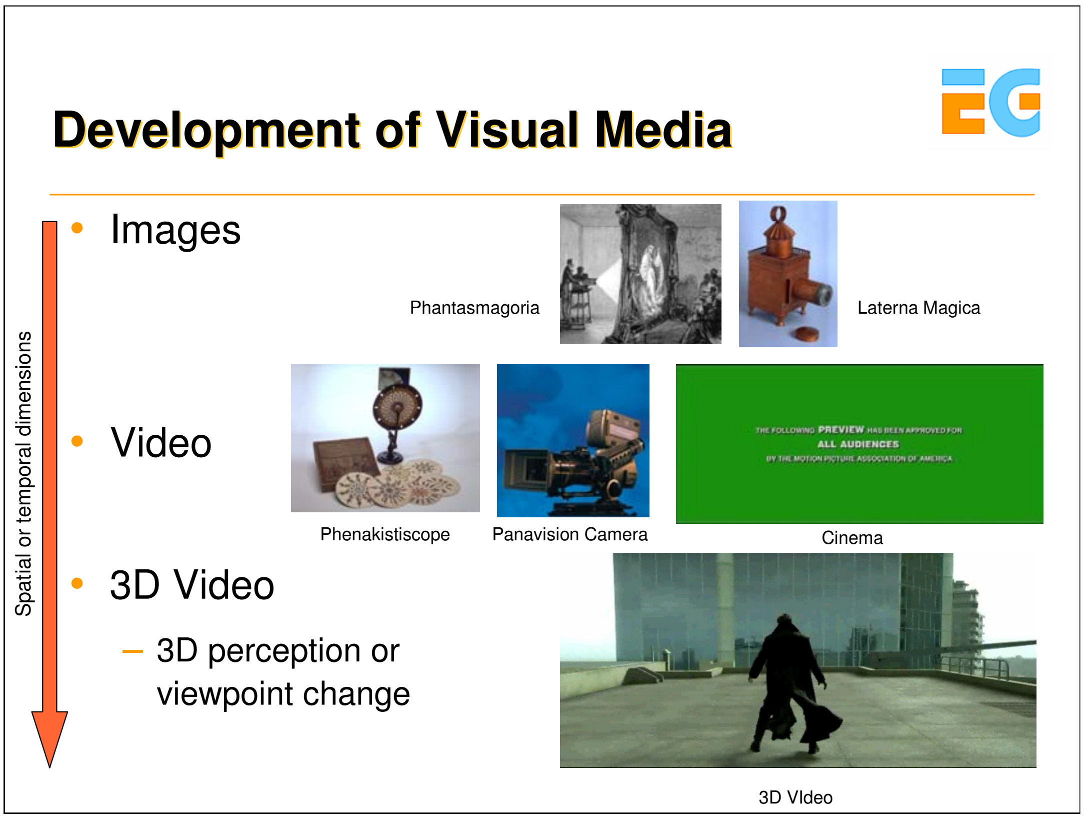
Christoph Niederberger
ETH Zürich/LiberoVision



Introduction
(15 min)

Christian Theobalt

Stanford University



Development of Visual Media

Spatial or temporal dimensions

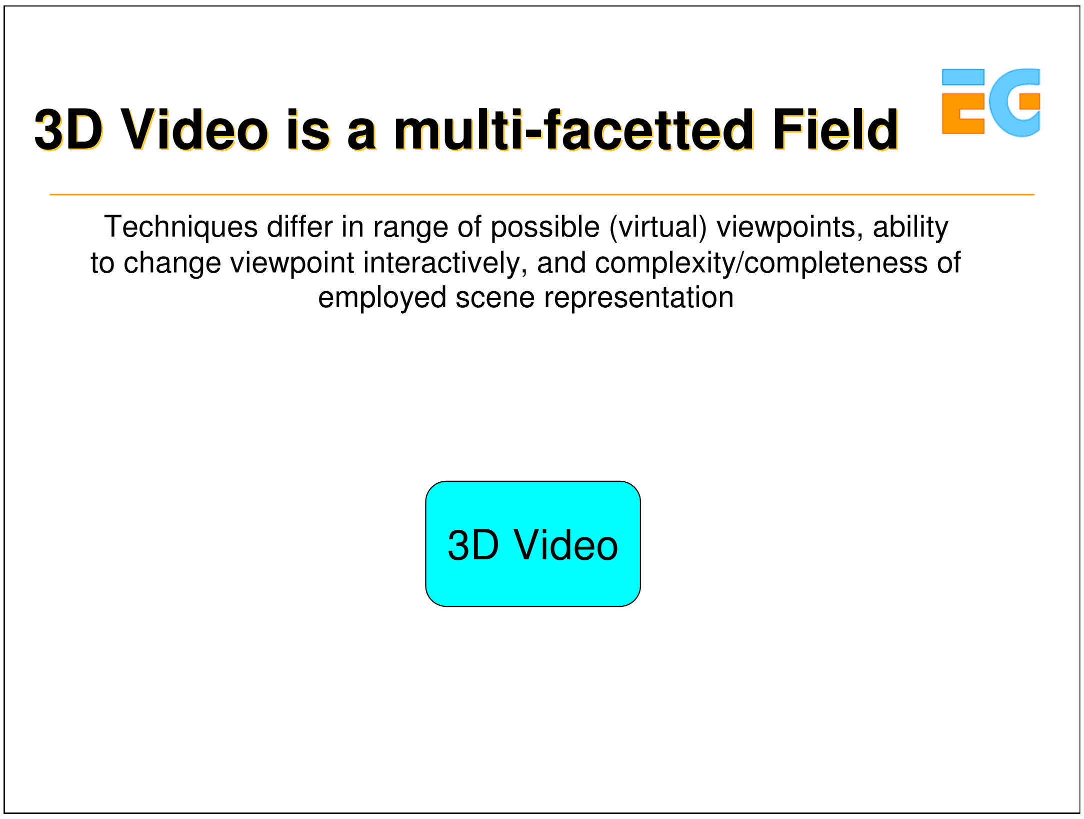
- Images
- Video
- 3D Video
  - 3D perception or viewpoint change

Phantasmagoria
Laterna Magica
Phenakistiscope
Panavision Camera
THE FOLLOWING PREVIEW HAS BEEN APPROVED FOR ALL AUDIENCES BY THE MOTION PICTURE ASSOCIATION OF AMERICA
Cinema
3D VIdeo

Vision is one of the most powerful senses that humans possess as it is one of the richest sources of psychological and physical stimuli. Visual media such as video or television capitalize on this fact and allow viewers to immerse with their imagination into scenes and events displayed to them.

In history, the ever ongoing technical improvement has caused several major changes in the way how visual media are produced and perceived. However, the most important change so far was due to the introduction of time as an additional dimension. While humans have been and are still fascinated to look at photographs, the first devices that were able to reproduce and capture moving images caused a major revolution that still dominates the type of visual media that we mostly use today, namely video (in its most general sense).

The availability of ever more powerful acquisition, computation and display hardware, has spawned a new field of research that aims at adding one more dimension to visual media, namely the third spatial dimension. This young and challenging field is still in its early days but, as we will show in this course, bears great potential to revolutionize visual media once more.



3D Video is a multi-facetted Field

Techniques differ in range of possible (virtual) viewpoints, ability to change viewpoint interactively, and complexity/completeness of employed scene representation

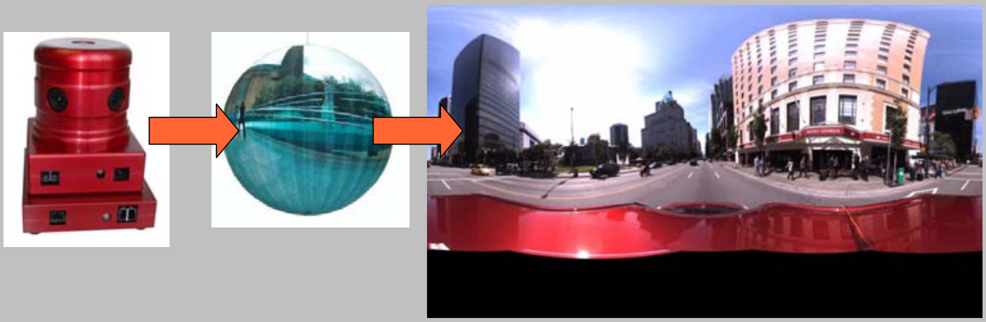
3D Video

The field of 3D video is multi-facetted as there exist several ways how the third dimension can be added. Here “3D” can, for instance, mean that the viewer is given the possibility to interact with a video and change his viewing direction on the fly while playing the content.

3D Video is a multi-facetted Field

Techniques differ in range of possible (virtual) viewpoints, ability to change viewpoint interactively, and complexity/completeness of employed scene representation

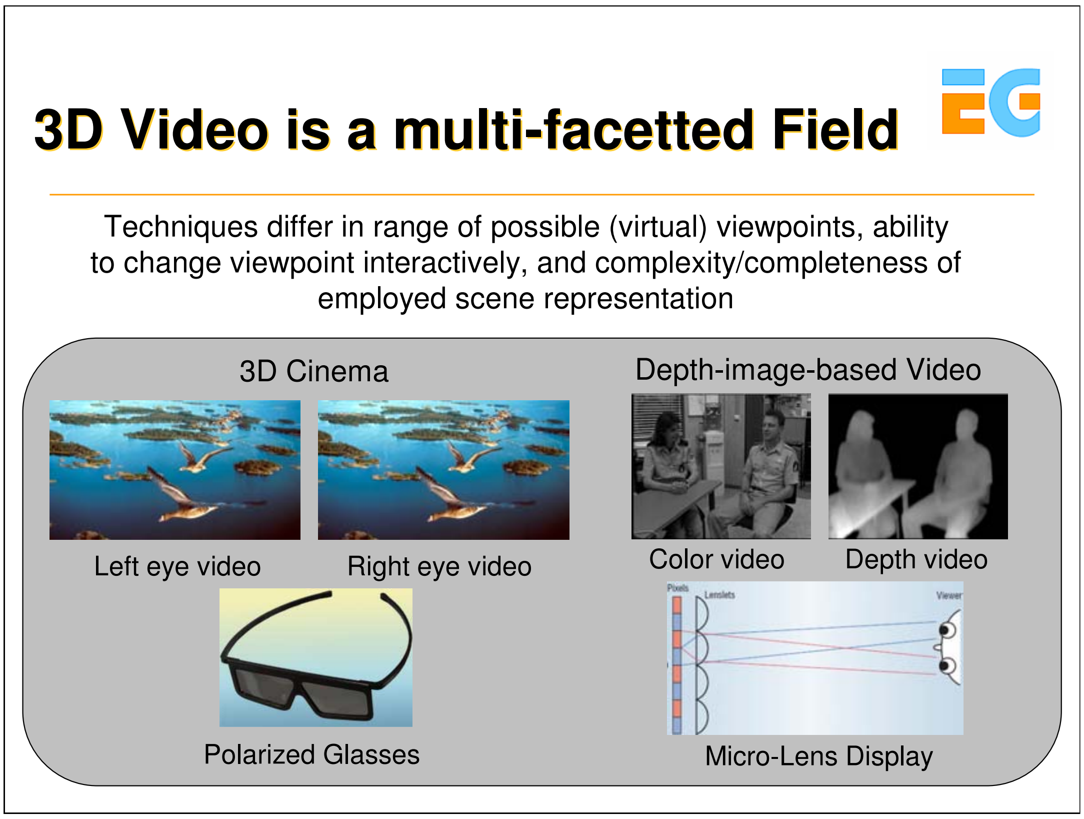
Omnidirectional Video



Free panning – predetermined viewpoints – no explicit 3D model

This type of immersive experience is, for example, generated by a technique known as omni-directional or panoramic video. Typically, this type of footage is recorded with an omni-directional camera. Such a camera either comprises of several synchronized cameras that simultaneously record all spherical directions (as the one in the image above), or of a normal camera and an attached panoramic mirror that also enables multidirectional recording. During display the captured footage is typically mapped onto a spherical or cylindrical surface such that the viewer can perform arbitrary rotations while traveling along a fixed path of camera positions.

Please refer to [1] for a detailed study of panoramic imaging techniques.



3D Video is a multi-facetted Field

Techniques differ in range of possible (virtual) viewpoints, ability to change viewpoint interactively, and complexity/completeness of employed scene representation

3D Cinema

Left eye video

Right eye video

Polarized Glasses

Depth-image-based Video

Color video

Depth video

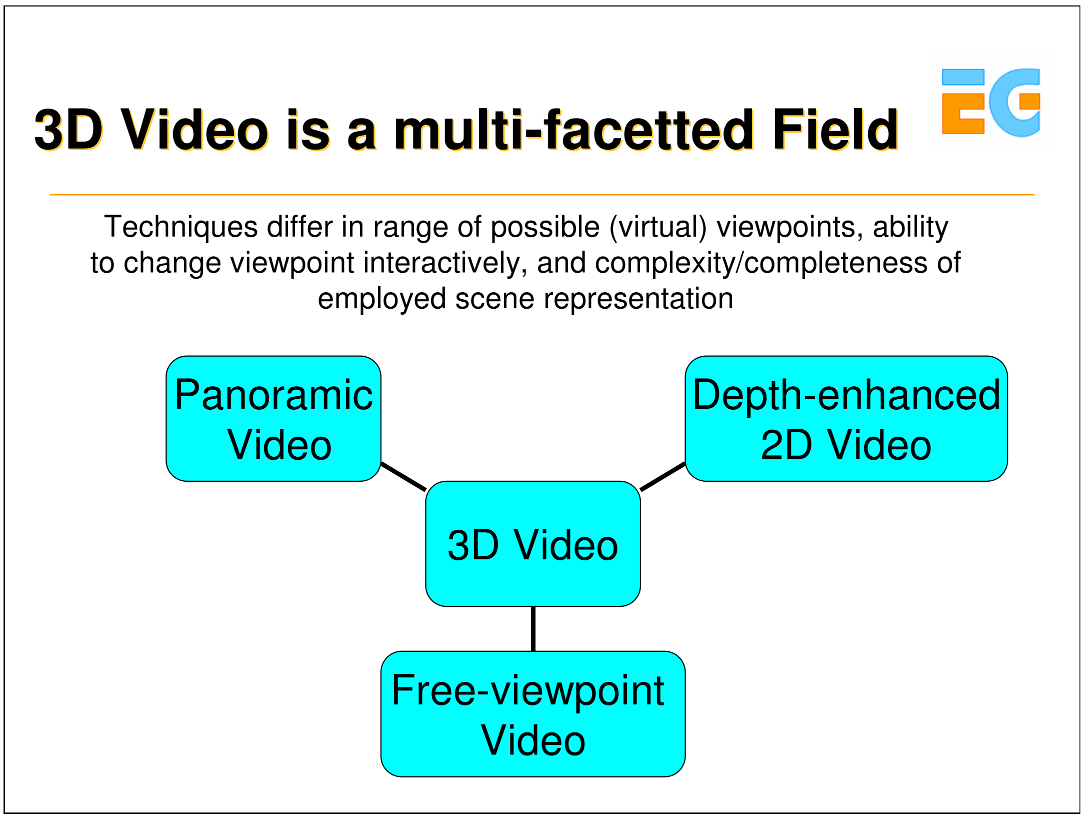
Micro-Lens Display

Pixels
Lenslets
Viewer

A different type of 3D video is provided by 3D Cinema or related depth-image based methods. Here, the main goal is to generate a true 3D depth perception while displaying video streams. However, the viewer cannot change a virtual camera viewpoint interactively, but can merely move his head in a very confined space to experience parallax effects.

While capturing a movie for 3D cinema, a stereo camera records independent video streams for the left and the right eye. During display, both streams are simultaneously rendered. Typically, some kind of stereo splitter technology is used to separate the left and the right signals from the displayed footage. A common method is to use two projectors with different polarizations and a pair of glasses with appropriate polarization filters for each eye.

Depth-image-based rendering [2] uses hybrid video streams comprising of a color stream and a synchronized depth map stream. During display, virtual images for the left and the right eye can be reconstructed on-the-fly thereby creating a similar depth-enhanced viewing experience as 3D cinema, for instance on an auto-stereoscopic micro-lens display.



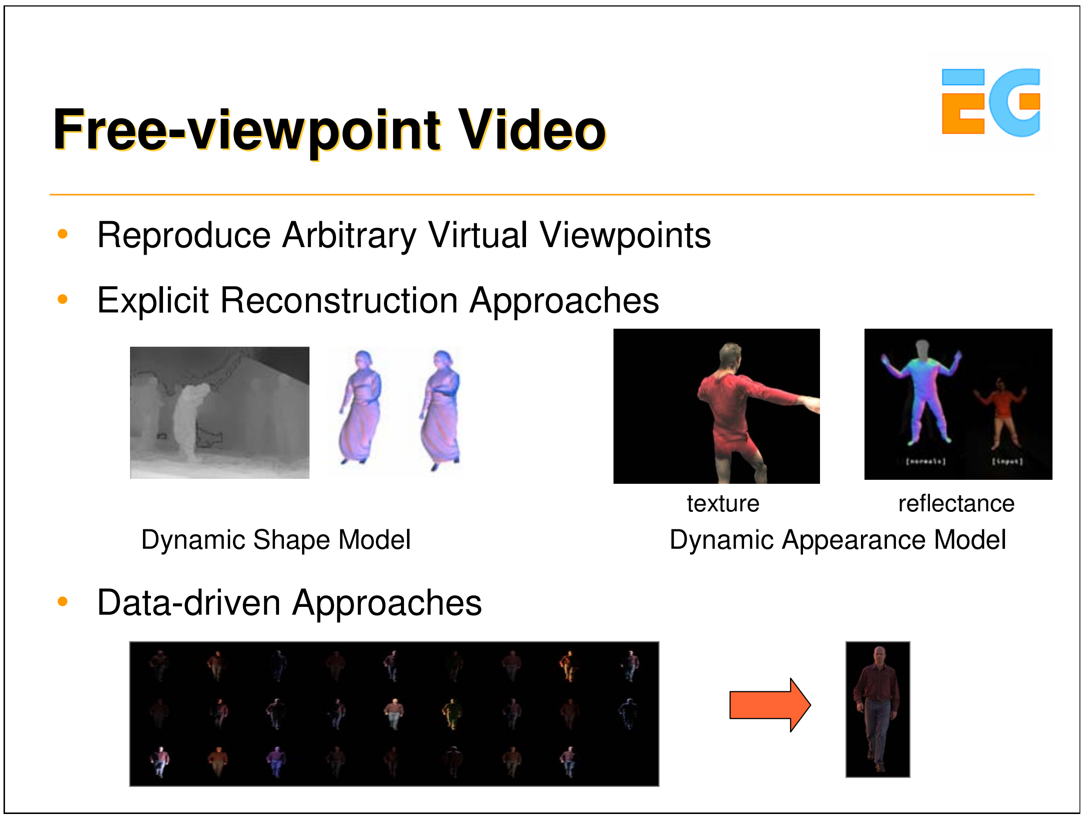
3D Video is a multi-facetted Field

Techniques differ in range of possible (virtual) viewpoints, ability to change viewpoint interactively, and complexity/completeness of employed scene representation

Panoramic Video
Depth-enhanced 2D Video
3D Video
Free-viewpoint Video

The previous two categories were mainly representative examples. A sea of other techniques exists that combines ideas from the two to perform, for instance, panoramic stereo to name just one example.

The techniques we will talk about in this course reconstruct and render the most general type of 3D videos, so-called free-viewpoint videos. This type of dynamic scene representation enables the display of captured real-world footage from arbitrary novel viewing positions and directions. As such, a free-viewpoint video representation is the most general type of 3D video as all other types of 3D video that we talked about before can be derived from it.



Free-viewpoint Video

- Reproduce Arbitrary Virtual Viewpoints
- Explicit Reconstruction Approaches

Dynamic Shape Model

texture reflectance

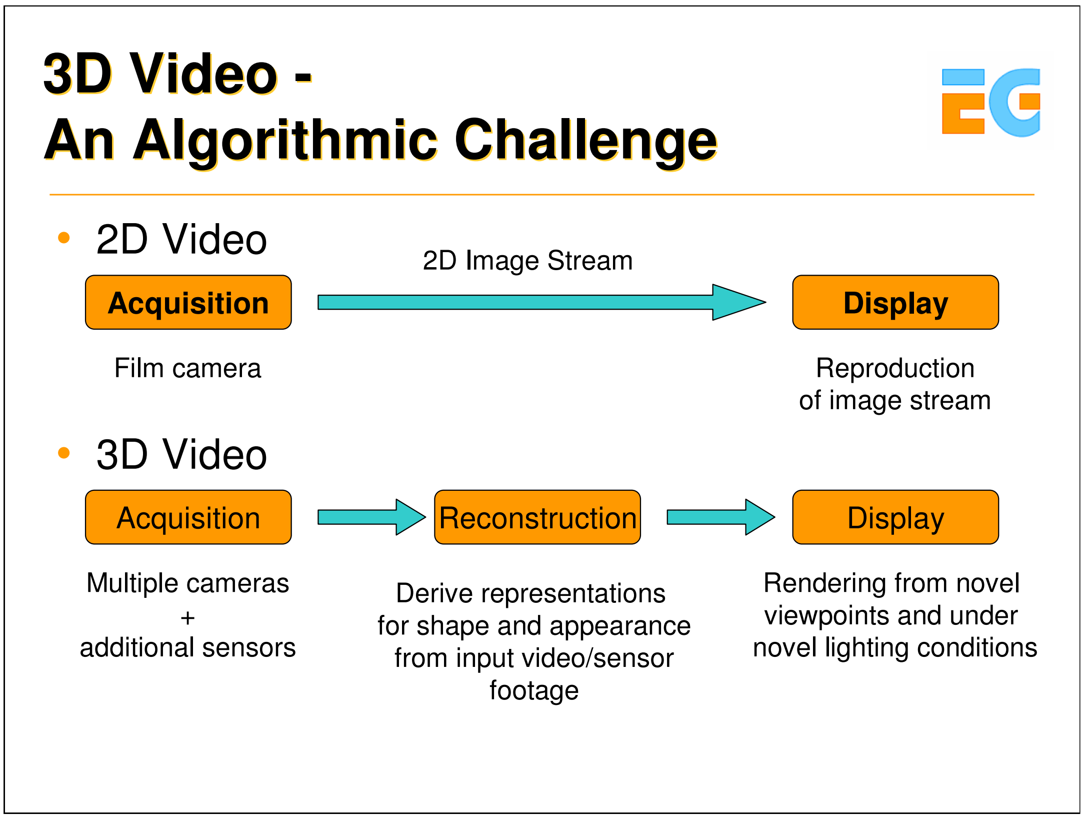
Dynamic Appearance Model

- Data-driven Approaches

Most 3D video approaches capture a full dynamic 3D representation of real-world scenes that comprises, for instance, of a dynamic shape model as well as a dynamic appearance model. As we will see later in the course, the dynamic shape models are typically dynamic 3D meshes or point primitive presentations. Multi-view appearance is normally represented by a set of multi-view textures. Recently, even dynamic surface reflectance could be reconstructed which we will also show in this course.

Instead of representing scene geometry and appearance explicitly, data-driven approaches sample the space of capturing viewpoints densely and reconstruct novel views by appropriately combining the captured raw image data. In the remainder of this course, we will see examples for either of these category of approaches.

Some images on this slide were kindly provided by Larry Zitnick from Microsoft Research and Paul Debevec from the University of Southern California.



3D Video - An Algorithmic Challenge

- 2D Video

Acquisition → 2D Image Stream → Display

Film camera — Reproduction of image stream

- 3D Video

Acquisition → Reconstruction → Display

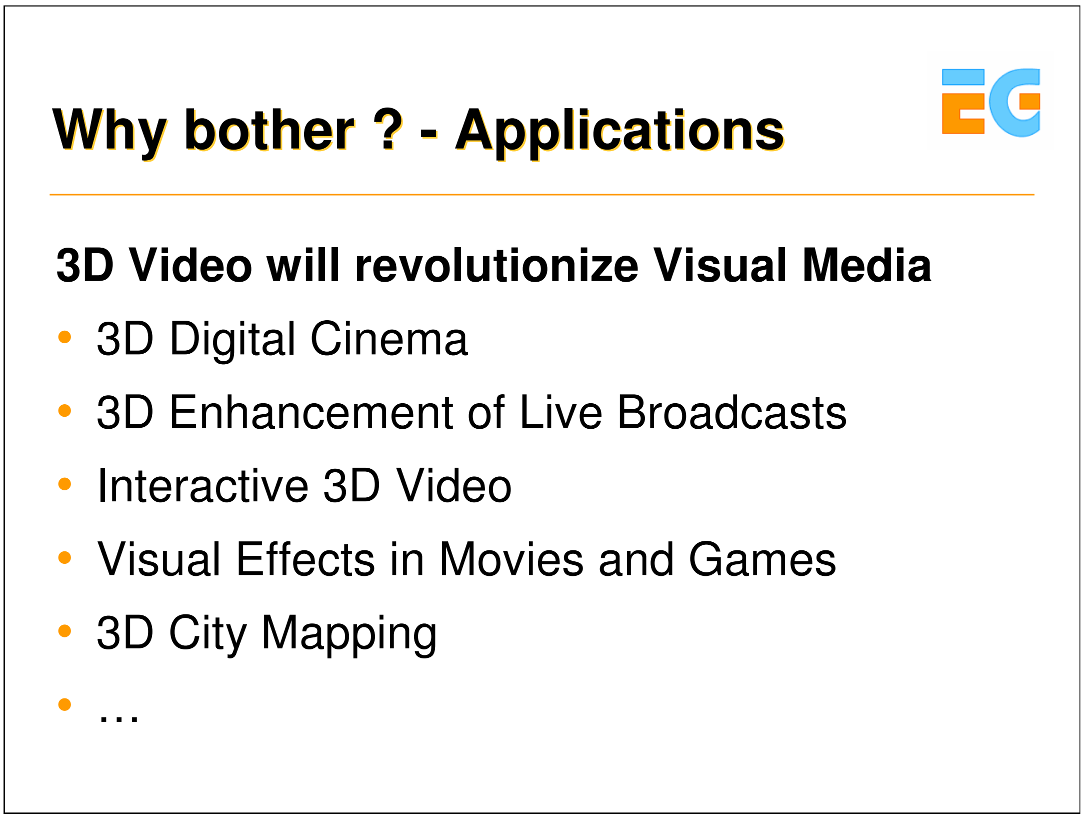
Multiple cameras + additional sensors — Derive representations for shape and appearance from input video/sensor footage — Rendering from novel viewpoints and under novel lighting conditions

The generation of 3D Video requires the solution to hard algorithmic problems that live on the boundary between the fields of Computer Vision and Computer Graphics.

The standard 2D video production pipeline shown above is fairly well understood an comprises of an acquisition and a display step. Acquisition is performed using standard camera systems and display, for the most part, is a replay of the captured streams on a display device.

The production of 3D video requires a fundamental rethinking of this pipeline. While still there is an acquisition and a display step involved, they have to be redesigned from scratch in terms of both the required engineering (sensors etc.) and the employed algorithms. Additionally, there is a reconstruction step involved which infers from the captured footage the underlying dynamic scene descriptions. It is this step which poses the hardest challenges as it requires the proper solution to several computer vision problems known to be notoriously hard.

In the remainder of this course, we will explain in more detail several possible solutions to each of the three steps.



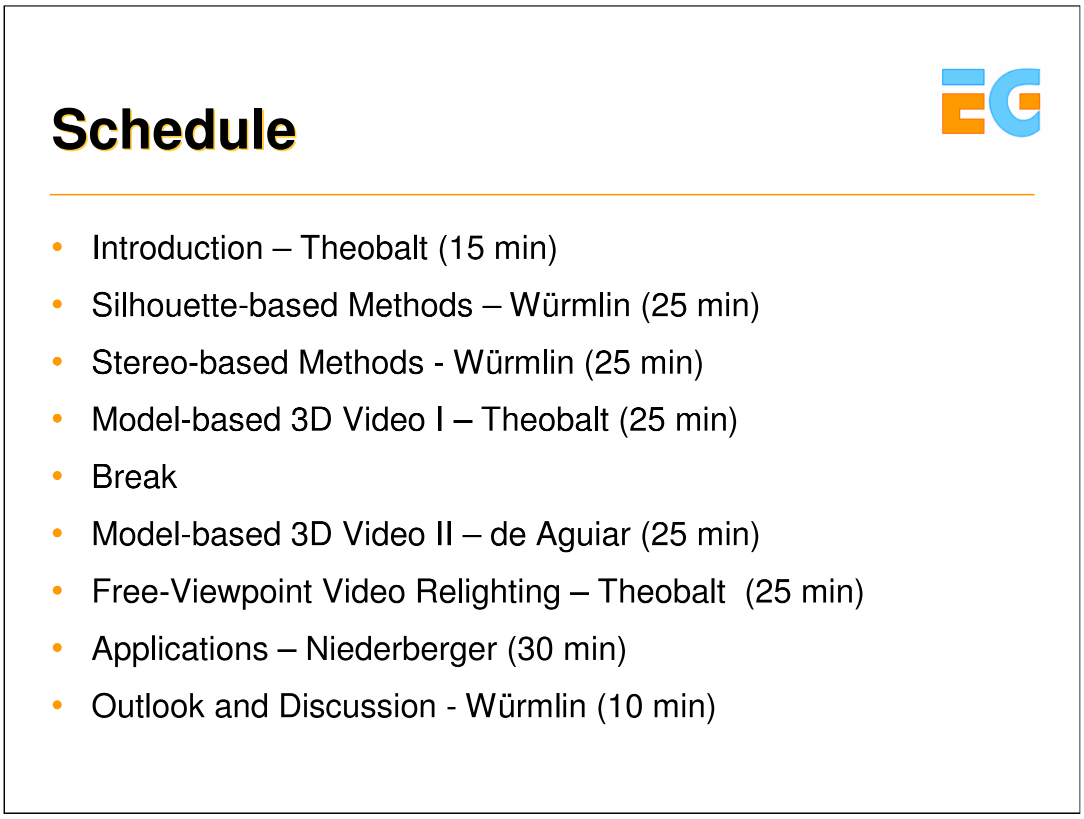
Why bother ? - Applications

3D Video will revolutionize Visual Media

- 3D Digital Cinema
- 3D Enhancement of Live Broadcasts
- Interactive 3D Video
- Visual Effects in Movies and Games
- 3D City Mapping
- …

Apart from the fact that 3D Video raises challenging algorithmic problems, the authors of this course believe that the technology has the potential to revolutionize the way How visual media are produced and presented.

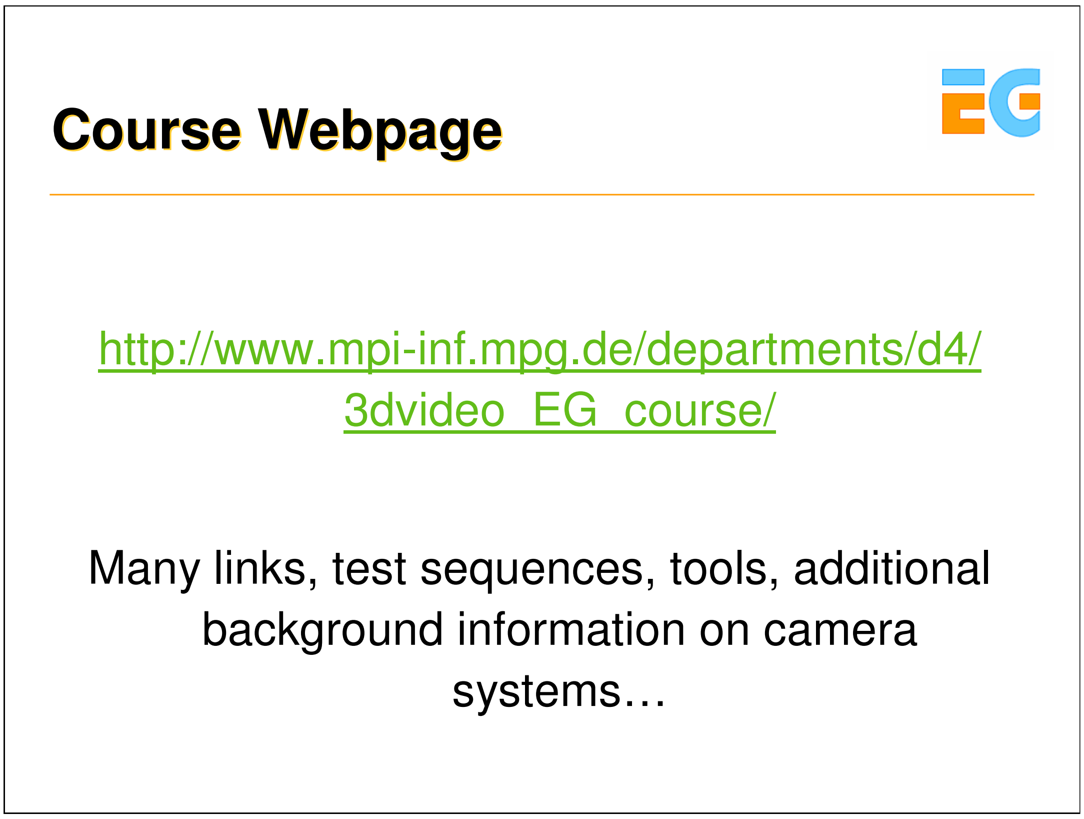
There is a variety of intriguing applications of 3D video technology in movie, TV and game productions that are currently developed. The list above just names of few of them. In the remainder of the course, we will have a closer look at some of these applications.



Schedule

- Introduction – Theobalt (15 min)
- Silhouette-based Methods – Würmlin (25 min)
- Stereo-based Methods - Würmlin (25 min)
- Model-based 3D Video I – Theobalt (25 min)
- Break
- Model-based 3D Video II – de Aguiar (25 min)
- Free-Viewpoint Video Relighting – Theobalt (25 min)
- Applications – Niederberger (30 min)
- Outlook and Discussion - Würmlin (10 min)

This slide illustrates the further schedule of the course. Please also refer to the beginning of the course notes for a more detailed schedule.



Course Webpage

http://www.mpi-inf.mpg.de/departments/d4/3dvideo_EG_course/

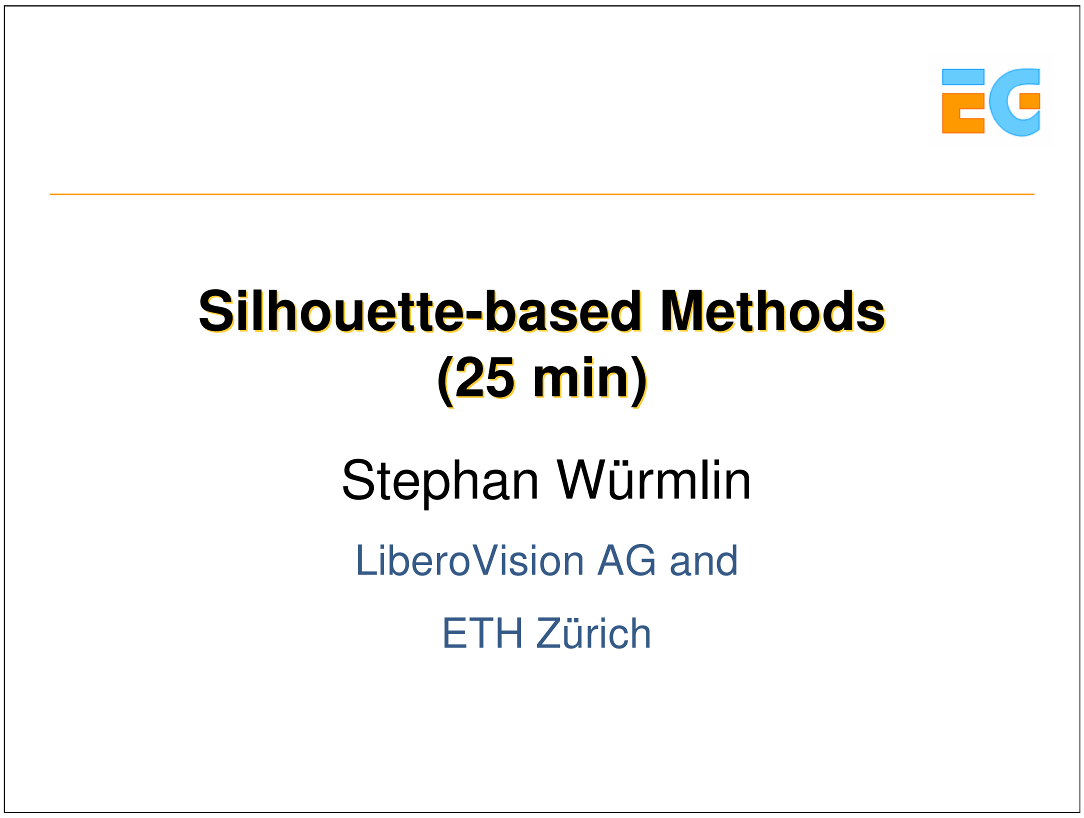
Many links, test sequences, tools, additional background information on camera systems…

The webpage accompanying this course lists some interesting links and also some test data set that the people who attended the course may want to use in their own work. The web page also features a detailed list of references to related work from the literature.

References

•[1] O. Faugeras, R. Benosman, S. B. Kang, *Panoramic Vision*, Springer, 2001.

•[2] C. Fehn. 3D-TV Using Depth-Image-Based Rendering (DIBR). In *Proceedings of Picture Coding Symposium*, San Francisco, CA, USA, December 2004.

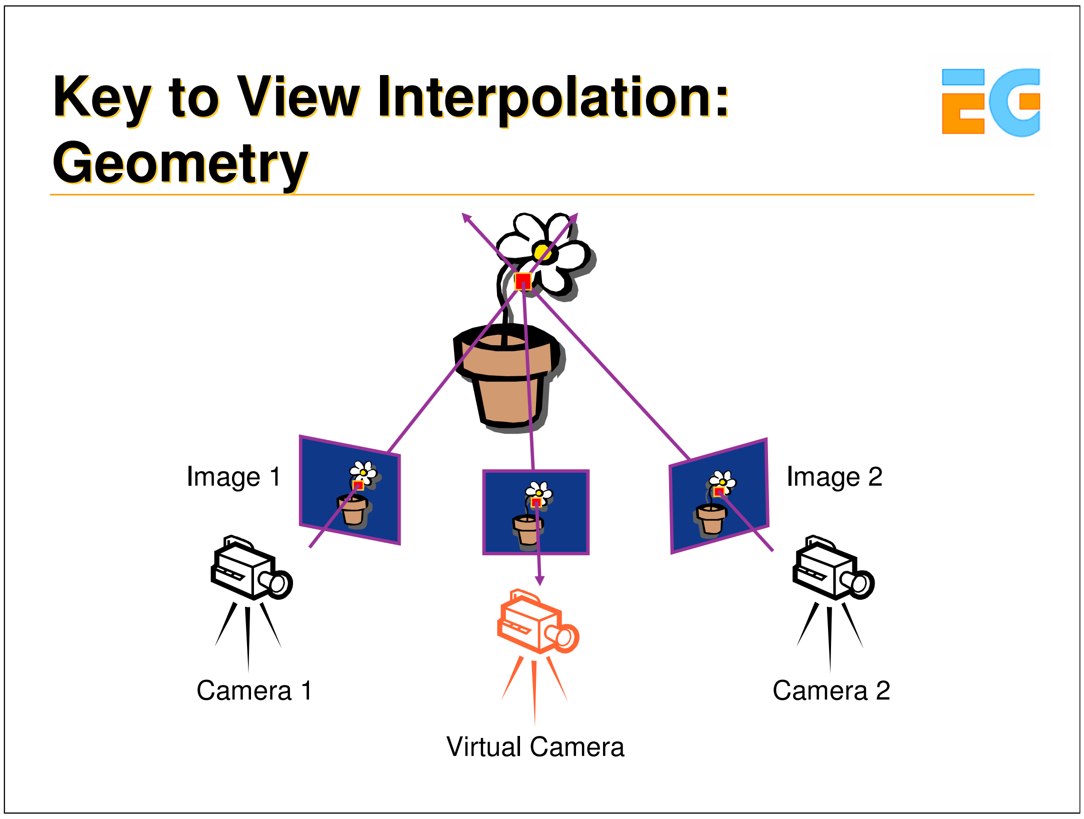


Silhouette-based Methods (25 min)

Stephan Würmlin

LiberoVision AG and

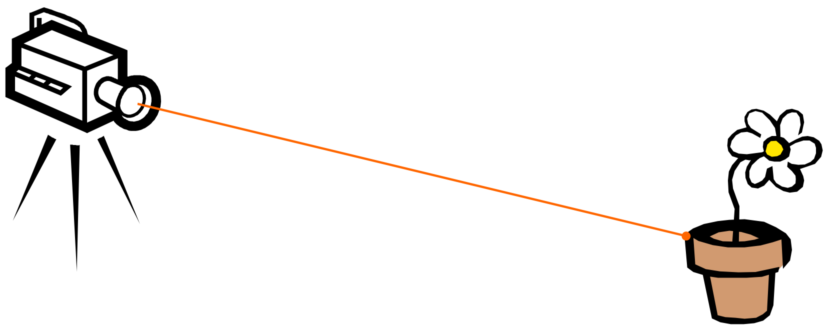
ETH Zürich



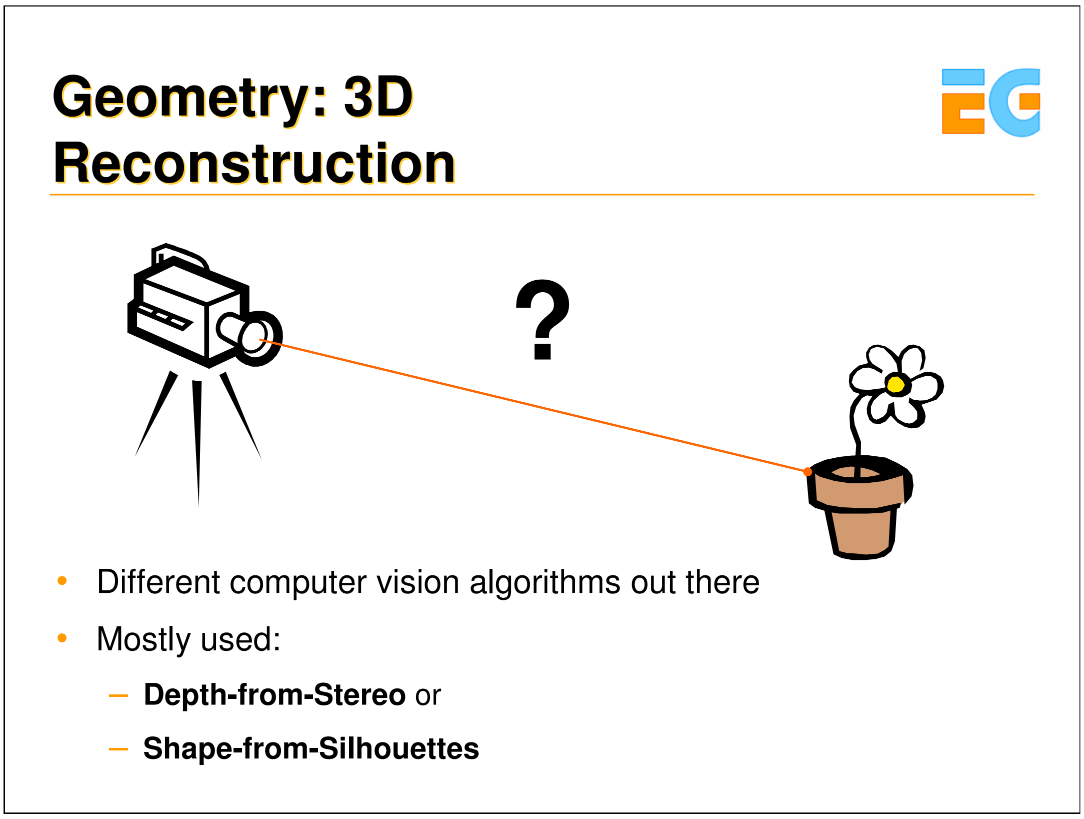
Key to View Interpolation: Geometry

3D video is mainly about how to generate or interpolate arbitrary views from a set of multiple camera images. There are many different methods that can achieve that, however, purely image-based approaches such as the Lightfield or the Lumigraph need a huge amount of different input images to smoothly interpolate novel views. Most researchers intend to design more practical systems, and for them it is key to include some sort of geometry or 3D information in the data.

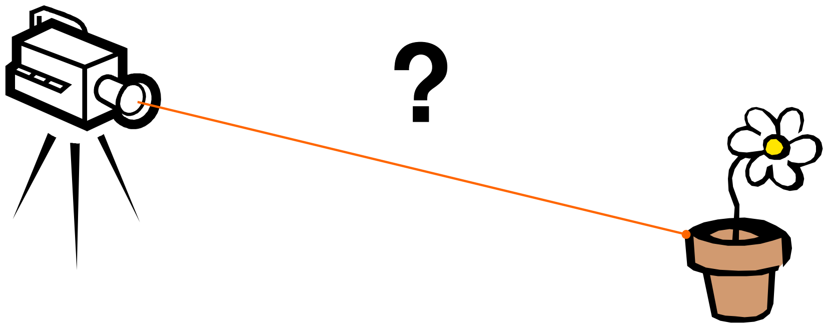
Image Acquisition



From images acquired by cameras…



Geometry: 3D Reconstruction



- Different computer vision algorithms out there
- Mostly used:
  - **Depth-from-Stereo** or
  - **Shape-from-Silhouettes**

… we want to know where each 3D scene point is that is image by the camera. In other words we want to compute the distance from the camera (or more precise the image plane of the camera) to the scene point.

There are basically two classes of algorithms that can compute this information from the images alone: (1) depth-from-stereo and (2) shape-from-silhouettes.

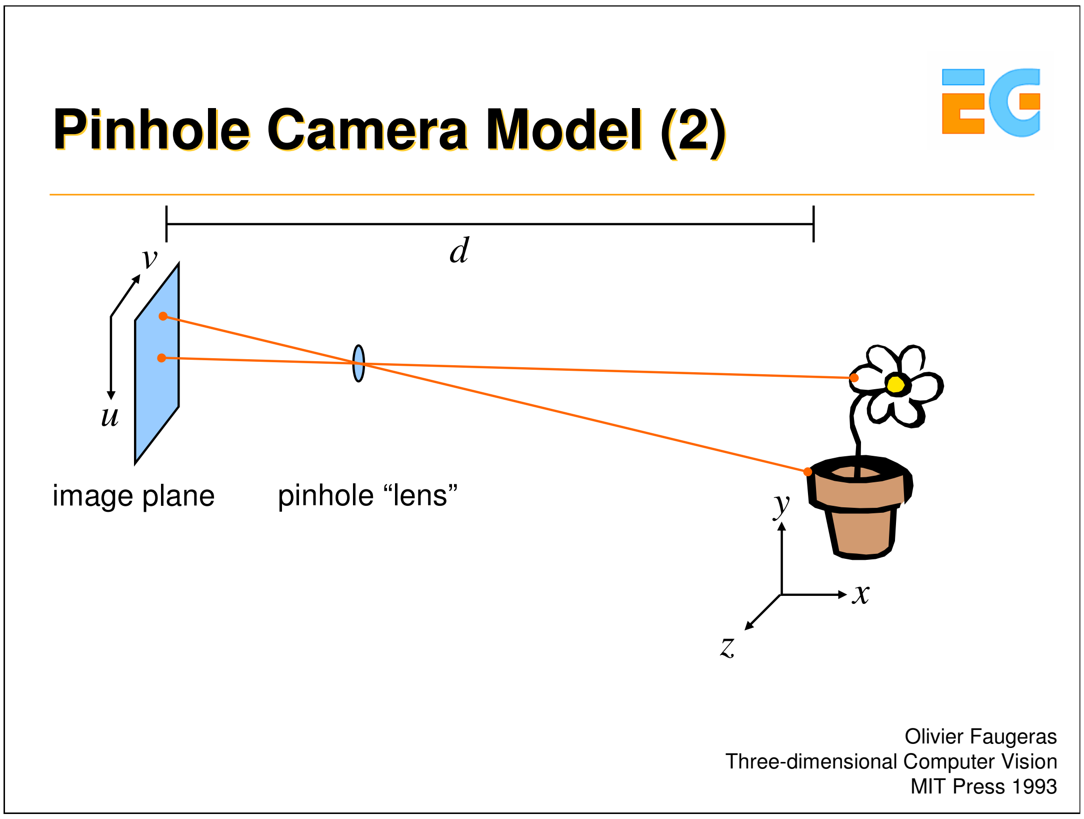
We will explain the fundamentals of both classes of algorithms and show some example methods and systems for 3D video.

Pinhole Camera Model

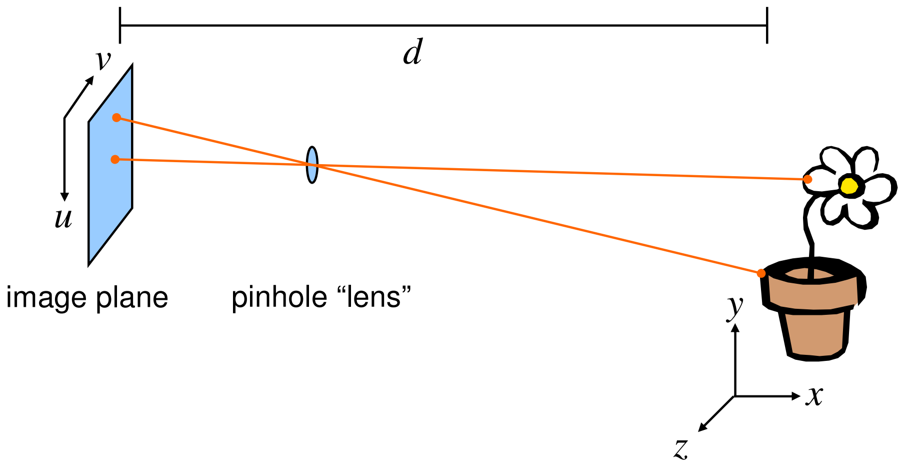
Albrecht Dürer, Man Drawing a Lute (The Draughtsman of the Lute), woodcut, 1525

Before we can do that we need to know how we model the camera. The most used model is the ideal pinhole camera model which is a sufficiently close approximation of a real camera. The geometric process for image formation in a pinhole camera has been nicely illustrated by Dürer. The process is completely determined by choosing a perspective projection center and a retinal plane. The projection of a scene point is then obtained as the intersection of a line passing through this point and the center of projection C with the retinal plane P.

Most cameras are described relatively well by this model. In some cases additional effects (e.g. radial distortion) have to be taken into account



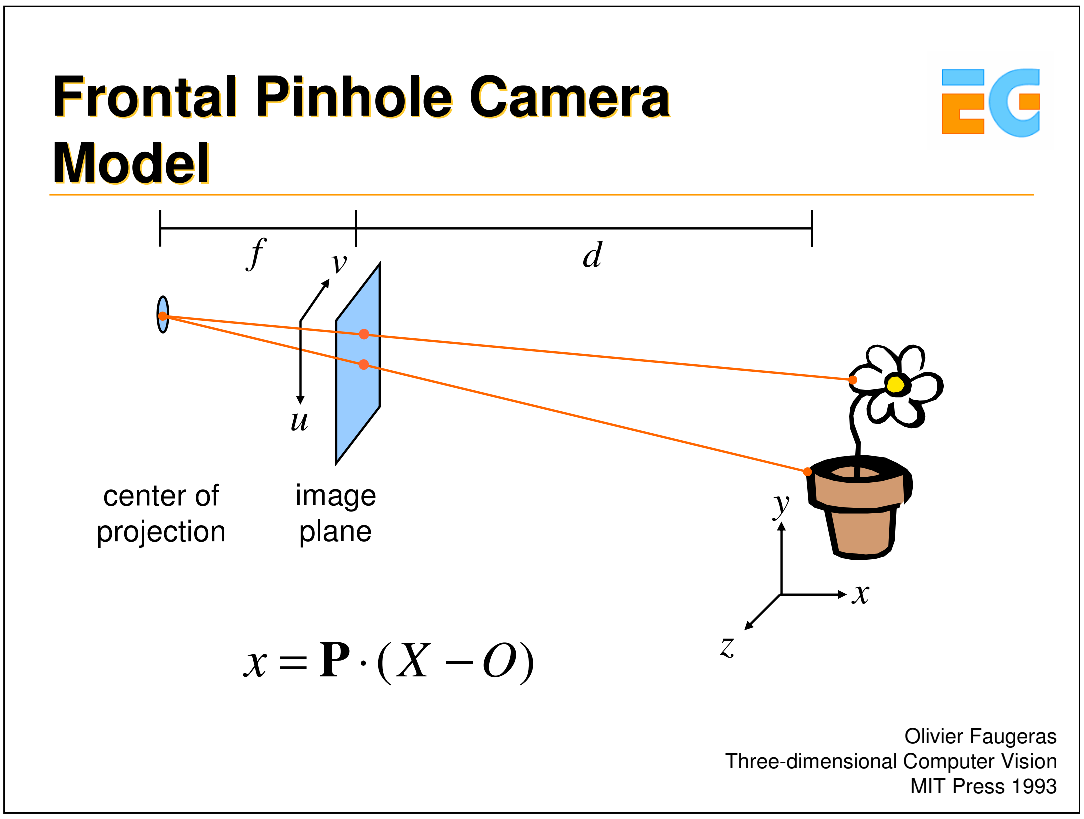
Pinhole Camera Model (2)



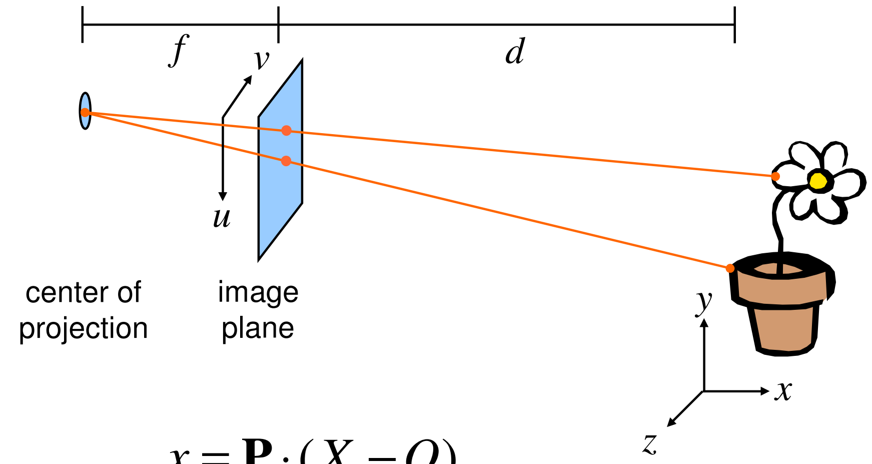
Olivier Faugeras
Three-dimensional Computer Vision
MIT Press 1993

Here is a more schematic overview.

There is a perspective transformation that transforms points in 3-space X, Y, Z to image plane pixels u, v.



Frontal Pinhole Camera Model



$$x = \mathbf{P} \cdot (X - O)$$

Olivier Faugeras
Three-dimensional Computer Vision
MIT Press 1993

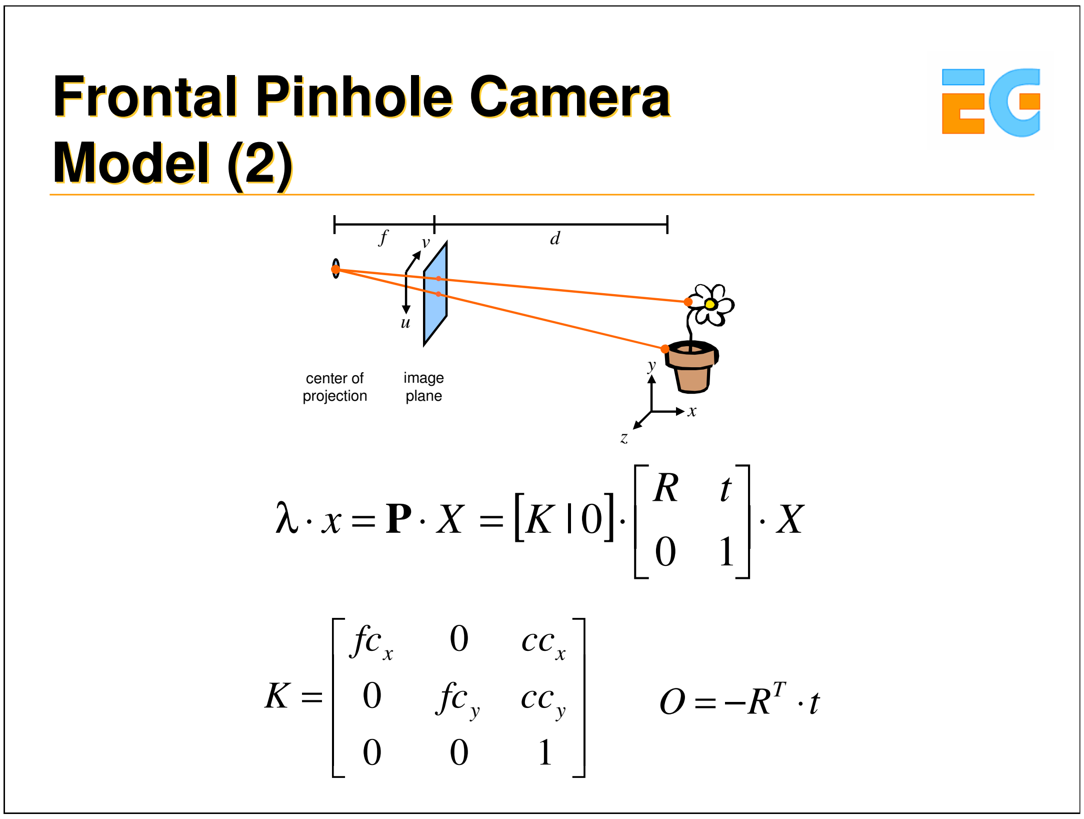
The frontal pinhole camera model is more easy to understand if one thinks about this. There all viewing rays converge in the pinhole which is now called the center of projection.

The projection of a camera (transforming 3D points into the camera's image plane) is defined by this equation.

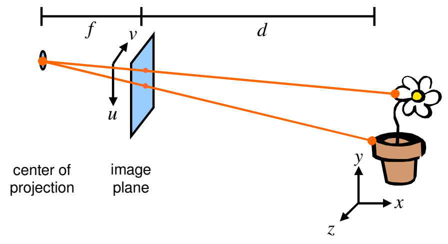
Where:

P is the matrix projecting viewing rays to image coordinates. The inverse of P would be the matrix transforming image coordinates to rays in 3D world space.

O represents the center of projection of the pinhole camera.



Frontal Pinhole Camera Model (2)



$$\lambda \cdot x = \mathbf{P} \cdot X = [K \mid 0] \cdot \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \cdot X$$

$$K = \begin{bmatrix} fc_x & 0 & cc_x \\ 0 & fc_y & cc_y \\ 0 & 0 & 1 \end{bmatrix} \qquad O = -R^T \cdot t$$

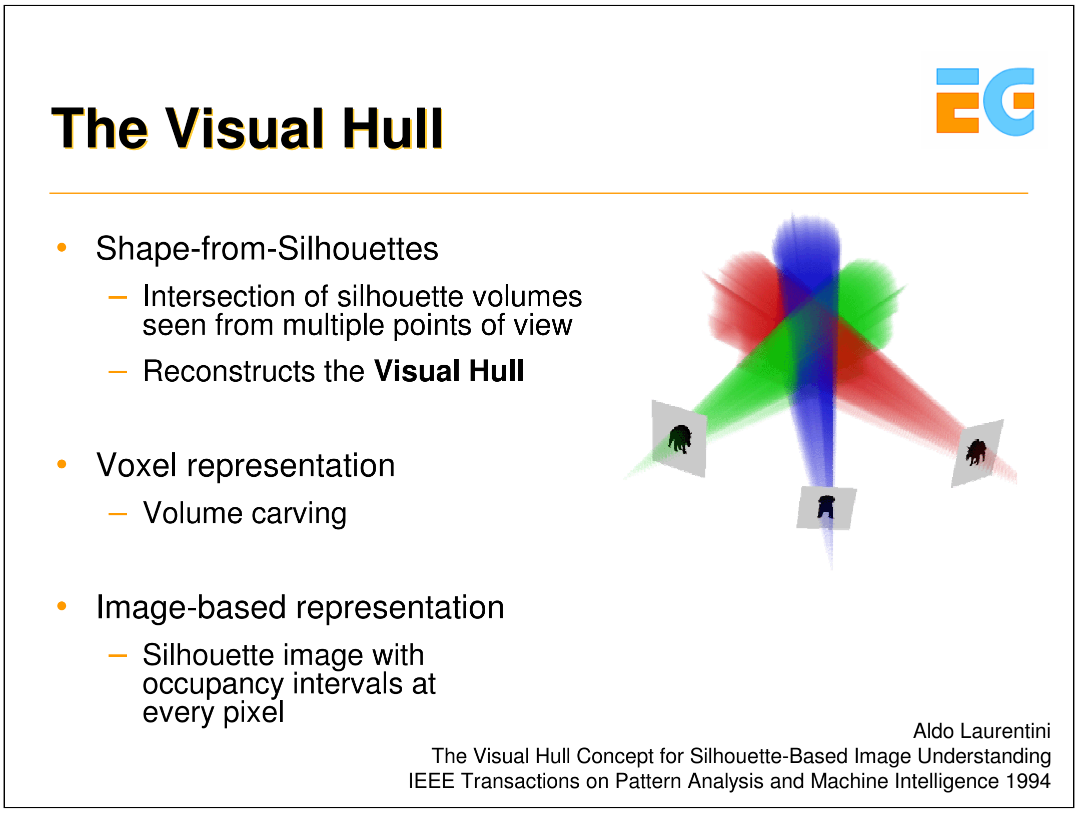
The mapping between a point in 3D space and the corresponding camera pixel can also be rewritten as…

Where:

K is an upper is an upper triangular 3x3 matrix containing the camera intrinsic parameters and **R** and t denote the rotation and translation between a world coordinate system W and the camera coordinate system C.

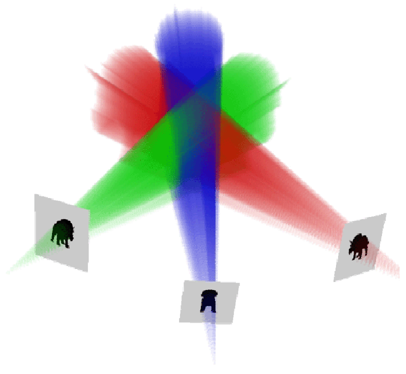
fcx and fcy are the focal lengths in effective horizontal and vertical pixel size units, and [ccx, ccy]^T represents the image center coordinates, i.e. the principal point.

The center of projection can easily be determined by…



The Visual Hull

- Shape-from-Silhouettes
  - Intersection of silhouette volumes seen from multiple points of view
  - Reconstructs the **Visual Hull**
- Voxel representation
  - Volume carving
- Image-based representation
  - Silhouette image with occupancy intervals at every pixel

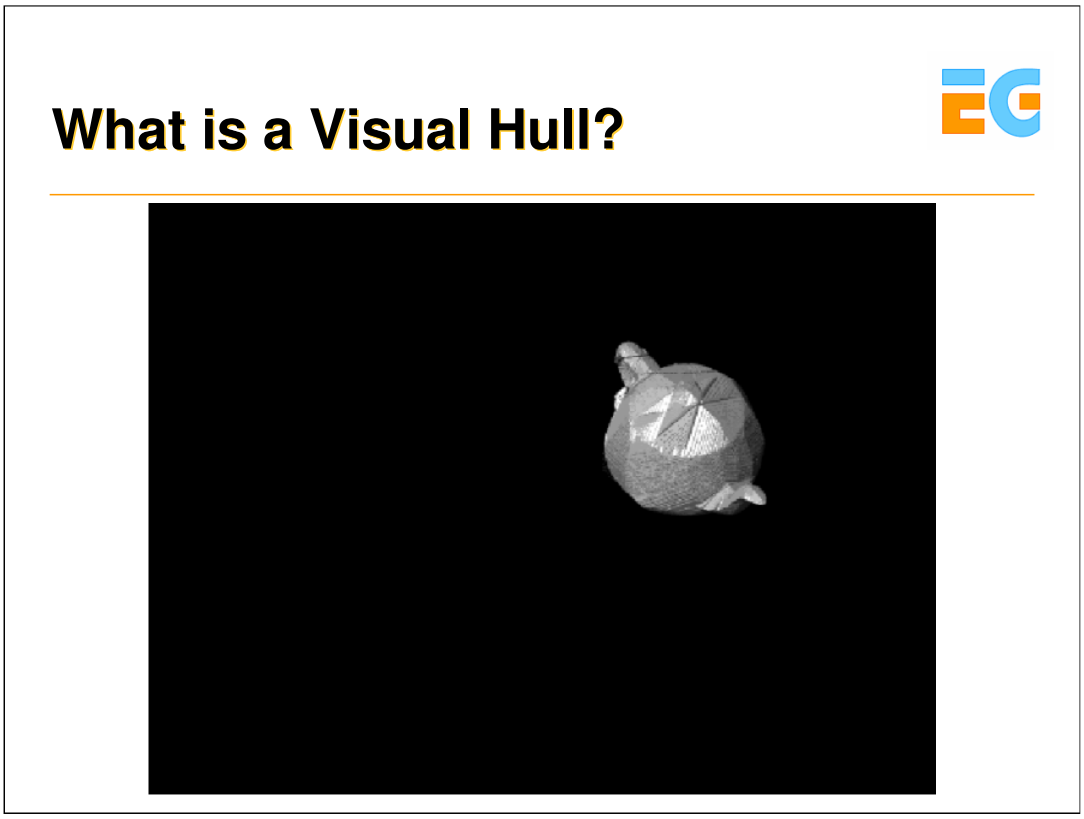


Aldo Laurentini
The Visual Hull Concept for Silhouette-Based Image Understanding
IEEE Transactions on Pattern Analysis and Machine Intelligence 1994

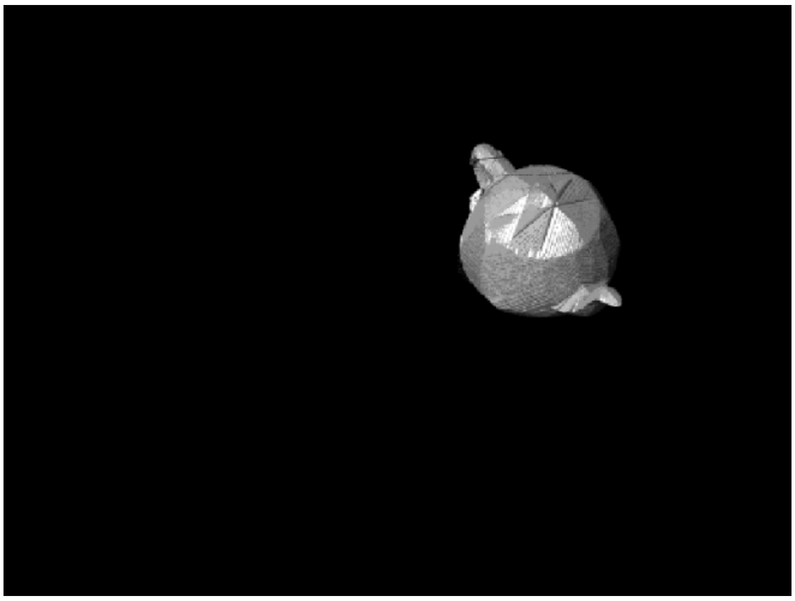
First we will tackle shape-from-silhouettes methods. Starting from the silhouettes extracted from the camera pictures, a conservative shell enveloping the true geometry of the object is computed. This generated shell is called the visual hull [Laurentini, 1994]. For 2D scenes, the visual hull is equal to the convex hull of the object, and for 3D scenes the visual hull is contained in the convex hull, where concavities are not removed but hyperbolic regions are. Even convex or hyperbolic points that are below the rim of a concavity (e.g. a marble inside a bowl) cannot be reconstructed. While the visual hull algorithms are efficient, the geometry they reconstruct is not very accurate. When observed by only a few cameras, the scene's visual hull is often much larger than the true scene. When rendering new views, one can partially compensate for such geometric inaccuracies by view-dependent texture-mapping [Debevec et al., 1996, Debevec et al., 1998]

Strictly, the visual hull is the maximal volume constructed from all possible silhouettes. In almost any practical setting, the visual hull of an object is computed with respect to a finite number of silhouettes. We call this type of visual hull the inferred visual hull.

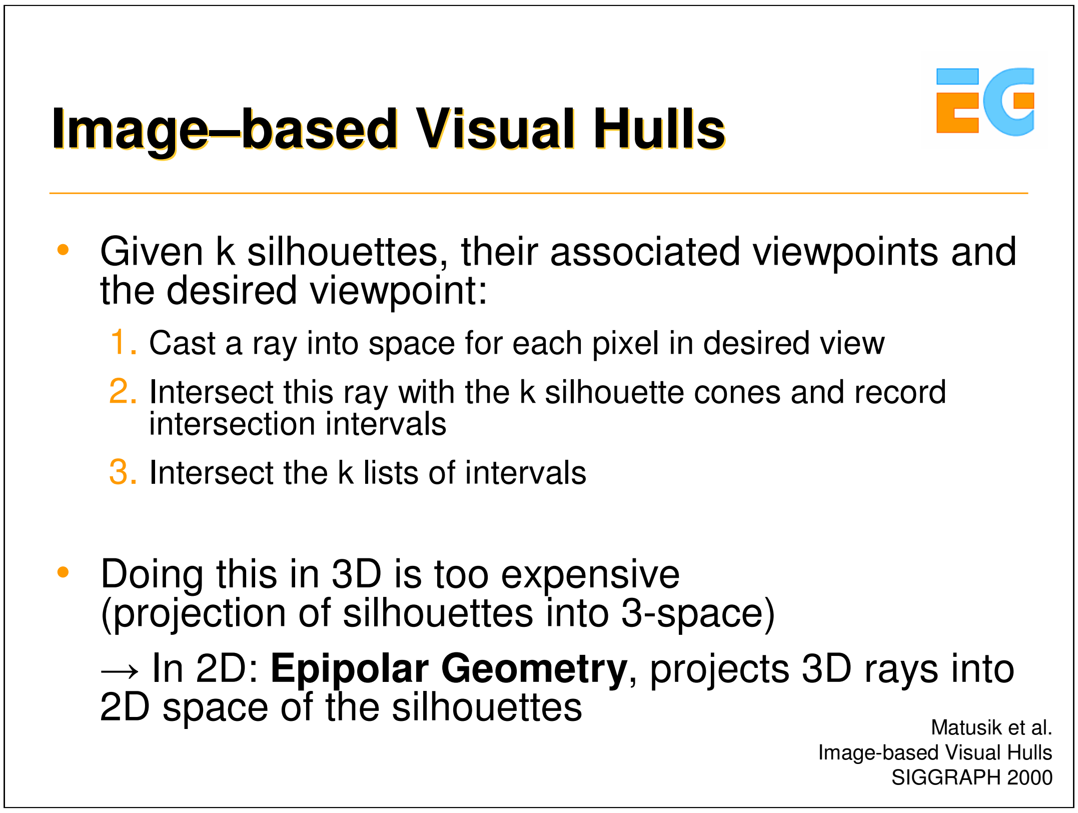
There exist two classes of methods to compute the visual hull, (1) voxel carving methods, which carve away all voxels that are not contained in the silhouettes of the acquisition cameras and (2) image-based methods, that exploit epipolar geometry and store so-called occupancy intervals at every pixel.



What is a Visual Hull?



Here is an animated illustration of how a visual hull is carved…



Image–based Visual Hulls

- Given k silhouettes, their associated viewpoints and the desired viewpoint:
    1. Cast a ray into space for each pixel in desired view
    2. Intersect this ray with the k silhouette cones and record intersection intervals
    3. Intersect the k lists of intervals

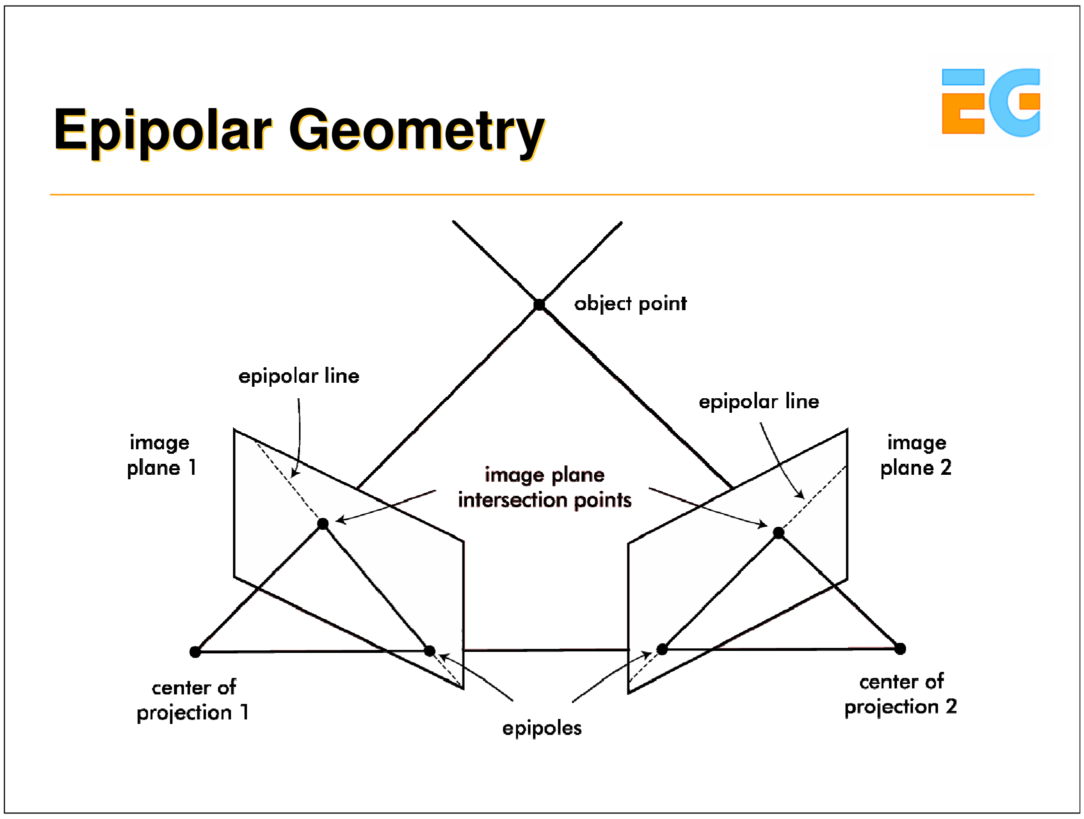
- Doing this in 3D is too expensive (projection of silhouettes into 3-space)

  → In 2D: **Epipolar Geometry**, projects 3D rays into 2D space of the silhouettes

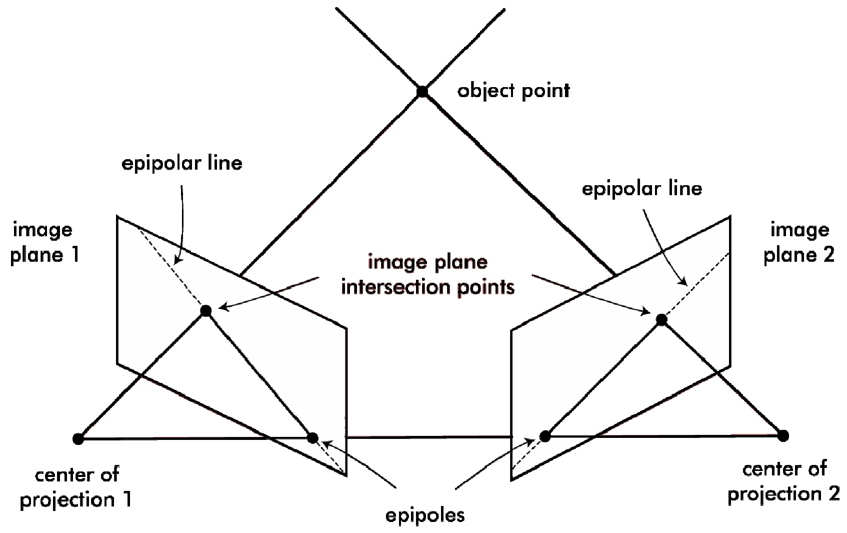
Matusik et al.
Image-based Visual Hulls
SIGGRAPH 2000

We explain a particularly fast shape-from-silhouettes algorithm – which is able to perform in real-time - the image-based visual hulls method as presented by [Matusik et al., 2000].

The IBVH method takes advantage of epipolar geometry to accelerate calculation of depth values and to achieve real-time performance. As opposed to volumetric reconstruction techniques, e.g. voxel carving, the IBVH algorithm does not suffer from limited resolution, or quantization artifacts due to the underlying explicit voxel representation.



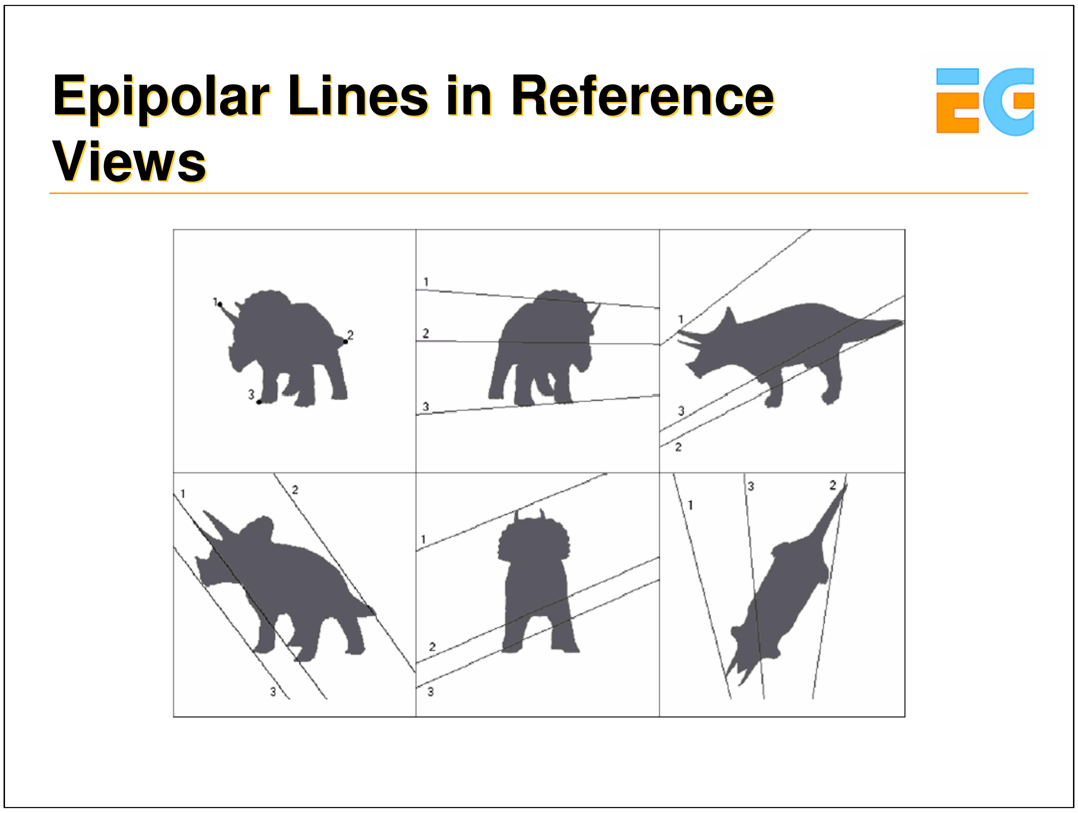
Epipolar Geometry



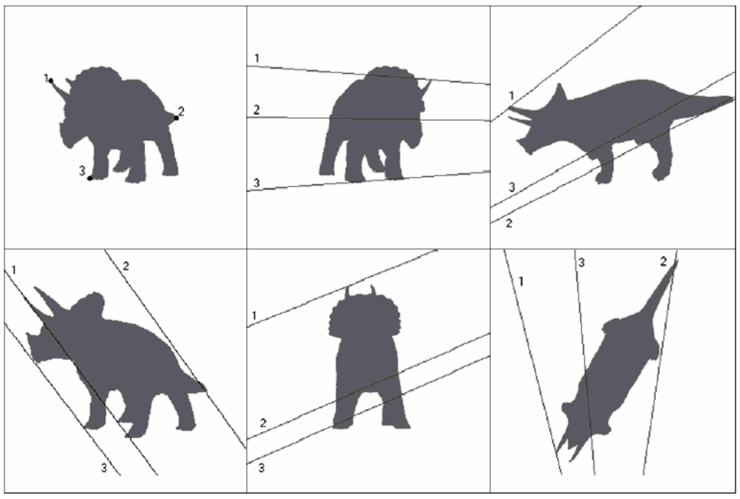
An *epipole* lies at the intersection of the baseline of the two cameras with the image plane of one of the cameras. Therefore, the epipole is the projection of the center of projection of one camera into the image plane of another camera.

An *epipolar plane* is defined by both centers of projection and a 3D point . Each plane containing the baseline is an epipolar plane, and intersects the image planes in corresponding epipolar lines, which also represent the projection of the ray from the center of projection of the other camera to the point . As the position of the 3D point varies, the epipolar planes “rotate” around the baseline. This one-parameter family of planes is known as an *epipolar pencil*. The respective epipolar lines intersect at the epipole.

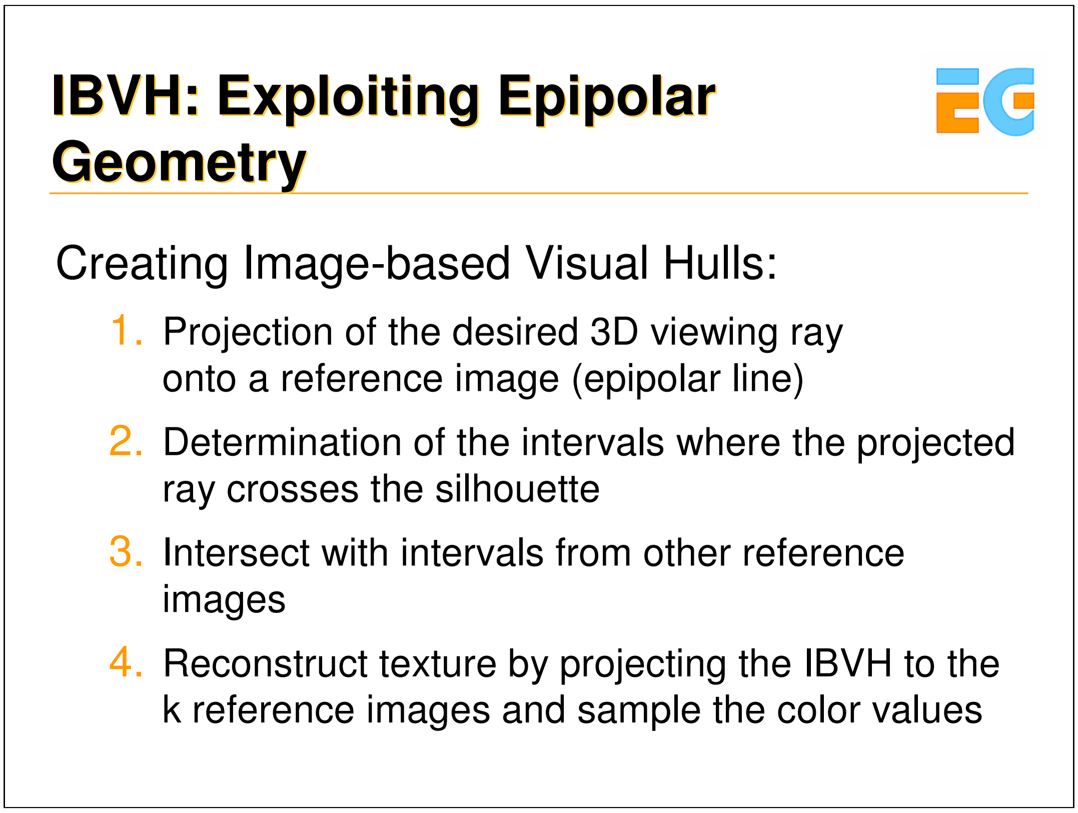
The benefit of epipolar geometry in terms of a 3D reconstruction algorithm is that the search for a point corresponding to a point in another image plane need not cover the entire image plane, but can be restricted to an epipolar line.



Epipolar Lines in Reference Views



Here’s an illustration of the epipolar lines of some points in one reference image, projected into the other images.



IBVH: Exploiting Epipolar Geometry

Creating Image-based Visual Hulls:

1. Projection of the desired 3D viewing ray onto a reference image (epipolar line)
2. Determination of the intervals where the projected ray crosses the silhouette
3. Intersect with intervals from other reference images
4. Reconstruct texture by projecting the IBVH to the k reference images and sample the color values

For estimation depth for a given pixel or fragment a ray has to be cast into space from that pixel. By making use of epipolar geometry this ray is projected to line segments in all other reference images (1).

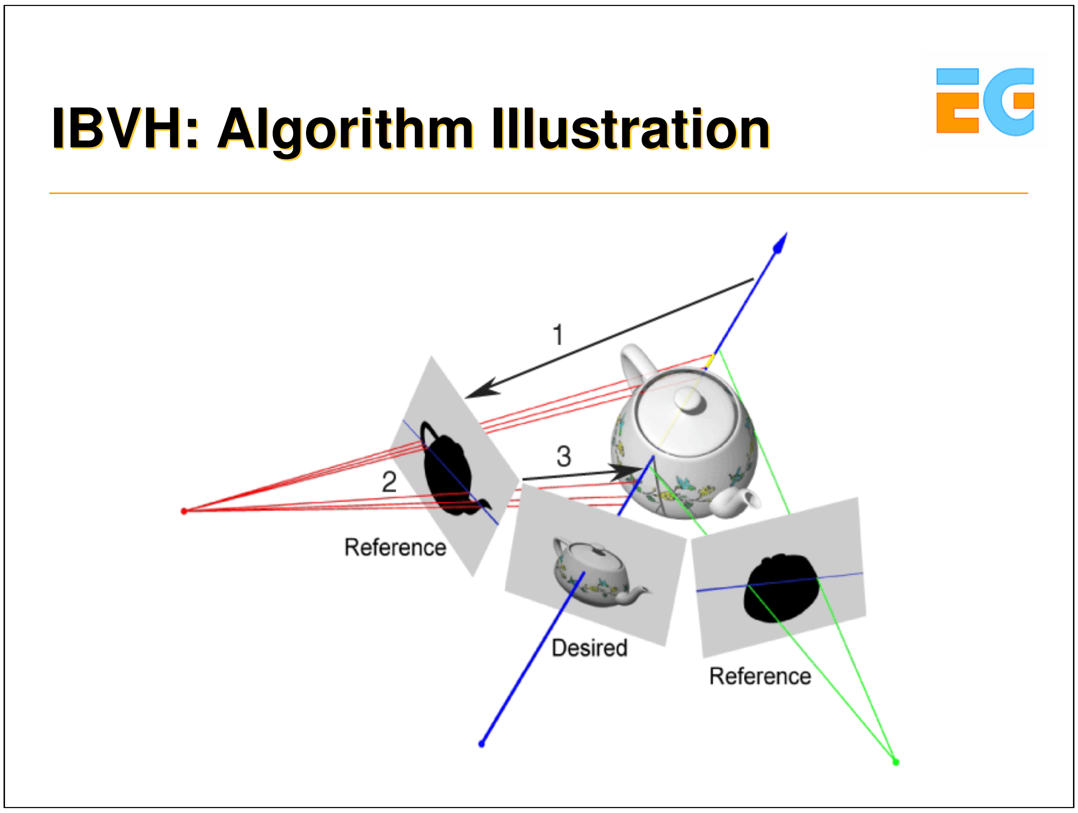
There, the intersection/intervals is calculated with the binary silhouette (2). The resulting intersection points are lifted back onto the original ray where intersection intervals are built. They are represented as pairs of enter/exit points.

The intervals can be intersected with intervals from all other reference images (3).

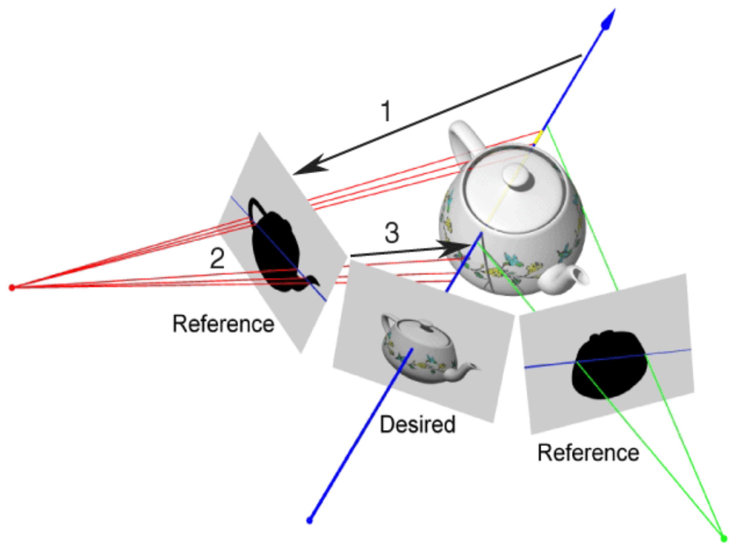
Finally, texture is reconstructed in the desired view by projecting the IBVH data to all reference images and blending the color values together.

The result is basically a LDI representation of the geometry as seen from a specific camera. The key aspect

of the IBVH algorithm is that all intersection calculations can be done in two dimensions rather than three.



IBVH: Algorithm Illustration



And here’s an illustration of the IBVH process for the notes.

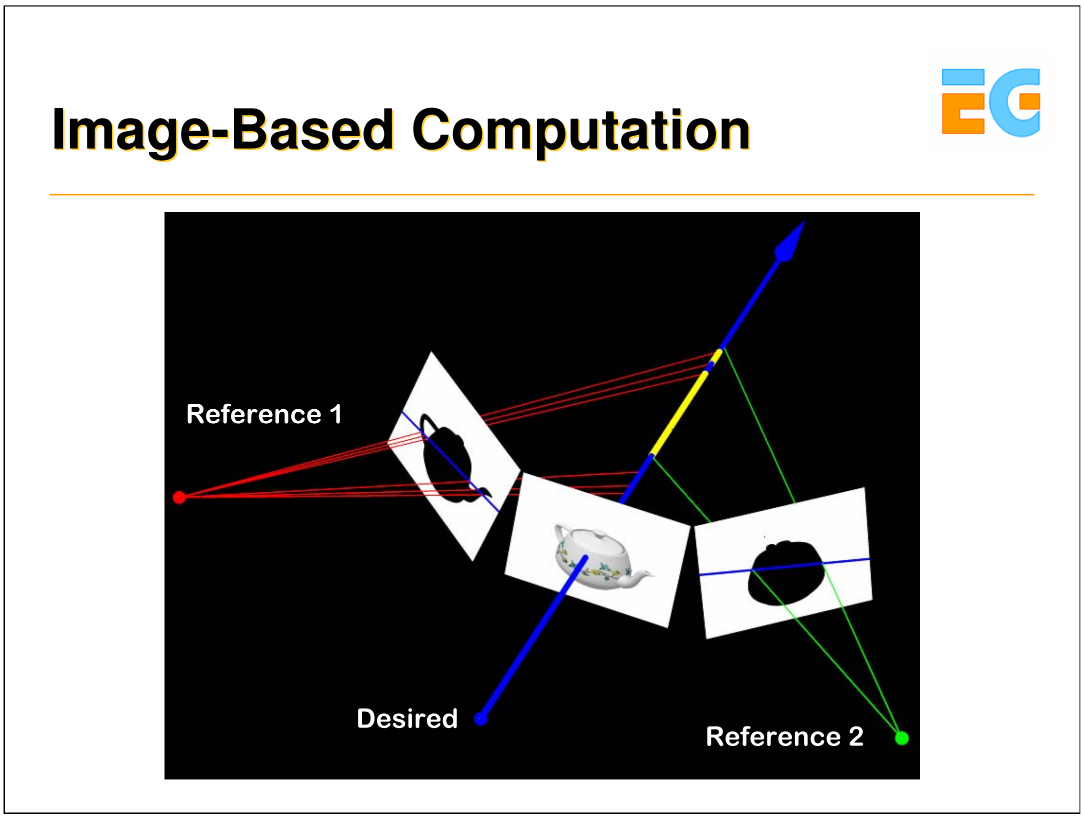
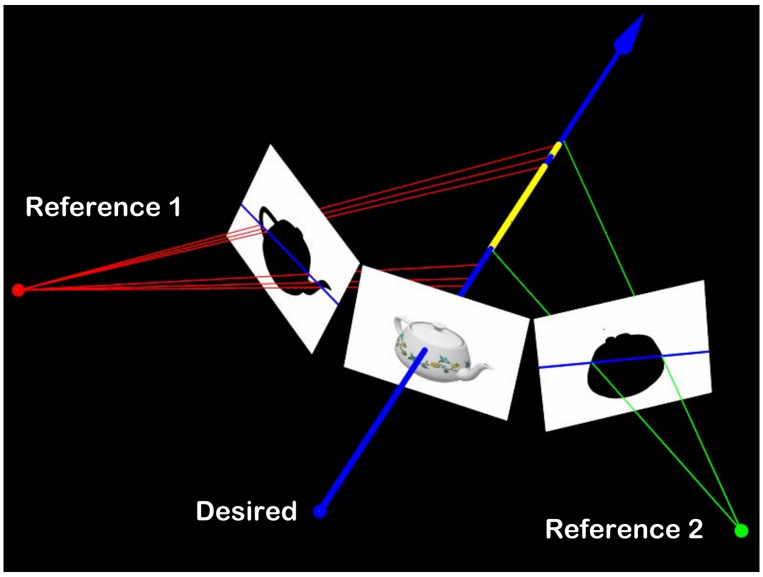
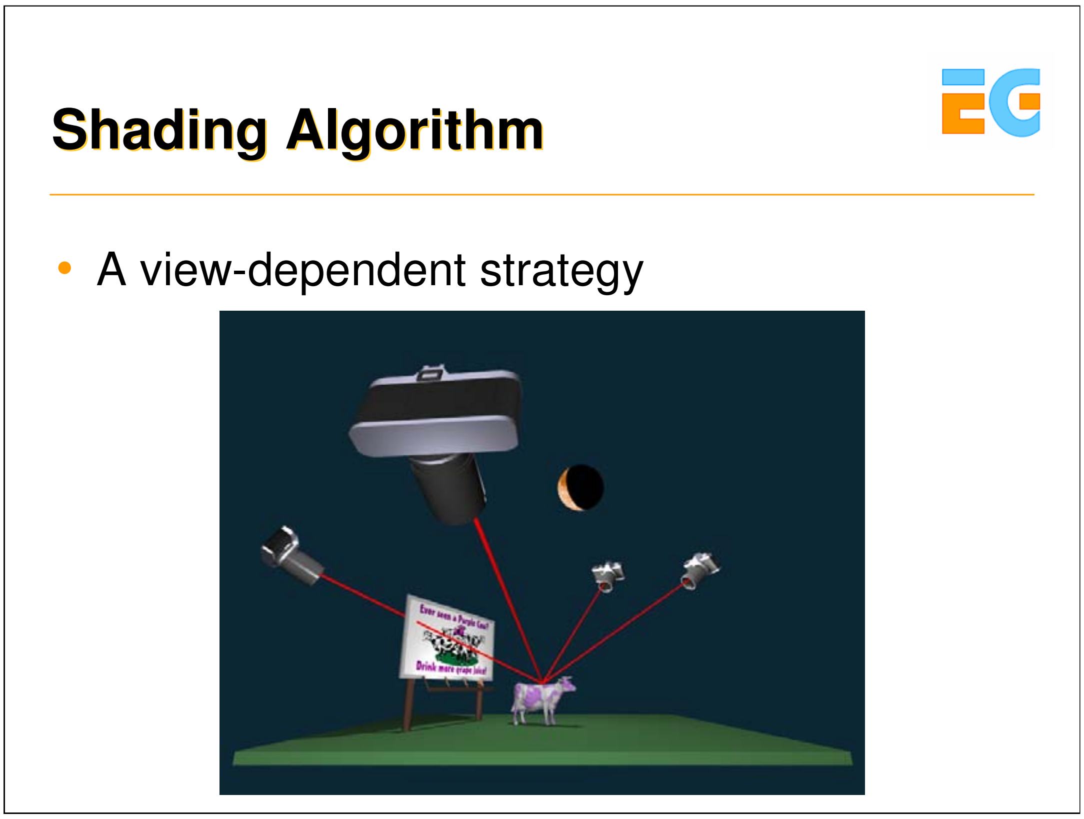


Image-Based Computation

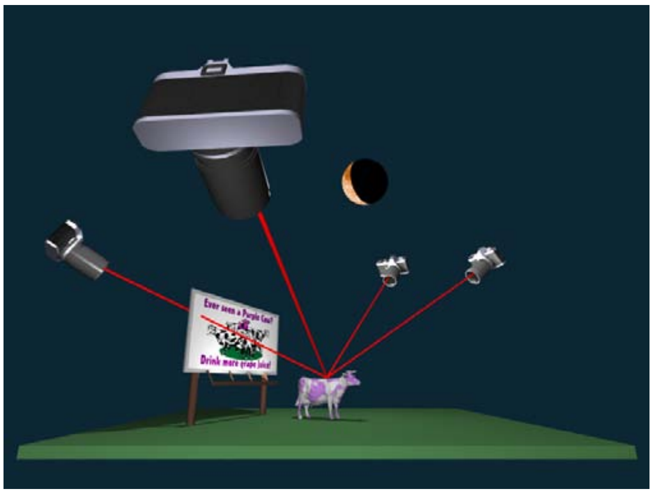


And here's an animated illustration of the IBVH process.



Shading Algorithm

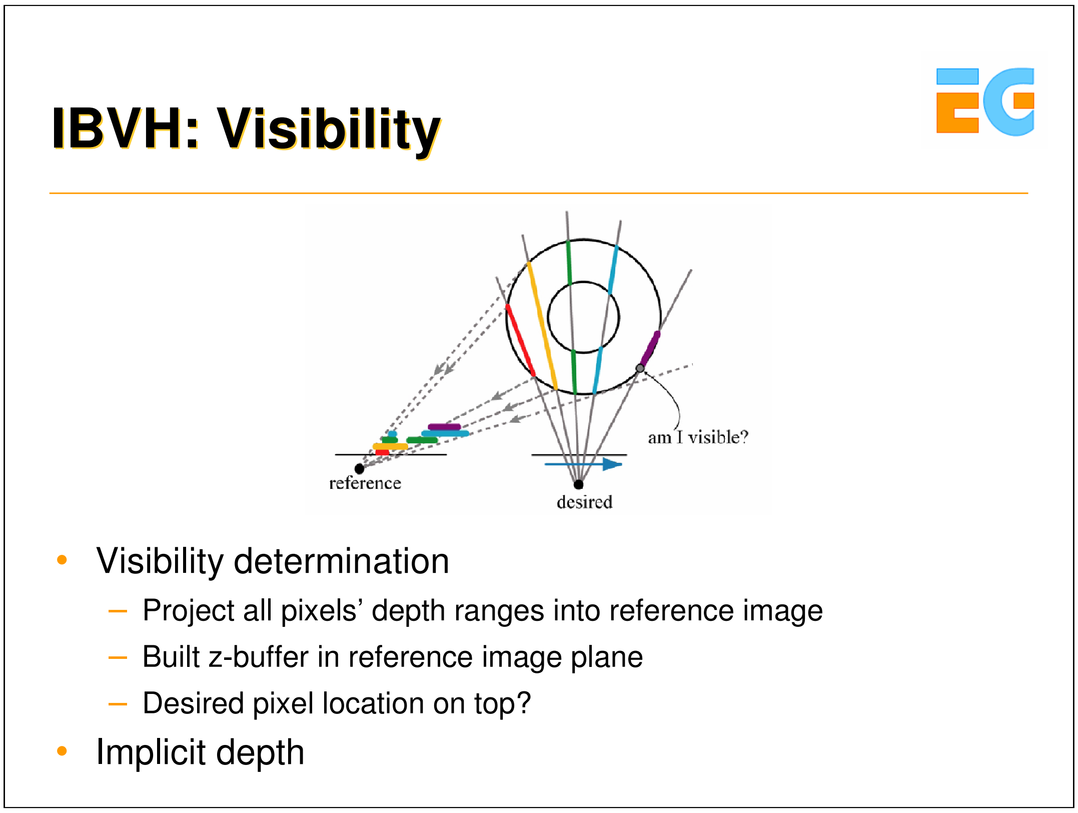
- A view-dependent strategy



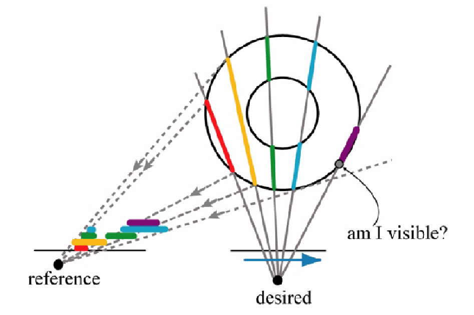
Finally, texture is reconstructed in the desired view by projecting the IBVH data to all reference images and blending the color values together.

Different techniques exist to blend the textures together, mostly applied is the Unstructured Lumigraph Rendering framework.

Care has to be taken for visibility.



IBVH: Visibility



- Visibility determination
  - Project all pixels' depth ranges into reference image
  - Built z-buffer in reference image plane
  - Desired pixel location on top?
- Implicit depth

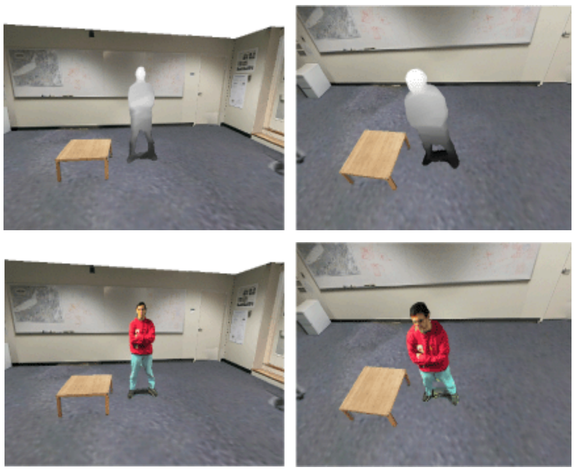
In order to compute the visibility of an IBVH sample with respect to a given reference image, a series of IBVH intervals are projected back onto the reference image in an occlusion-compatible order. The front-most point of the interval is visible if it lies outside of the unions of all preceding intervals.

Once more we can take advantage of the epipolar geometry in order to incrementally determine the visibility of points on the visual hull.

IBVH: Original Results

Input:

4 segmented reference images



Upper: depth maps of the computed visual hulls
Lower: shaded renderings from the same viewpoint

And here are some results for the notes.

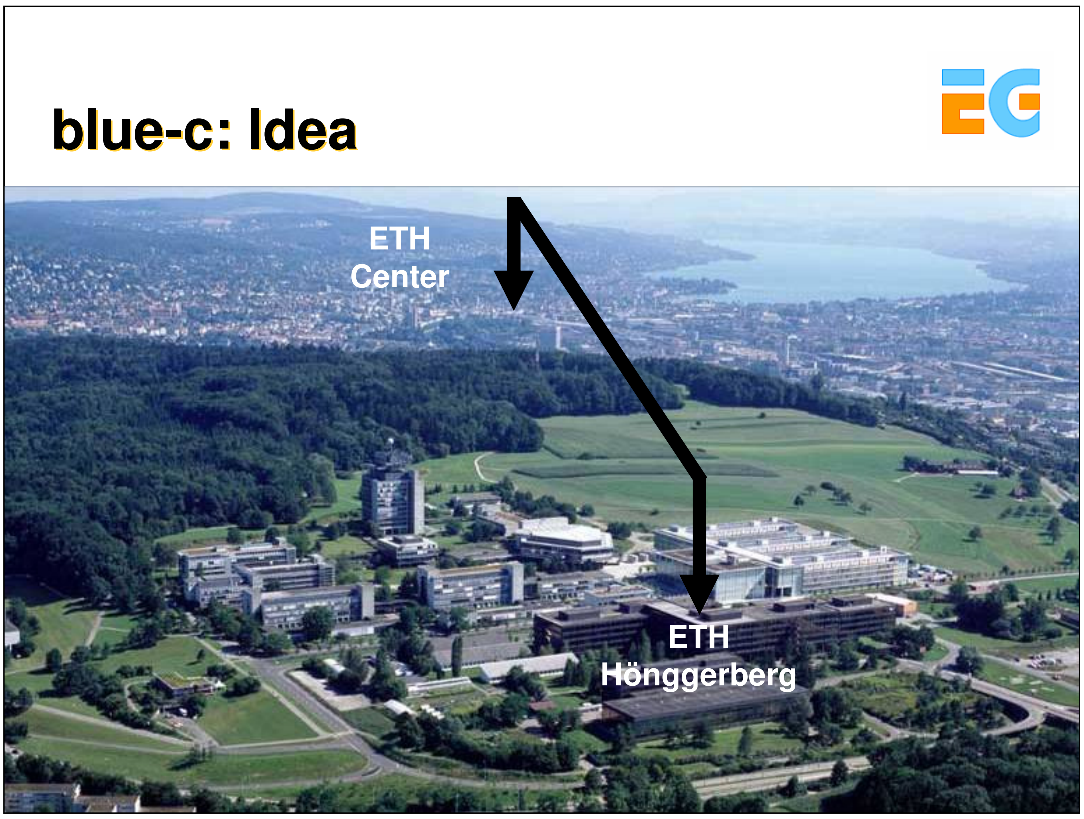
IBVH: Original Results Video

The blue-c

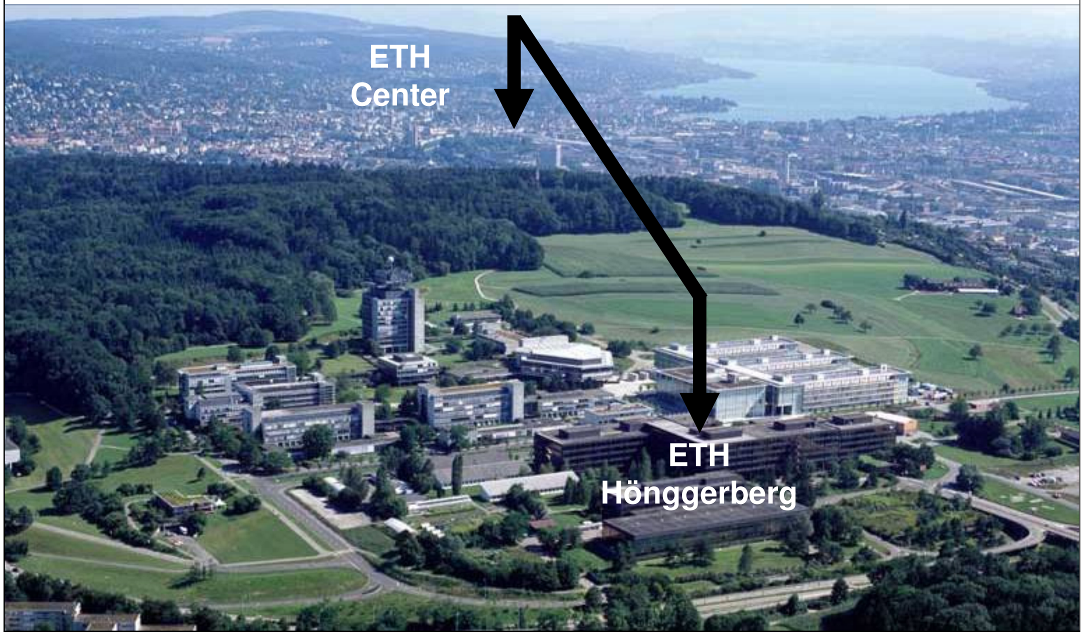
Gross et al.
The blue-c
SIGGRAPH 2003

Such techniques can now be exploited for telepresence applications since they provide 360 degree viewing of persons. ETH Zurich conducted a huge project in 1999-2004 which is called the blue-c. It exploited shape-from-silhouettes algorithms to connect two spatially immersive environments to be able to have telecollaboration sessions – seeing the remote participant in full 3D.

Here is example of how blue-c works. It connects a 3 sided CAVE enviironment located inthe ETH computer center downtown Zurich with a second site on our campus outside Zurich. This second site consists of a signle stereo projection panel only. As a central feature of our system, both sites are equipped with 16 video cameras capturing 3D video of the blue-c users. This allows for immersive 3D telepresence applications as the one you can see in the video clips.

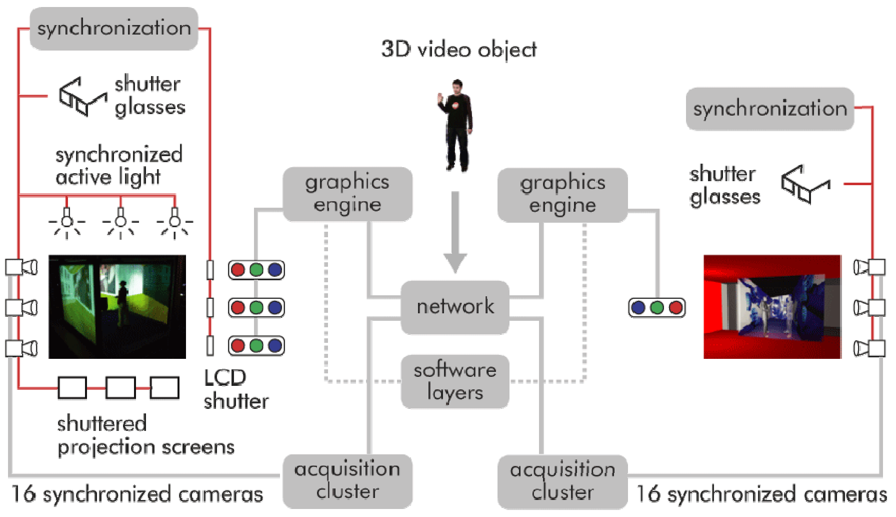


blue-c: Idea



The blue-c connects the two physically remote ETH campuses in Zurich, a distance of approx. 10 miles. The basic idea was to connect both campuses with a next-generation telepresence system.

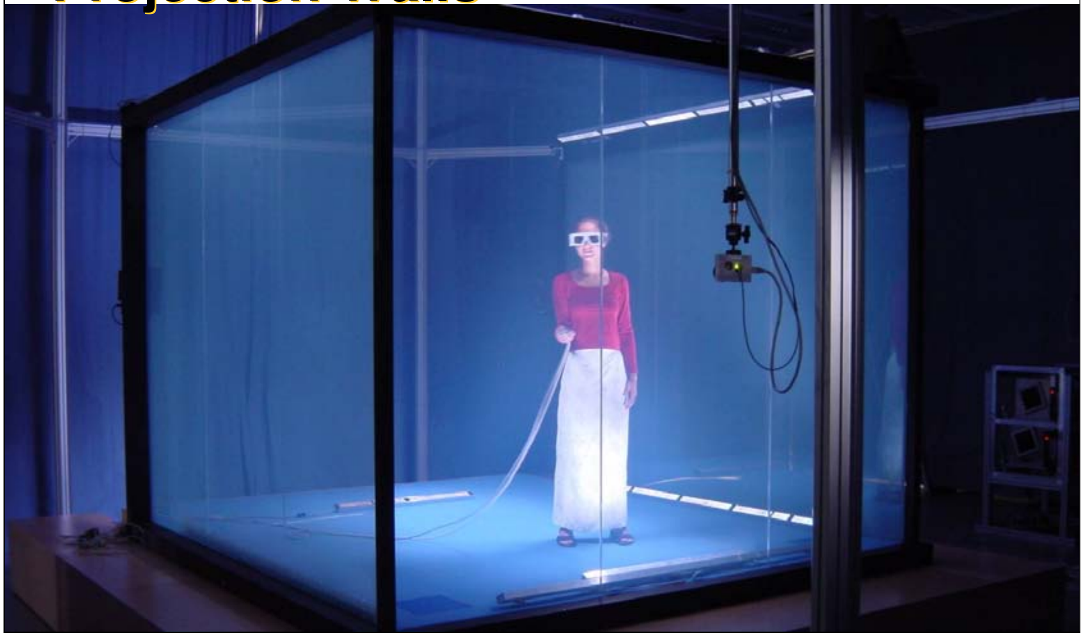
blue-c: System Overview



This picture displays an overview of the system architecture. We can clearly see that our setup is asymmetric. Besides costs, the major reason for this asymmetric design was to demonstrate scalability.

On the left we see the core hardware components being involved to accomplish simulaneous immersive projection and acquisition. This includes multiple cameras, shuttered projection screens, shutter glasses, an active lighting system, and an actively shuttered projection system. All hardware components are synchronized using a specially designed sync hardware. The cameras transfer 2D video frames to a PC cluster which computes a 3D video inlay of the user in realtime. This inlay is streamed over the network to the partner site and is composited into the synthetic scene by the graphics engine. We use both PCs and SGI Onyx 3200.

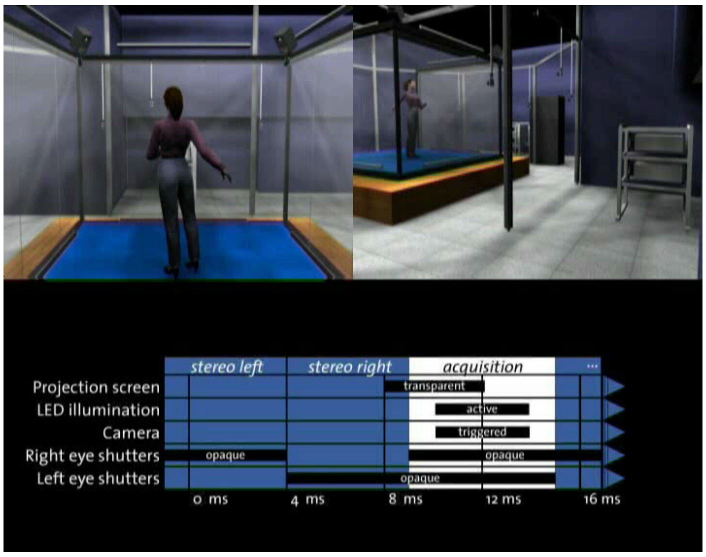
blue-c: Switchable Projection Walls



One of the core technical challenges when combining video acquisition and immersive projection is the placement of the cameras. As a central part of our design we place most of the cameras outside the projection space, which are, hence, not visible to the user. 5 remaining cameras are attached to the upper corners and to the ceiling to facilitate color calibration and texture acquisition.

It is easy to see that the projection screens occlude the user from the outside cameras. We solve this problem by using phase dispersed liquid crystal panels. These panels are switched from an opaque state during projection to an transparent state during acquisition. We do this at 62.5 Hz which is well above the fusion frequency of the human visual system.

blue-c: Timing

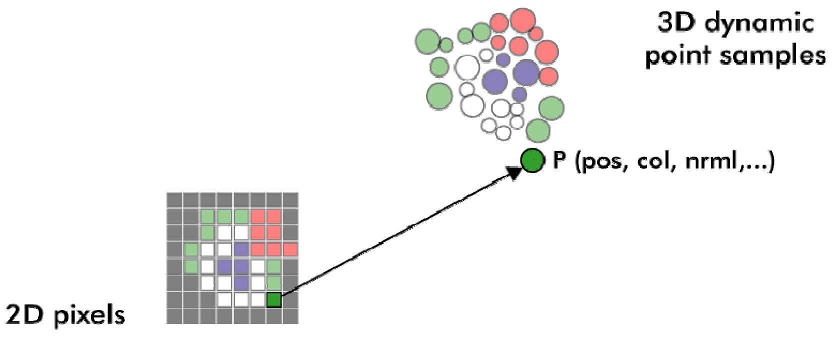


The following video illustrates the timing and synchronization of the involved hardware components.

We first project the image for the left eye then the image for the right eye.

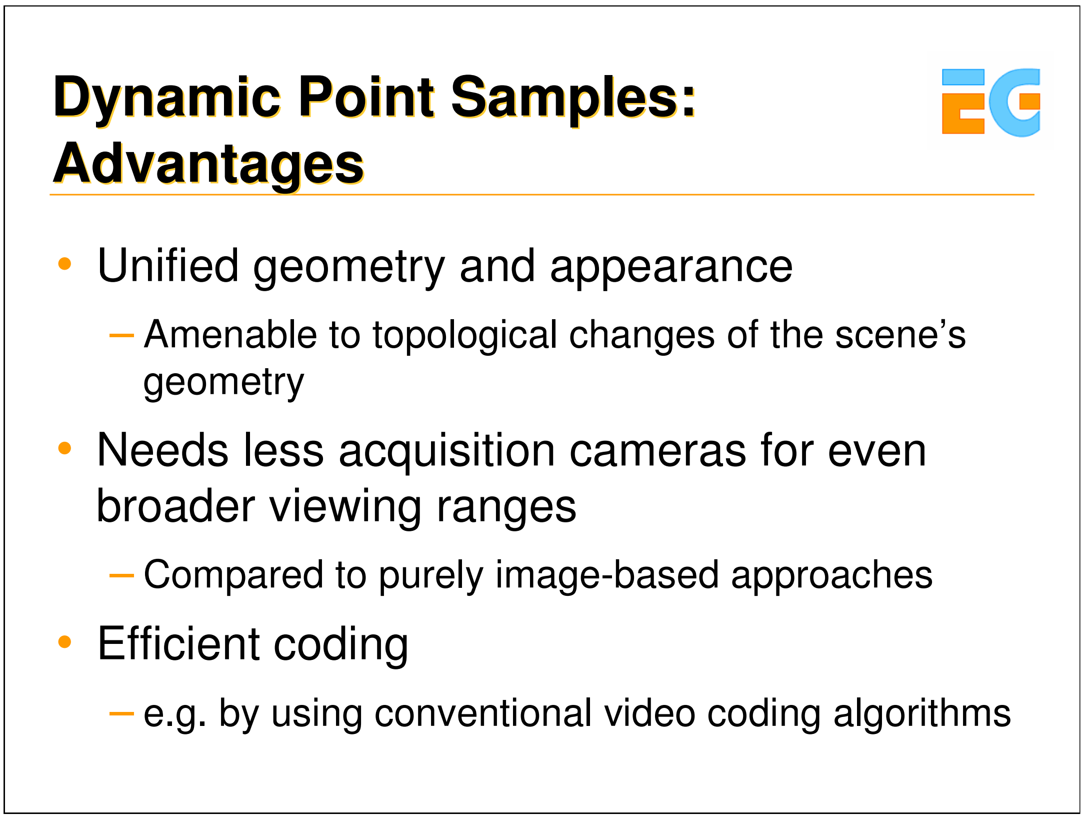
During a small time window of about 4 ms between the projection cycles, we open the walls and acquire the video frame. Due to hardware limitations the system currently graps frames in every 7th window resulting in 9 Hz update rate. To improve the quality of the texture acquisition, we built an active lighting system which is synchronized with the video acquisition.

Dynamic Point Samples



Würmlin et al.
3D video fragments
Computers and Graphics 2004

The basic primitives of the pipeline are 3D video fragments, which are dynamic point samples with attributes like, e.g., a position, a surface normal vector, and a color. 3D Video Fragments are a generalization of 2D video pixels towards 3D irregular point samples and we can therefore benefit from earlier work on point-based graphics.



Dynamic Point Samples: Advantages

- Unified geometry and appearance
  - Amenable to topological changes of the scene's geometry
- Needs less acquisition cameras for even broader viewing ranges
  - Compared to purely image-based approaches
- Efficient coding
  - e.g. by using conventional video coding algorithms

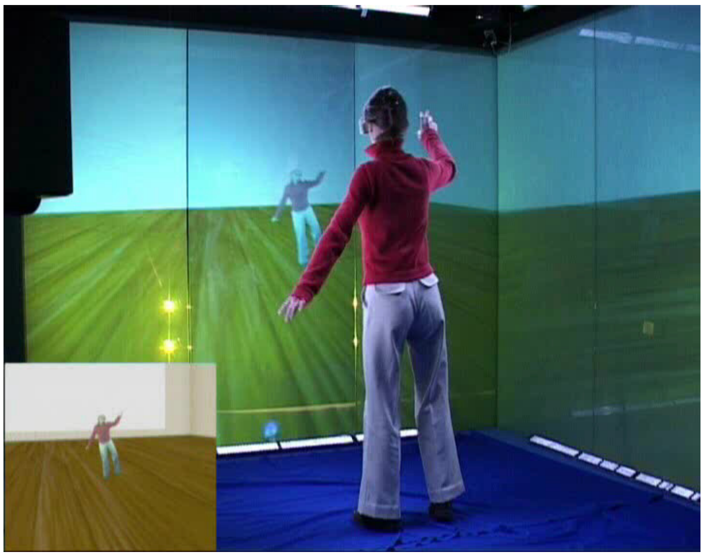
Dynamic points have some advantages over other primitives.

(1) It is a unified representation, holding geometry and appearance as one, and are amenable to topological changes of the scene's geometry

(2) It needs less acquisition cameras for even broader viewing ranges compared to purely image-based approaches because it explicitly encodes the scene's geometry information.

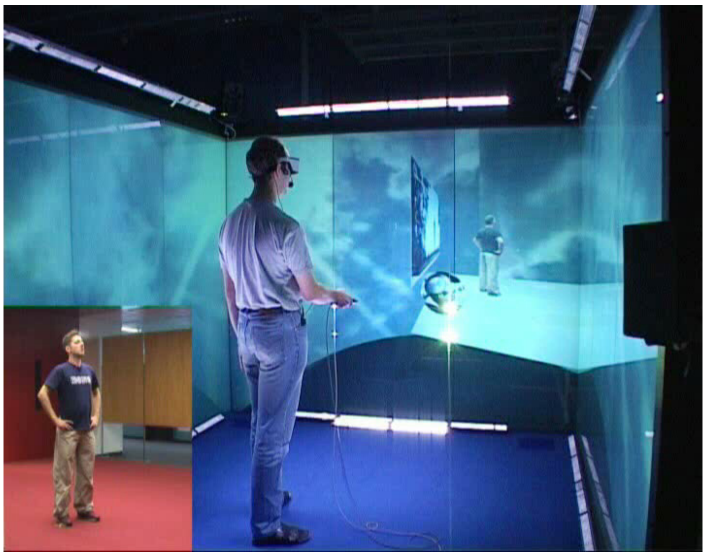
(3) It has potential for efficient coding schemes due to its simplicity, e.g. by using conventional video coding algorithms when stored in an image-space representation.

blue-c: 3D Mirror



The following example shows a 3D mirror application we built to demonstrate the concept. The user can experience herself in full 3D. She can freely move the camera and fly around herself. The cameras are now looking through the projection screens. This video gives a good feeling of the projection quality. It was recorded in realtime using a conventional unsynchronized camcorder.

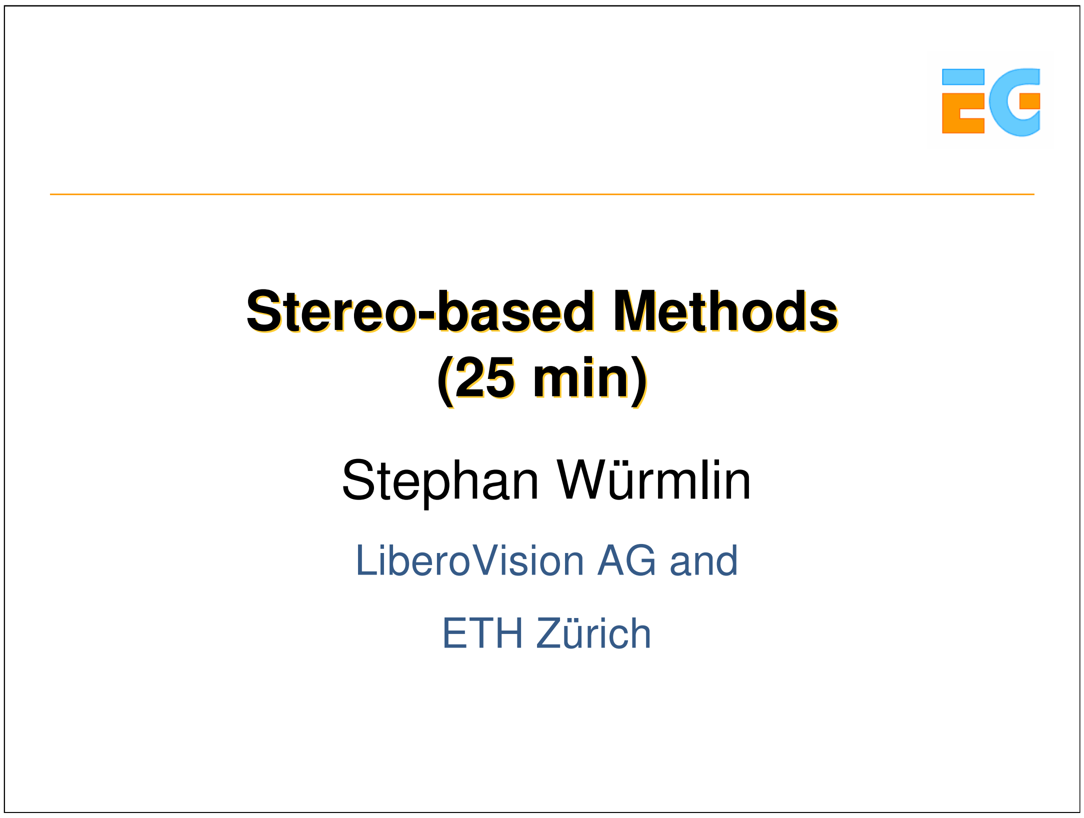
Results – blue-c Video



And here’s a video with results.

Acquisition was done at ETH Hoenggerberg outside of Zurich as illustrated in the video inlay in the bottom-left corner.

The 3D video inlay is then streamed to the blue-c installation at ETH Computing Center in real-time and composited with the virtual scene.

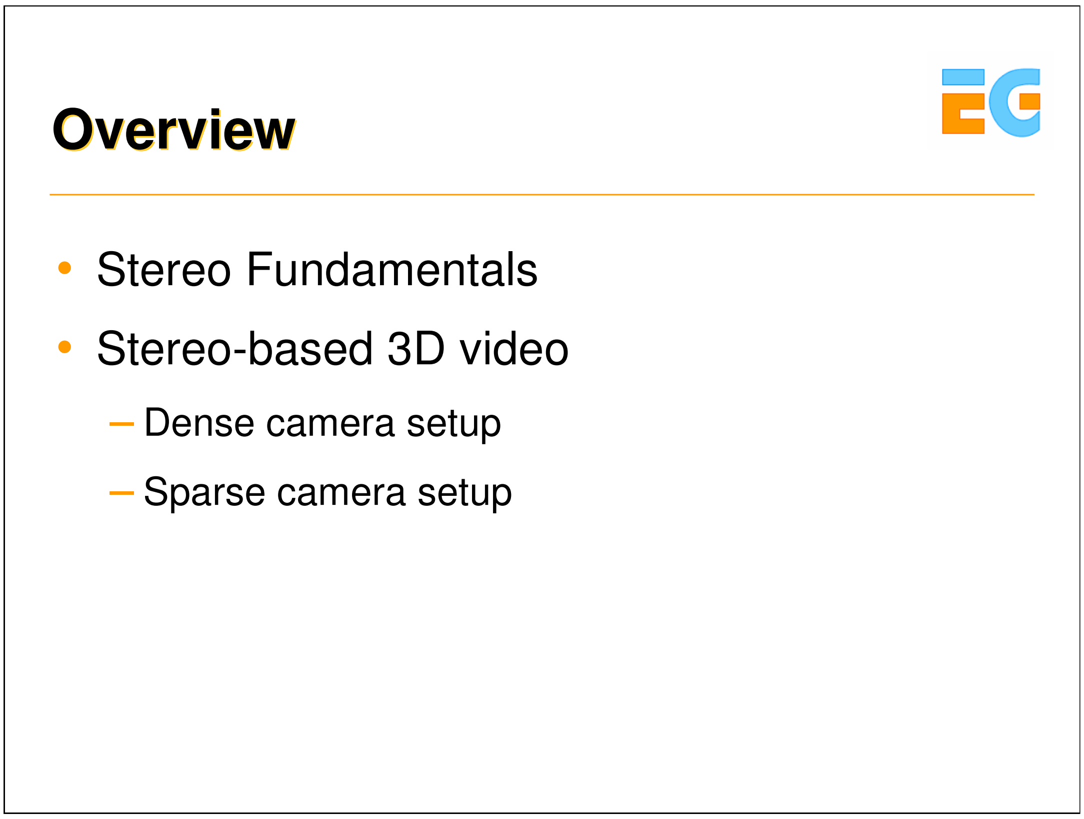


Stereo-based Methods (25 min)

Stephan Würmlin

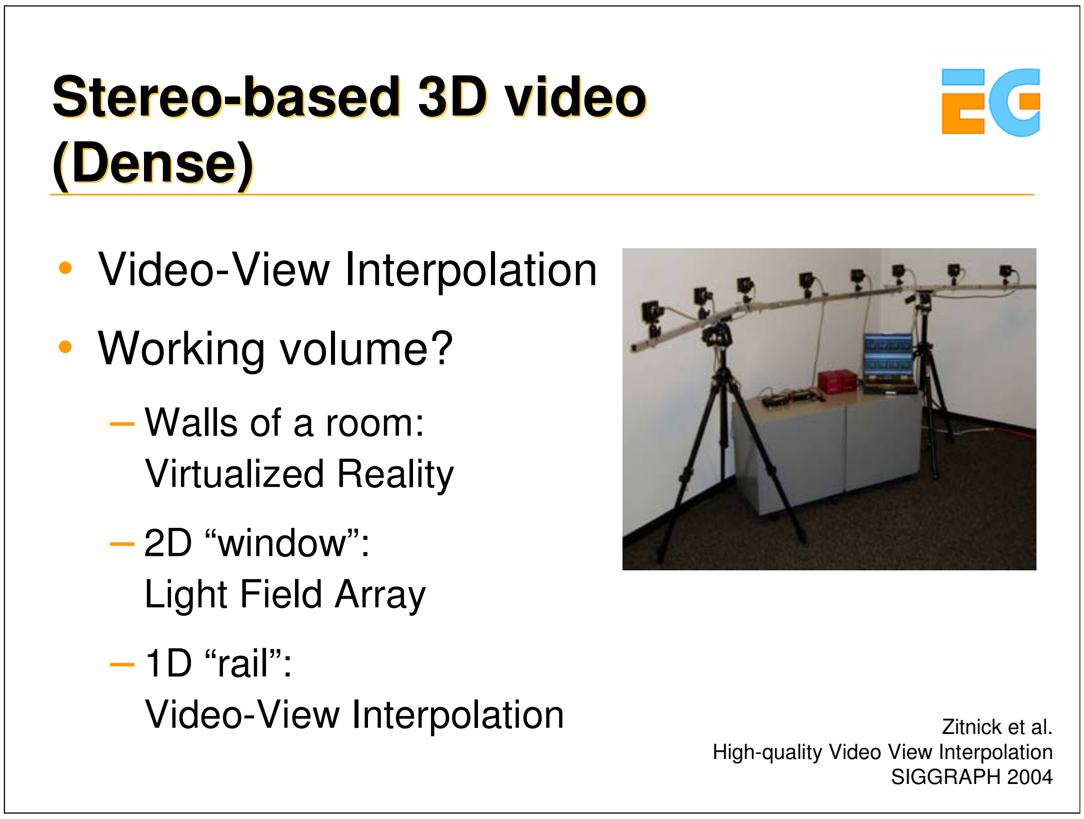
LiberoVision AG and

ETH Zürich



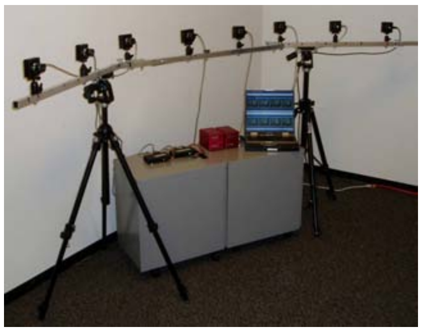
Overview

- Stereo Fundamentals
- Stereo-based 3D video
  - Dense camera setup
  - Sparse camera setup



Stereo-based 3D video (Dense)

- Video-View Interpolation
- Working volume?
  - Walls of a room: Virtualized Reality
  - 2D “window”: Light Field Array
  - 1D “rail”: Video-View Interpolation

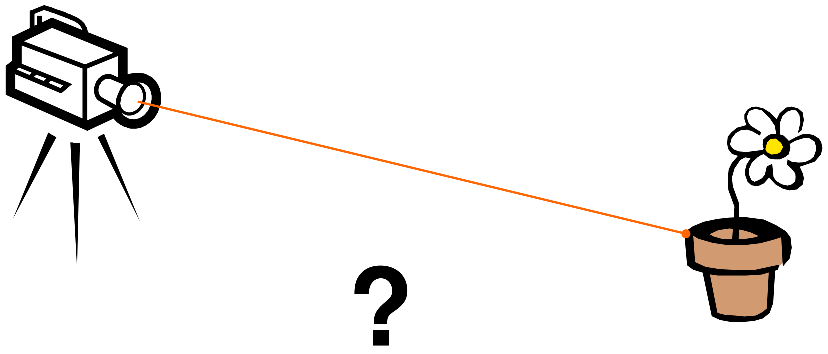


Zitnick et al.
High-quality Video View Interpolation
SIGGRAPH 2004

Stereo-based 3D video is able to not only capture one or two objects – due to the constraint of separable silhouettes for the employed shape-from-silhouettes algorithm – but can handle entire scenes. Techniques vary depending on the amount of freedom in navigation that a system wants to achieve. As an example, the Virtualized Reality project at CMU tried to enable full 360 degree freedom while a light field array only gives the possibility to give the user a 2D window into the world.

An interesting approach where I want to go in a little more detail is the video-view interpolation project at MSR where they tried to come up with a production-quality 3D video system but give the viewer only the ability to navigate on a 1D “rail”. For that they employed depth-from-stereo algorithms and the camera’s were placed rather dense as you can see in the image.

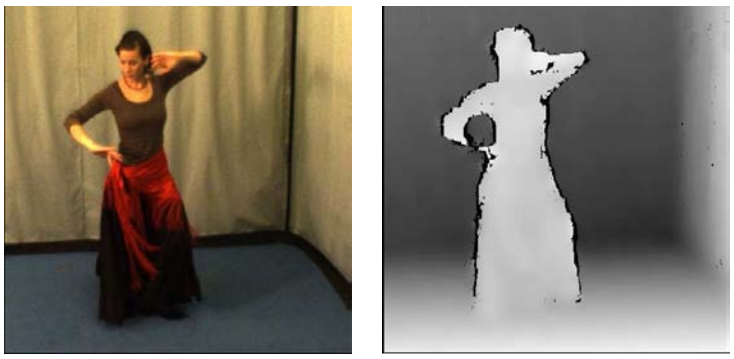
3D Reconstruction



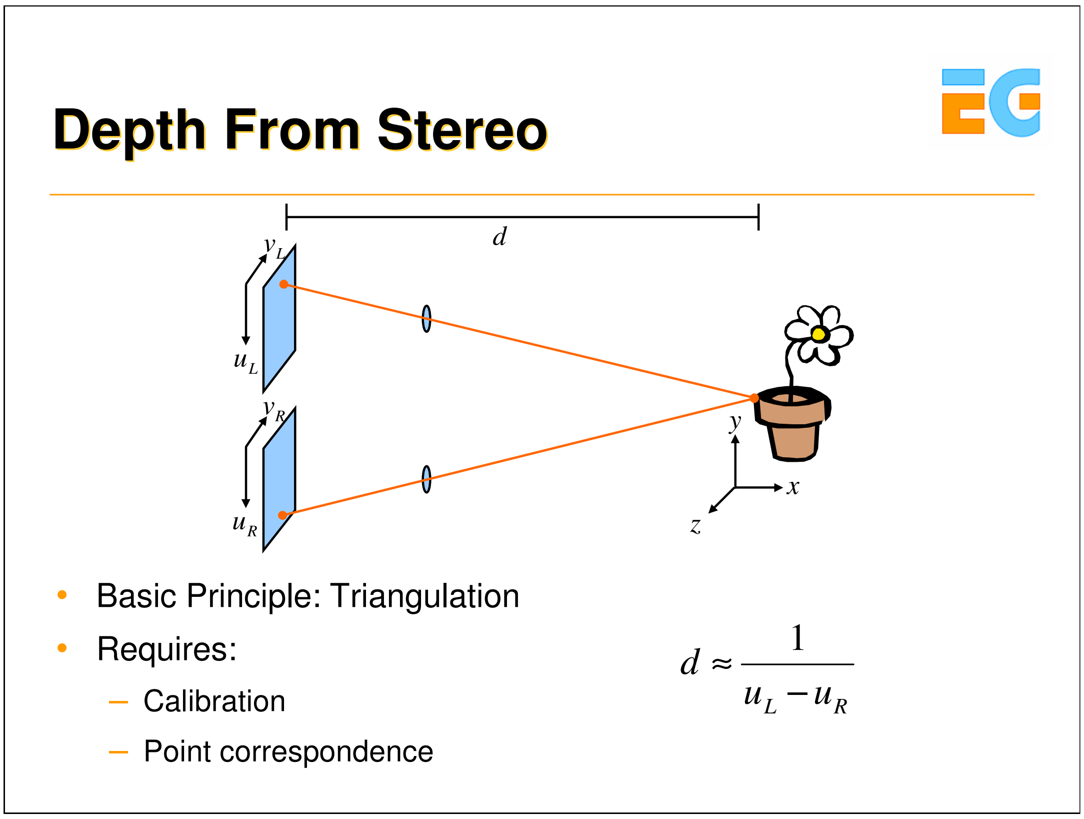
Again we have to know where the 3D points are that we image by the camera.

Depth Map

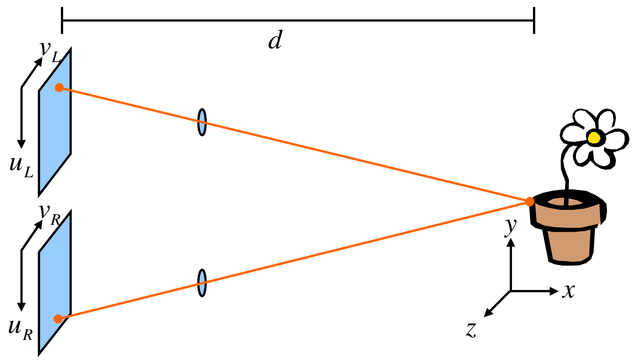
- Gray value encodes distance from camera



This means that we should calculate a depth map, indicating the distance from each pixel to the 3D surface point. On the right you see gray-coded the distance from the camera, with darker regions indicating farther away surface points and brighter regions indicate closer surfaces.



Depth From Stereo



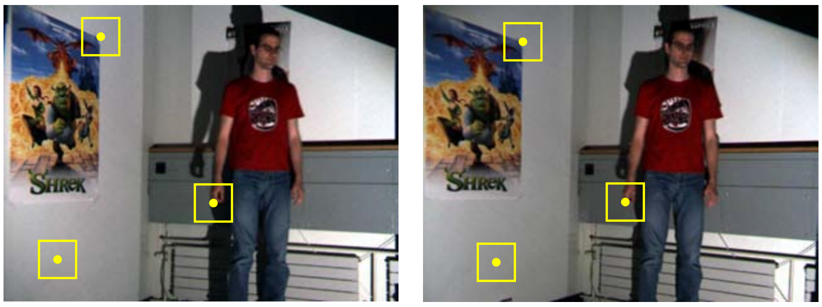
- Basic Principle: Triangulation
- Requires:
  - Calibration
  - Point correspondence

$$d \approx \frac{1}{u_L - u_R}$$

The basic principle of depth-from-stereo is triangulation. When you know where a surface point is projected in two camera images you can – with an appropriate calibration of the cameras triangulate the distance that point has from the image. But to be able to do that you need point correspondences.

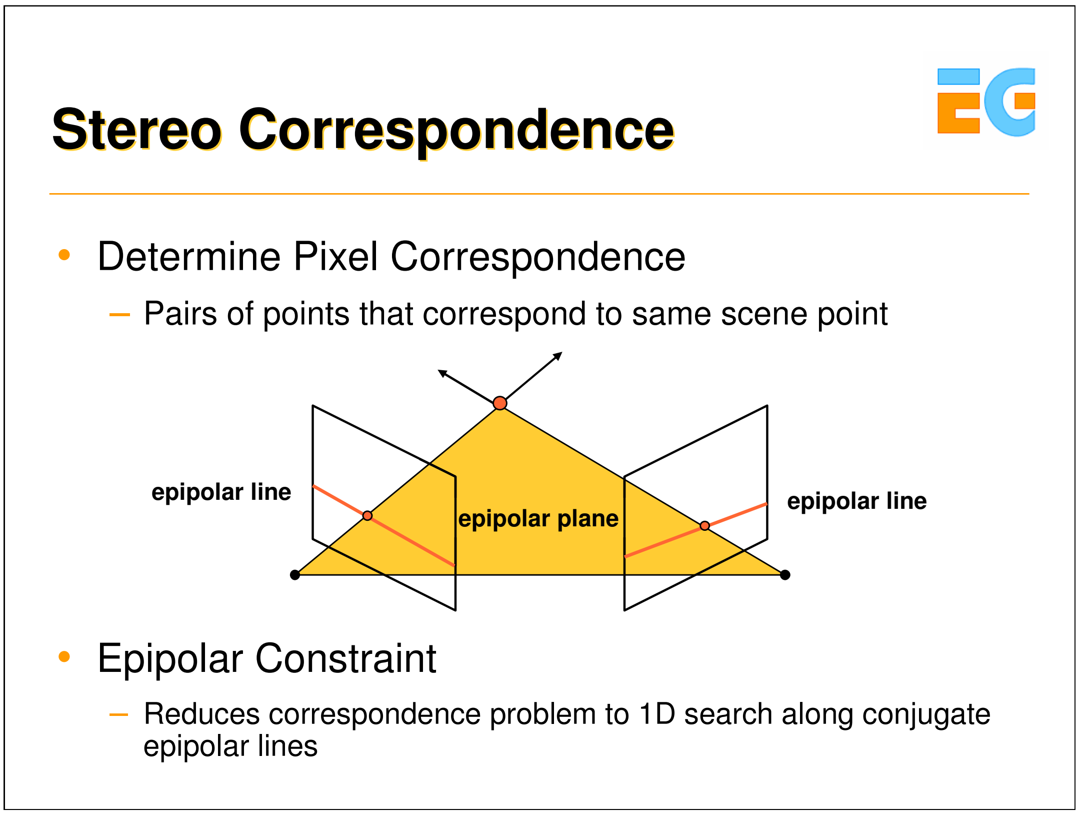
Stereo Vision

- Search for corresponding pixels



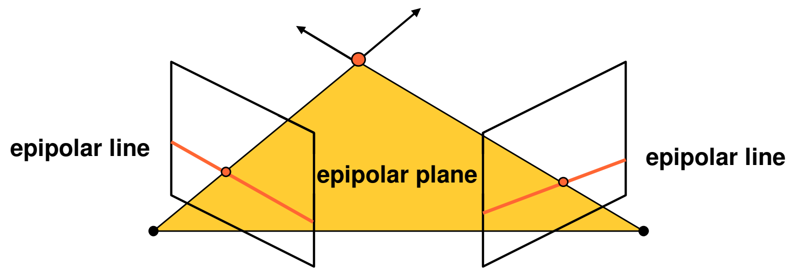
- Use windows to help you
  - But can still fail due to lack of texture!

Here is an example of what an algorithm should do. Instead of only calculating color similarities on single pixels, many methods employ a window-based approach. However, this can still lead to ambiguities and false depths in regions where there is not enough texture detail.



Stereo Correspondence

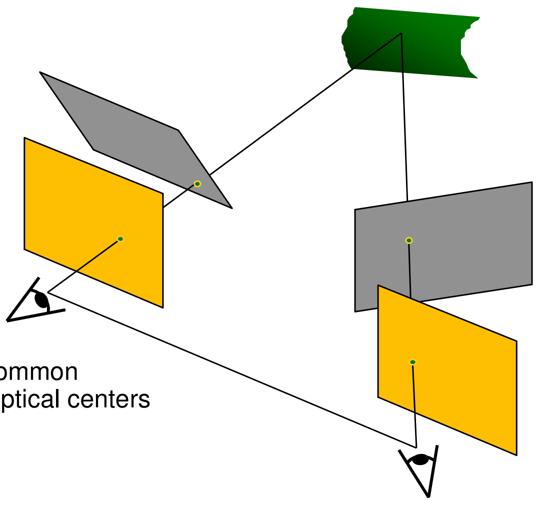
- Determine Pixel Correspondence
  - Pairs of points that correspond to same scene point



- Epipolar Constraint
  - Reduces correspondence problem to 1D search along conjugate epipolar lines

To determine pixel correspondences you need to search for pairs of points that correspond to the same scene point. This can be arbitrarily difficult to find in general – and hence arbitrarily time consuming because you need to do an exhaustive search. By employing again an epipolar constraint we can reduce the correspondence problem to a 1D search along conjugate epipolar lines as indicated in the picture.

Stereo Image Rectification

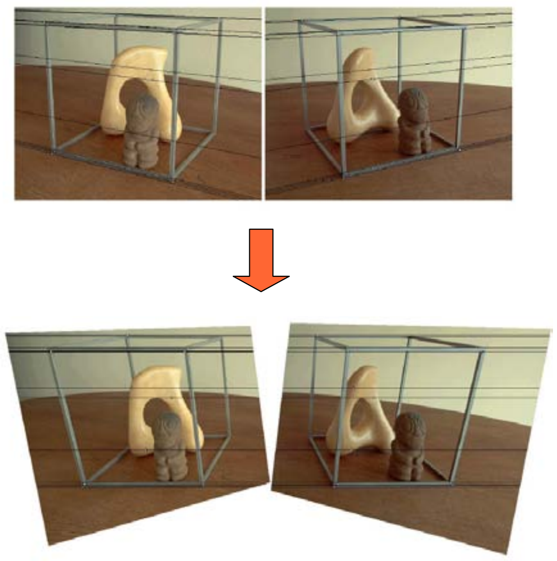


- Image Reprojection
  - Reproject image planes onto common plane parallel to line between optical centers
  - A homography (3x3 transform) applied to both input images
  - Pixel motion is horizontal after this transformation

Loop and Zhang
Computing Rectifying Homographies for Stereo Vision
IEEE Conf. Computer Vision and Pattern Recognition 1999

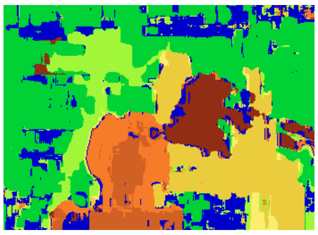
For that we need to rectify the image pair which means that we reproject the image planes onto a common plane parallel to the line between the optical centers. This can be performed by applying a homography – a 3x3 transform – applied to both images. After rectification pixel motion is horizontal and we can search the correspondence on the same horizontal lines in the other image.

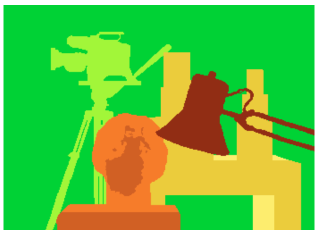
Stereo Rectification

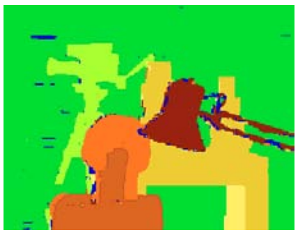


This is an example of a stereo rectification where you clearly see that the features are afterwards located on horizontal lines.

Different Stereo Methods Exist...


Window-based matching


Ground truth

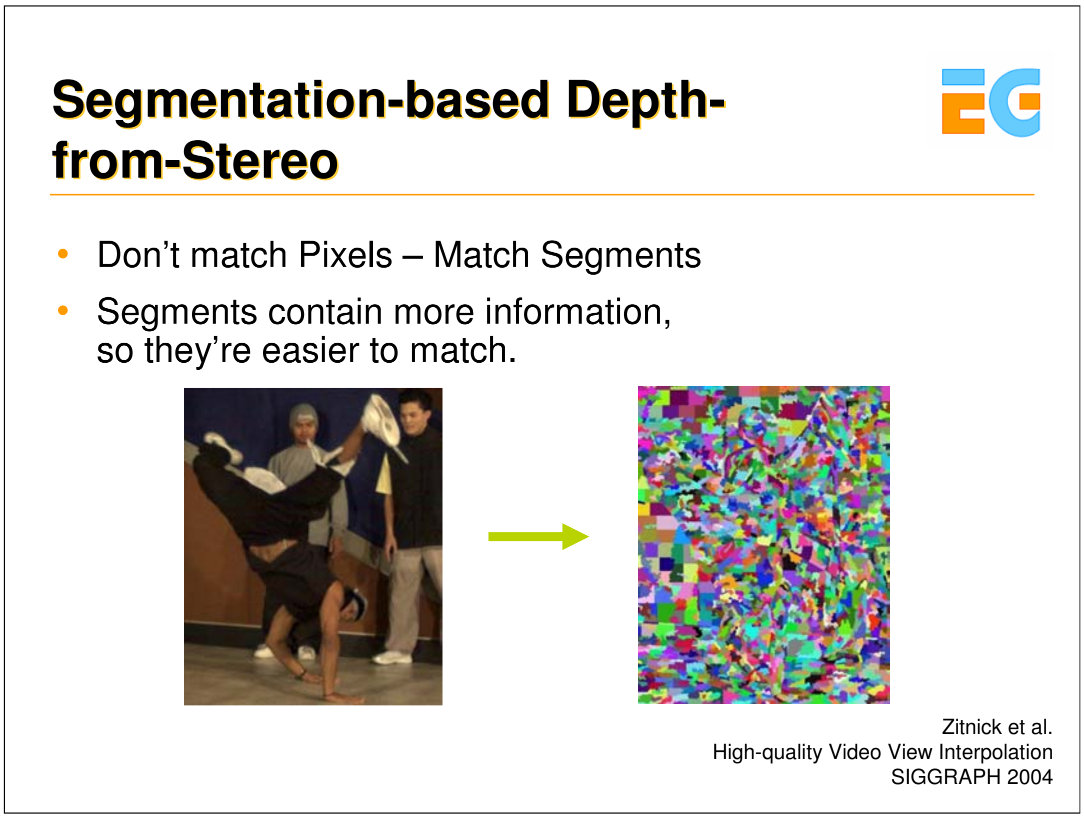

State of the art

Middlebury Stereo Vision Page: http://cat.middlebury.edu/stereo/

State of the art method:
Boykov et al.
Fast Approximate Energy Minimization via Graph Cuts
International Conference on Computer Vision, 1999.

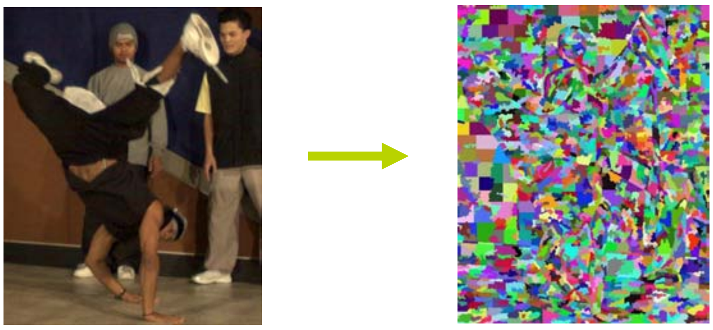
Based on this basic principle researchers developed a multitude of different stereo methods, some of it using the already mentioned window-based correlation methods for better robustness or by applying graph cut based optimization shemes.

The Middlebury Stereo Vision Page contains material for taxonomy and experimental comparison of stereo correspondence algorithms. It contains stereo data sets with ground truth, the overall comparison of algorithms, instructions on how to evaluate stereo algorithms, and stereo correspondence software.



Segmentation-based Depth-from-Stereo

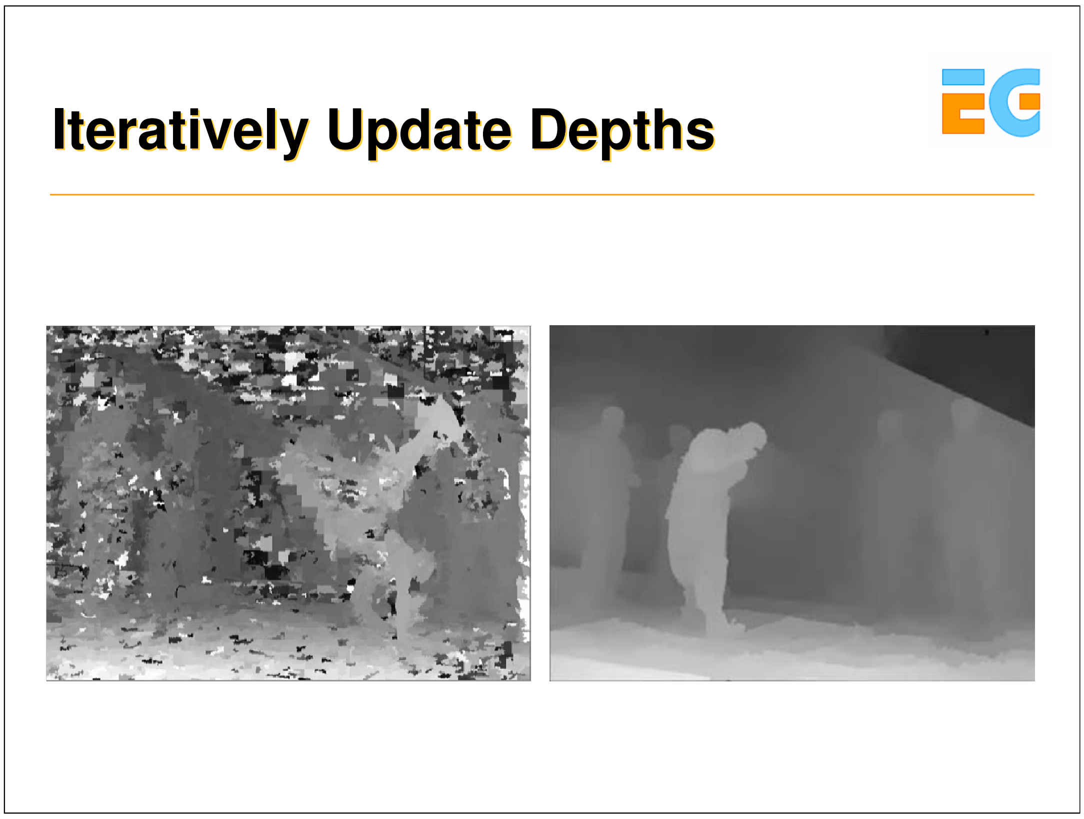
- Don’t match Pixels – Match Segments
- Segments contain more information, so they’re easier to match.



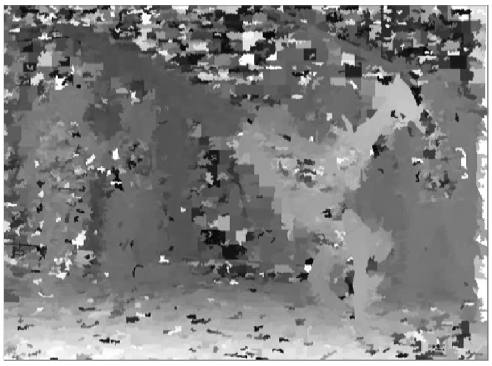
Zitnick et al.
High-quality Video View Interpolation
SIGGRAPH 2004

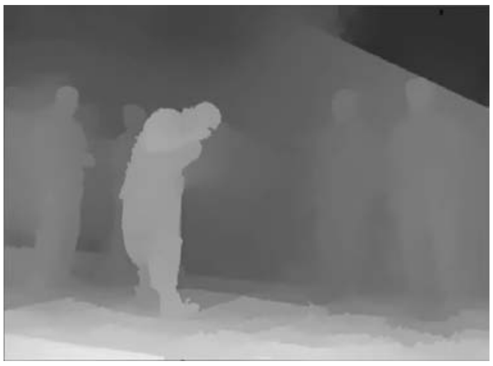
Now the approach by Zitnick et al. was probably one of the first really high quality 3D video systems out there. It combined a novel segmentation-based stereo algorithm with a multi-layered representation which could then interpolate views along a 1D-rail.

Segmentation-based approaches to stereo try to overcome some of the limitations of the pixel-based algorithms. Pixels are inherently hard to match and by correlating entire segments the algorithm produces much better depth maps. However, it relies on the assumption that all pixels of a segment belong to the same surface – so no discontinuities are allowed in the segments. Hence, a over-segmentation has to be produced during a pre-process step.

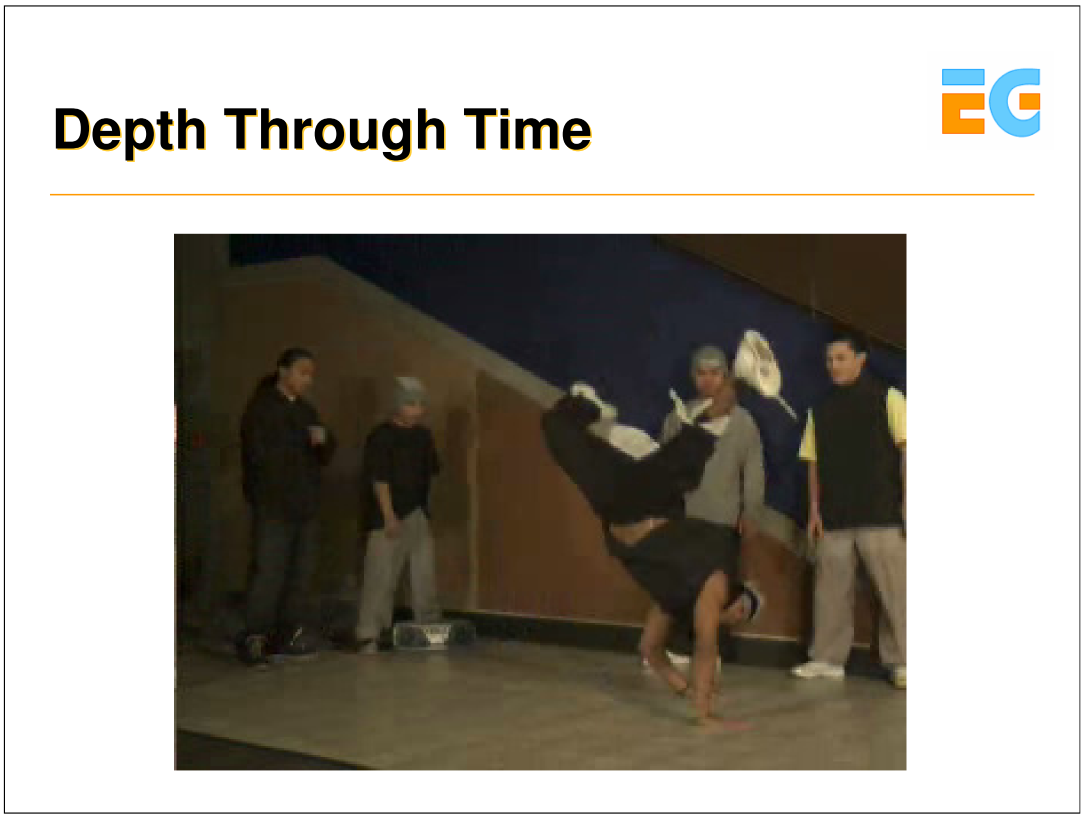


Iteratively Update Depths





Here is an example of that work and how they can iteratively update the depths by taking into account all camera pairs of their system (for course notes).

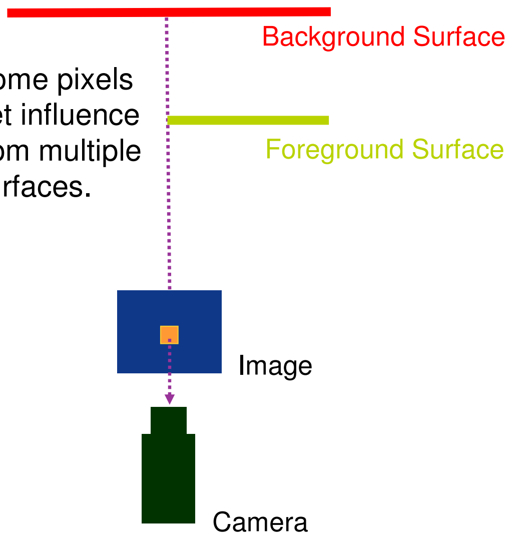


Depth Through Time

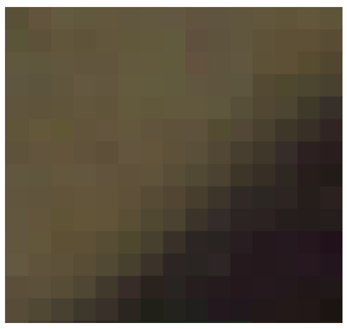
Here is an example of that work and how they can iteratively update the depths by taking into account all camera pairs of their system.

Matting for High-quality Rendering

Some pixels get influence from multiple surfaces.

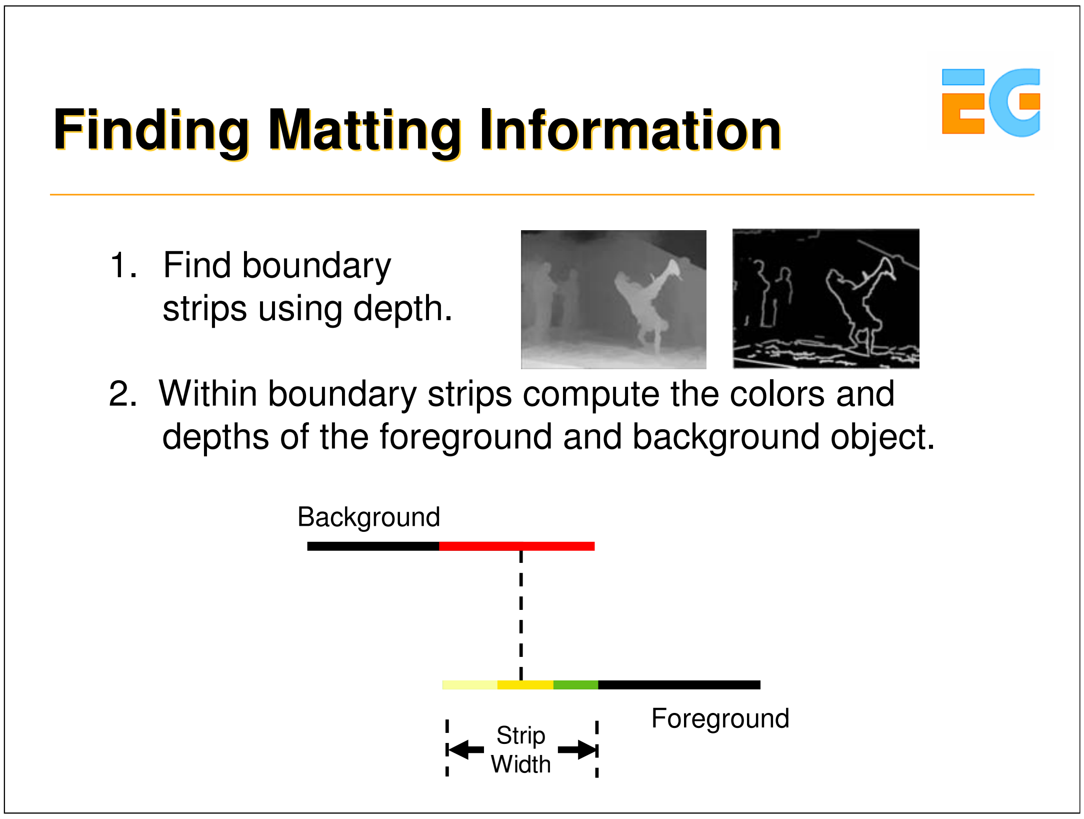


Close up of real image:



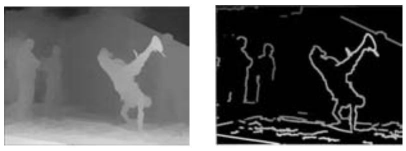
Multiple colors and depths at boundary pixels…

To be able to achieve high-quality re-renderings they apply a novel alpha matting technique. This helps significantly to reduce the artifacts around the depth discontinuities. There, some pixels get influence from multiple surfaces, they are called mixed pixels.

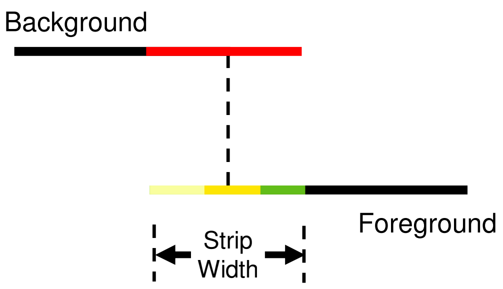


Finding Matting Information

1. Find boundary strips using depth.



2. Within boundary strips compute the colors and depths of the foreground and background object.

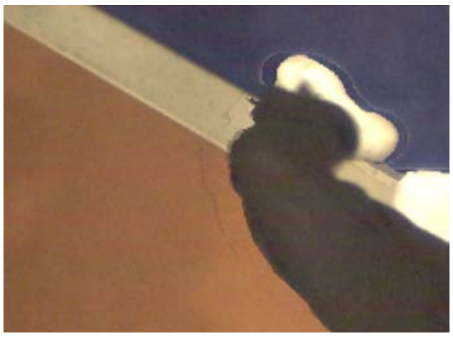


The algorithm first extracts a thin boundary strip around these surfaces which can be easily extracted using the depth information.

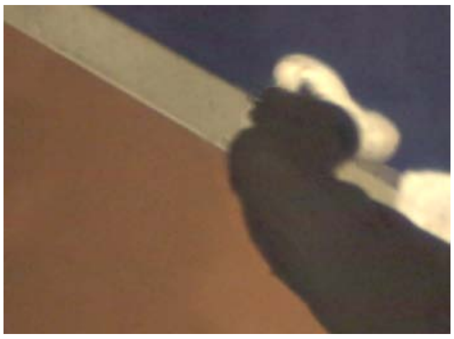
Then, within the boundary strips they compute colors and depth for both the foreground and the background object. In other words they try to separate this mixed pixel.

Why Matting is Important

No Matting

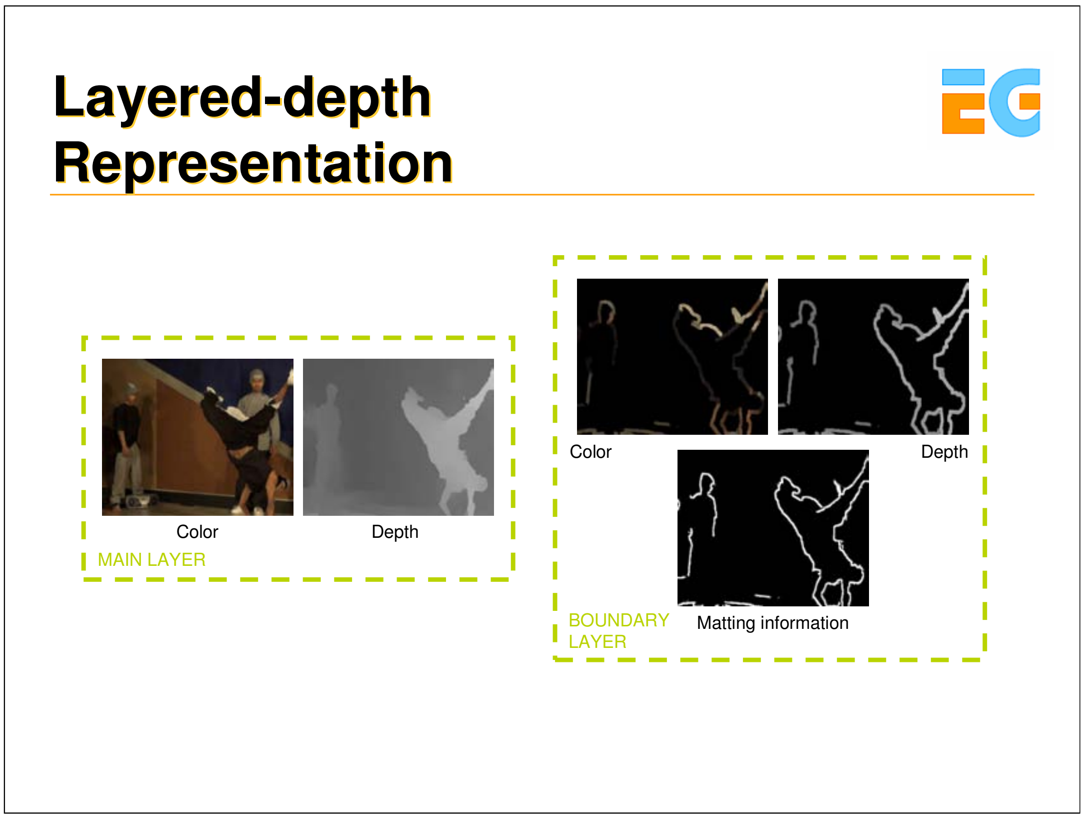


Matting

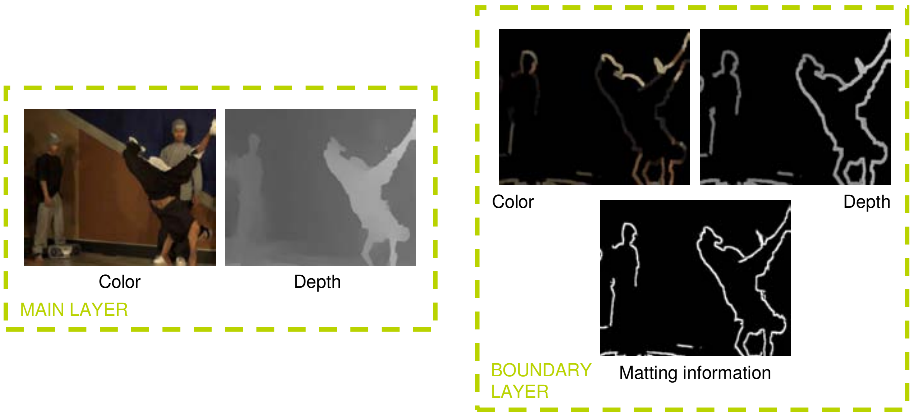


And here you see why matting is so important for high-quality renderings. On the left – without matting – the image has artifacts around the depth discontinuities which is visible as ghosting artifacts.

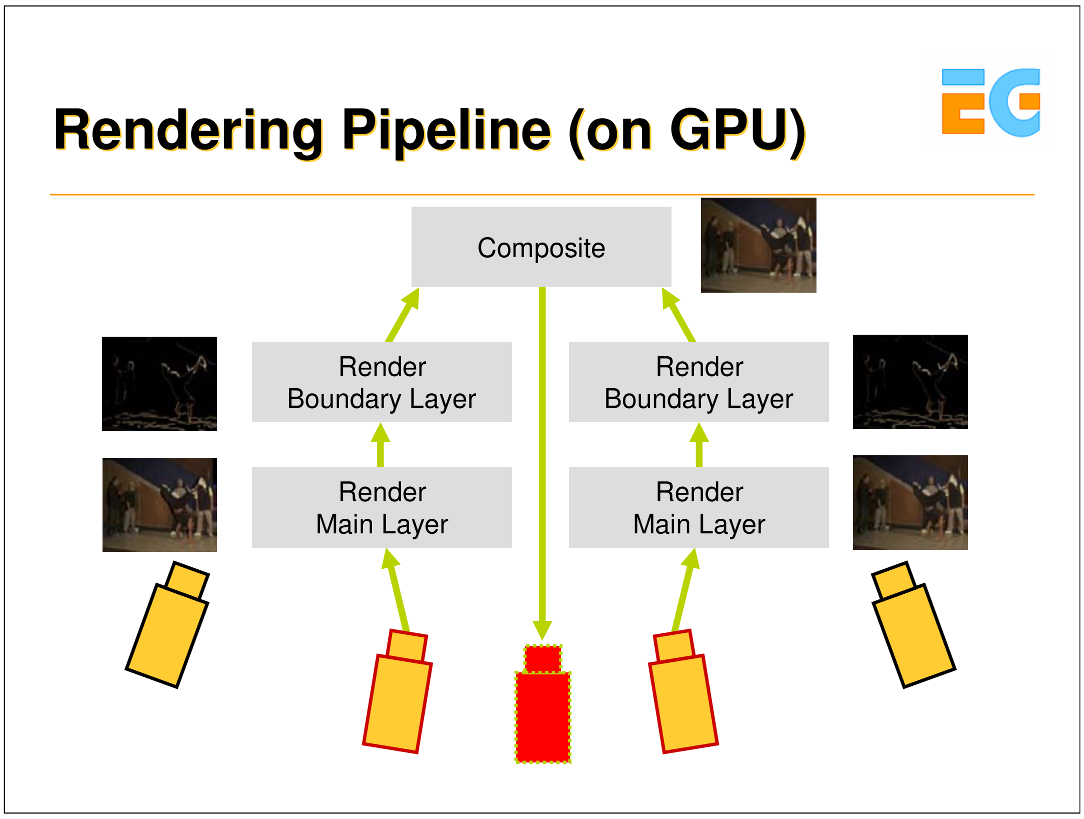
On the right you see how matting can improve the final image quality.



Layered-depth Representation



All this information – color, depth, matting information – is put into a layered-depth representation, one layer holds the main data, and another layer holds the boundary strip information.

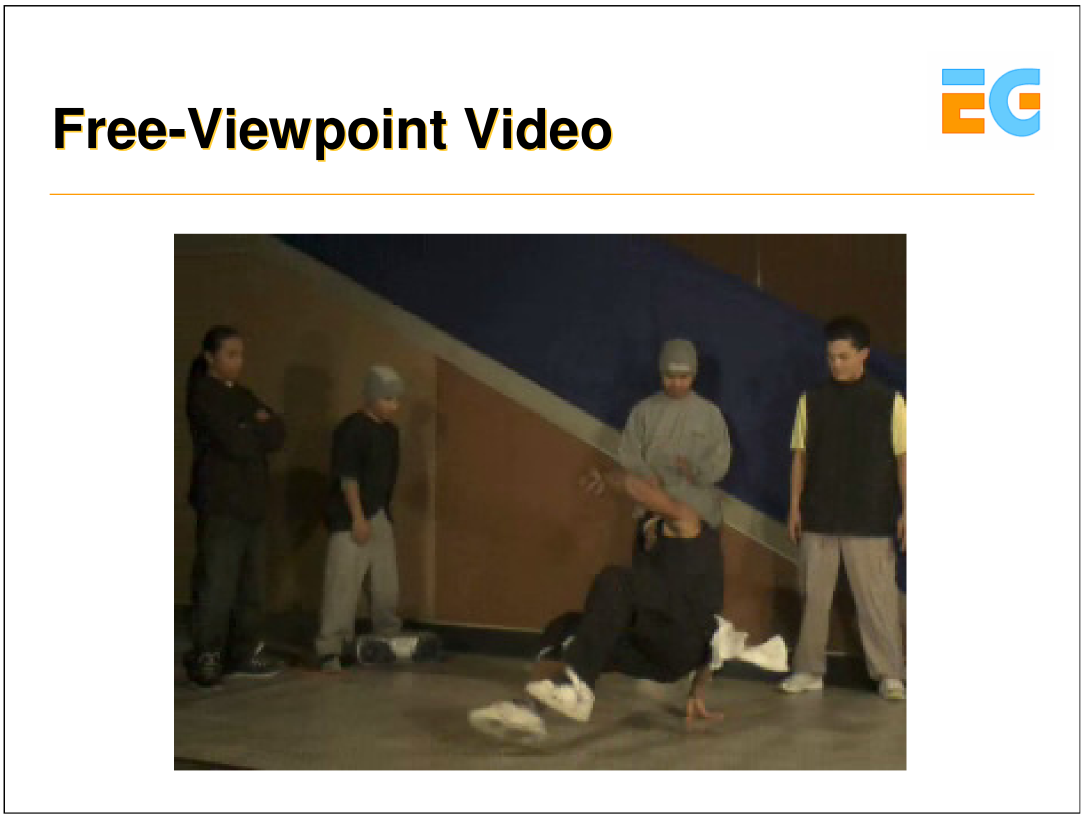


Rendering Pipeline (on GPU)

This can then be rendered efficiently on the GPU by a multi-pass approach and final composite on the fragment level.

Interactive Session Video

Here is an interactive session of this system…



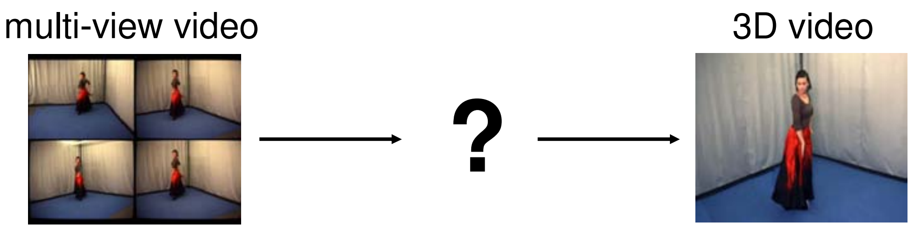
Free-Viewpoint Video

And here is a final result where the system was applied for a commercial music video (for course notes).

1D-Rail “3D” Video

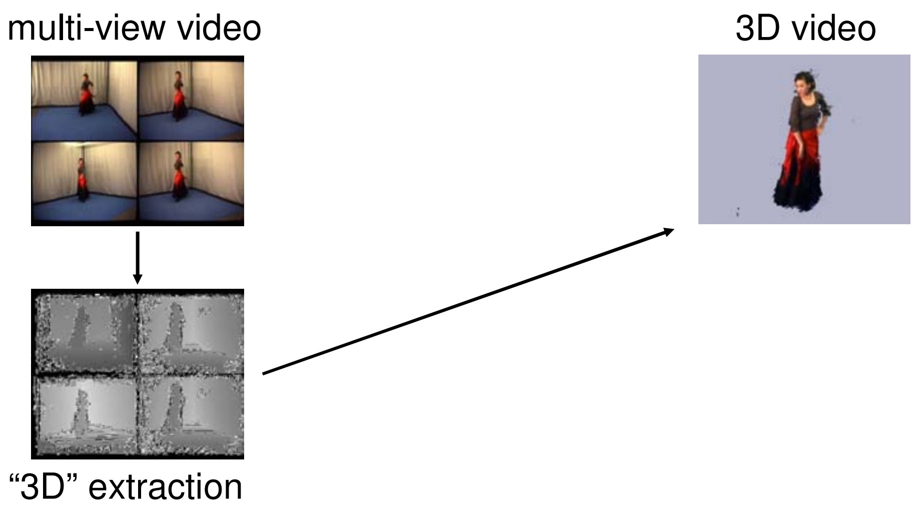
And here is a final result where the system was applied for a commercial music video.

Stereo-based 3D video (Sparse): Hard to Produce!



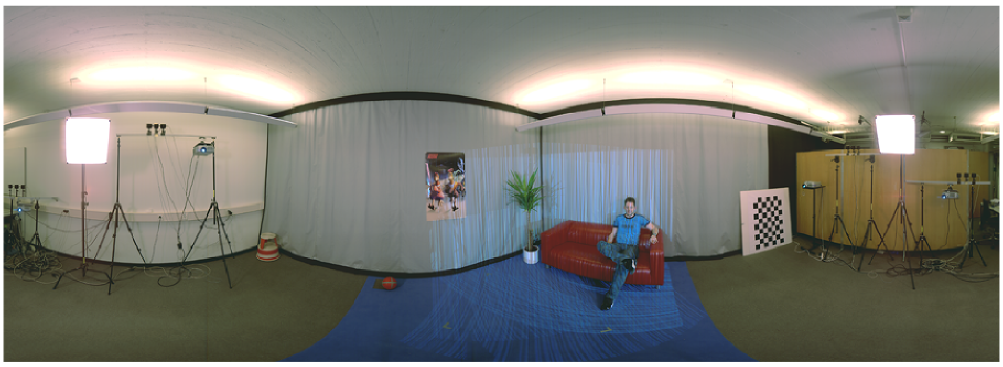
Now if we want to capture scenes not with a dense setup of cameras but instead with a sparse setup, this is even harder to do. Why? The re-rendering has to interpolate even broader views and much more occlusions can occur.

3D Extraction Error Prone

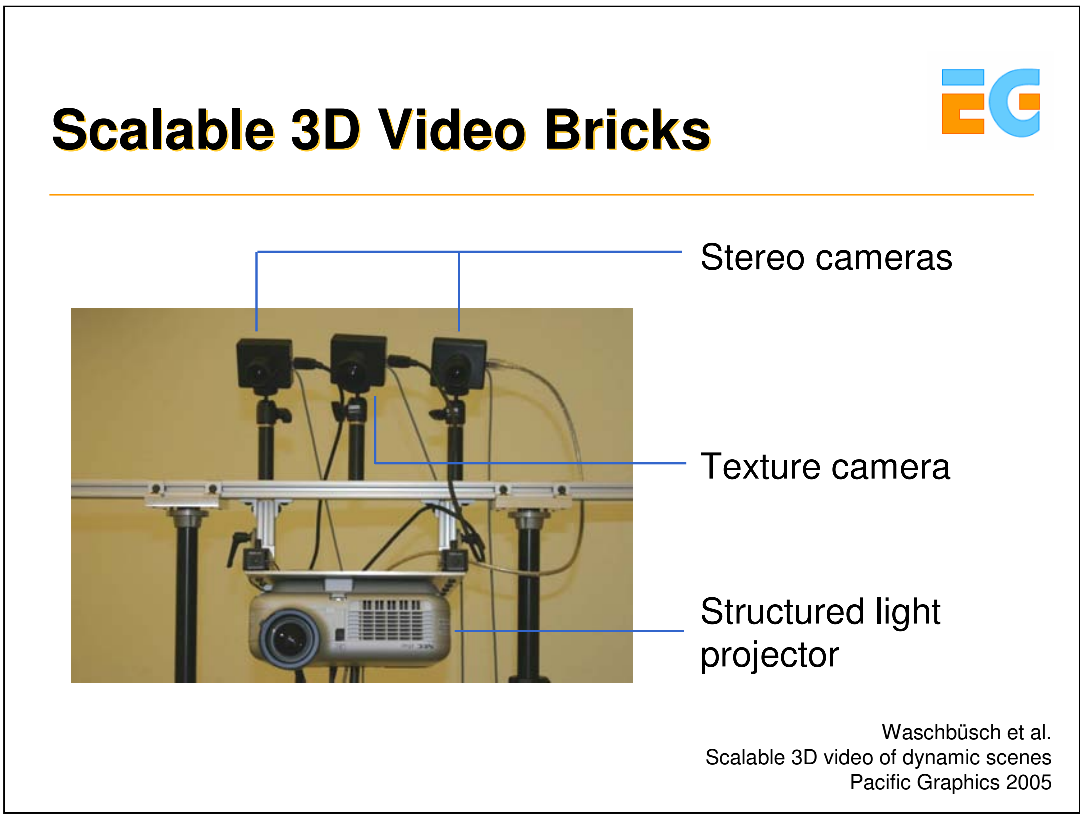


Moreover, the 3D extraction – whether silhouette or depth based – is highly error prone. You see an example in the bottom left and the resulting 3D video in the top right.

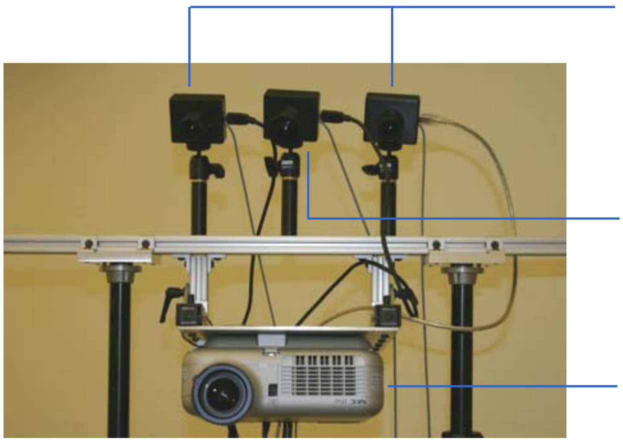
3D Video Studio at ETH Zurich



Here you see a snapshot of the 3D video studio we built at ETH Zurich. The main idea here was to overcome these 3D extraction problems by adding projectors to the studio that help to extract higher quality depth maps. The concept is based on so-called 3D video bricks.



Scalable 3D Video Bricks



Waschbüsch et al.
Scalable 3D video of dynamic scenes
Pacific Graphics 2005

Here you see an image of one such 3D video brick.

Each brick is equipped with one calibrated color camera for acquiring textures.

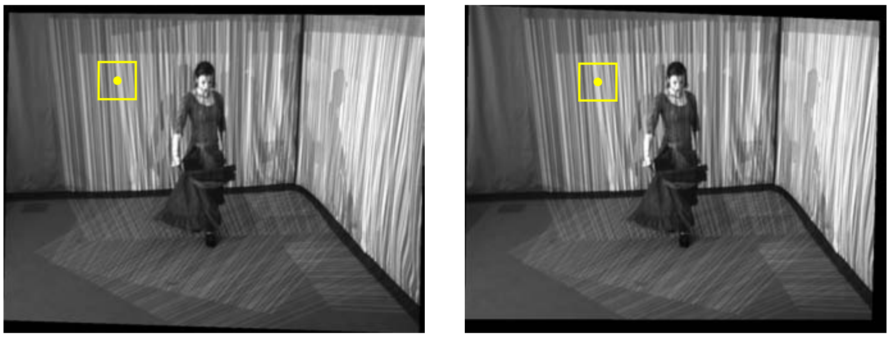
Two calibrated grayscale cameras together with a projector acquire stereo structured-light images used for depth extraction.

The projector additionally serves as an illumination source for the texture camera.

Furthermore, each brick is equipped with one PC for doing the acquisition and depth extraction.

Stereo Vision

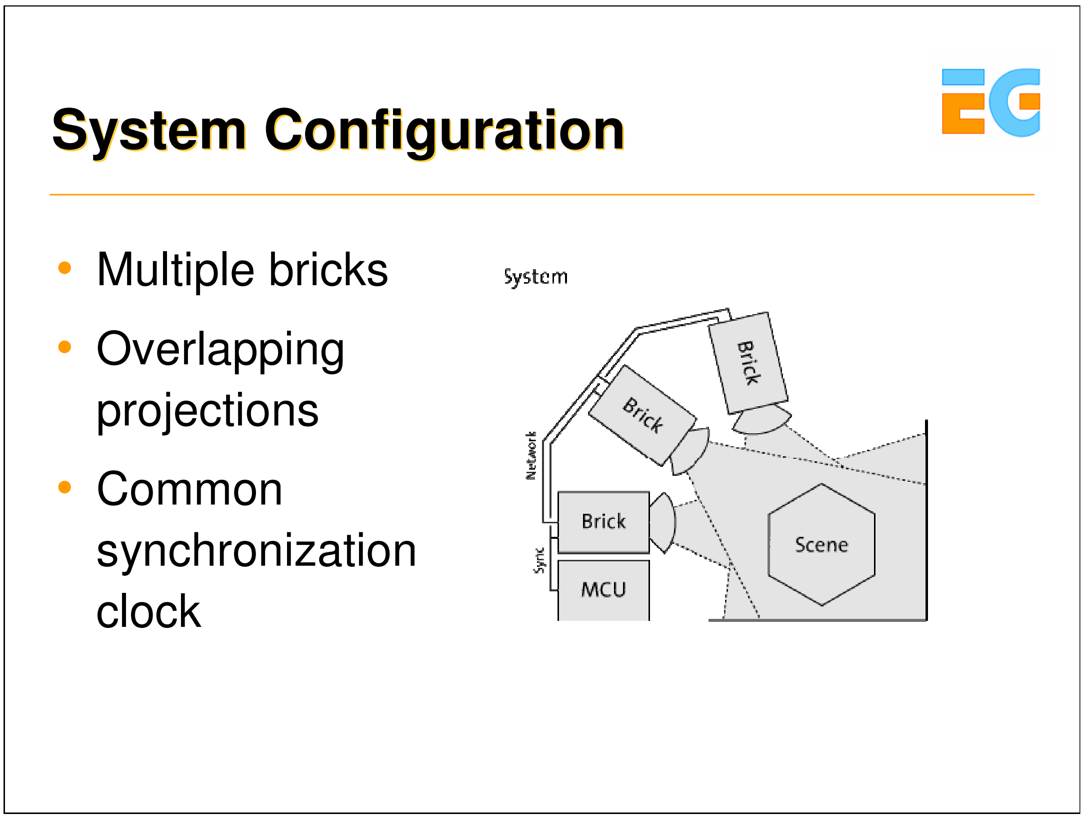
- Structured illumination resolves ambiguities



Discontinuity-preserving stereo on structured light

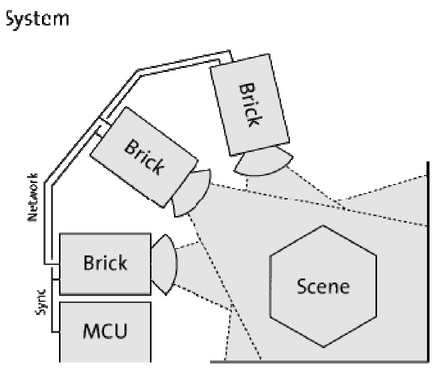
Waschbüsch et al.
Point-Sampled 3D Video of Real-World Scenes
Signal Processing: Image Communication 2007

Why do we want to use projectors in the studio? They can project structured light onto the scene which helps us to resolve ambiguities in regions where there is no texture.

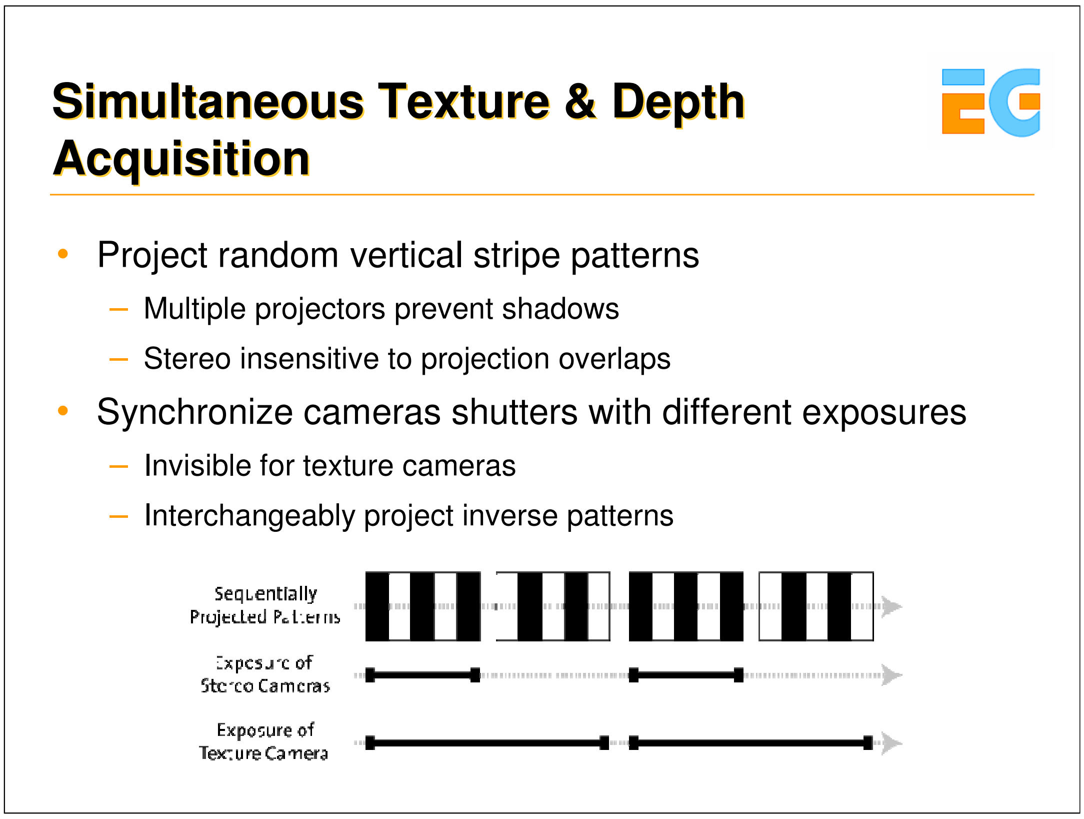


System Configuration

- Multiple bricks
- Overlapping projections
- Common synchronization clock

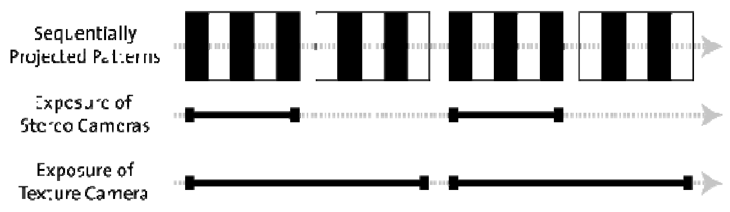


Here you see a system configuration schematic of the 3D video studio with three bricks and overlapping projections. Note that since the structured light is only used to improve the stereo computation, this is no problem.



Simultaneous Texture & Depth Acquisition

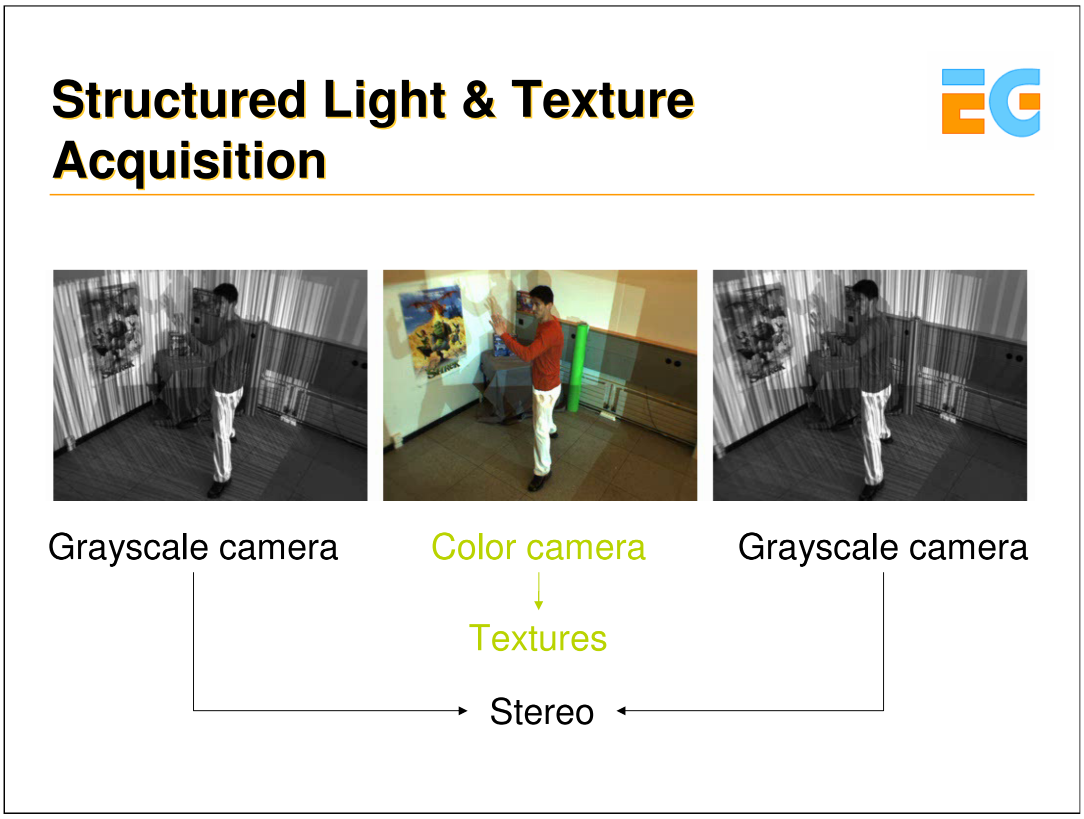
- Project random vertical stripe patterns
  - Multiple projectors prevent shadows
  - Stereo insensitive to projection overlaps
- Synchronize cameras shutters with different exposures
  - Invisible for texture cameras
  - Interchangeably project inverse patterns


Sequentially Projected Patterns
Exposure of Stereo Cameras
Exposure of Texture Camera

To simultaneously acquire texture and depth maps, the structured light patterns should only be visible to the stereo but not to the texture cameras.

We achieve this by interchangeably projecting a pattern and its inverse while exposing the texture cameras to both projections. Thus they acquire the integral image of both patterns which has a uniform white color.

The stereo cameras, in contrast, are only exposed to the first structured light projection.



Structured Light & Texture Acquisition

Grayscale camera

Color camera

Textures

Grayscale camera

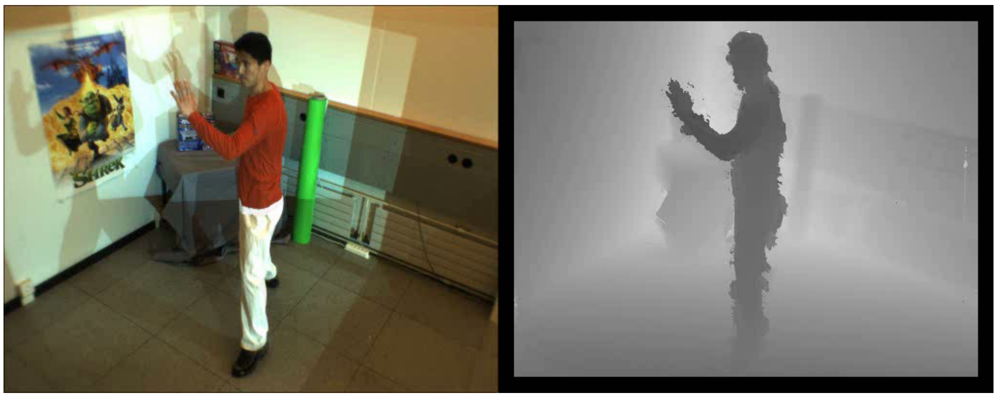
Stereo

Here you see the result of the acquisition.

Notice that the texture camera does not see the projected pattern.

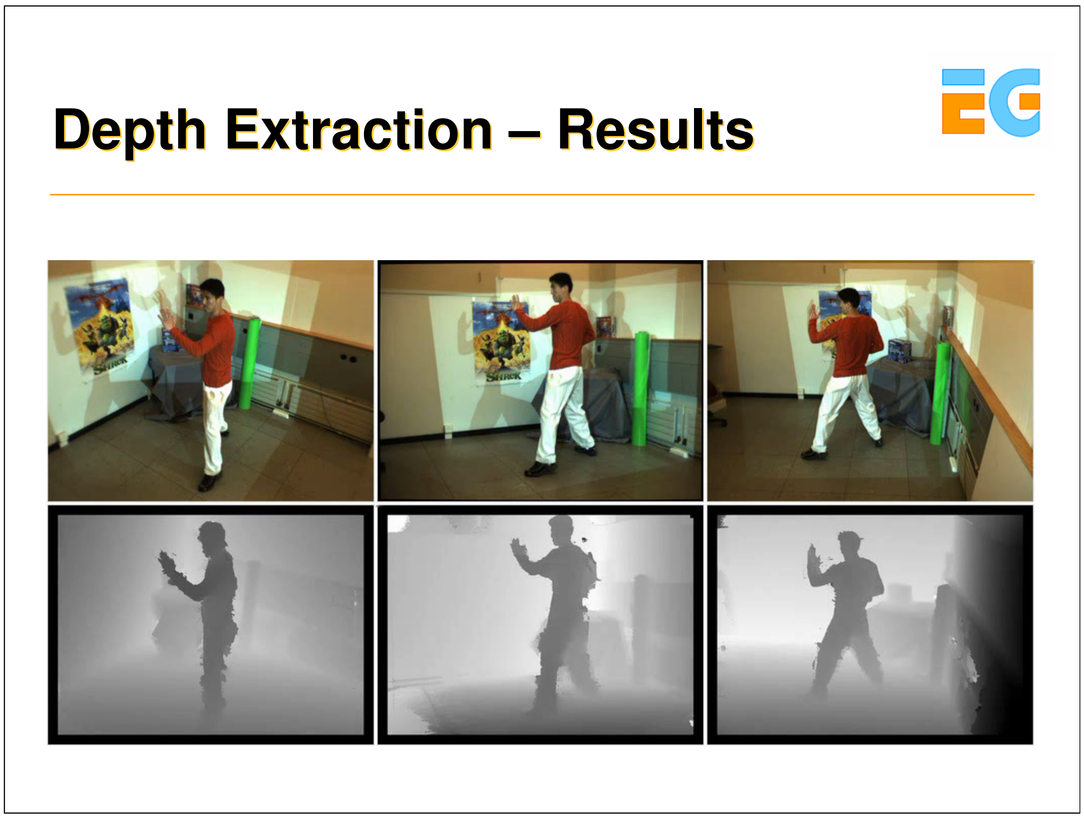
However, the white projector lights are still clearly visible. This was mainly causes by space restrictions of our laboratory which forced us to put the projectors quite close to the scene. Moving the projectors further away or equipping them with wide-angle lenses would cover the whole scene in uniform white light which would be less disturbing.

Depth Extraction

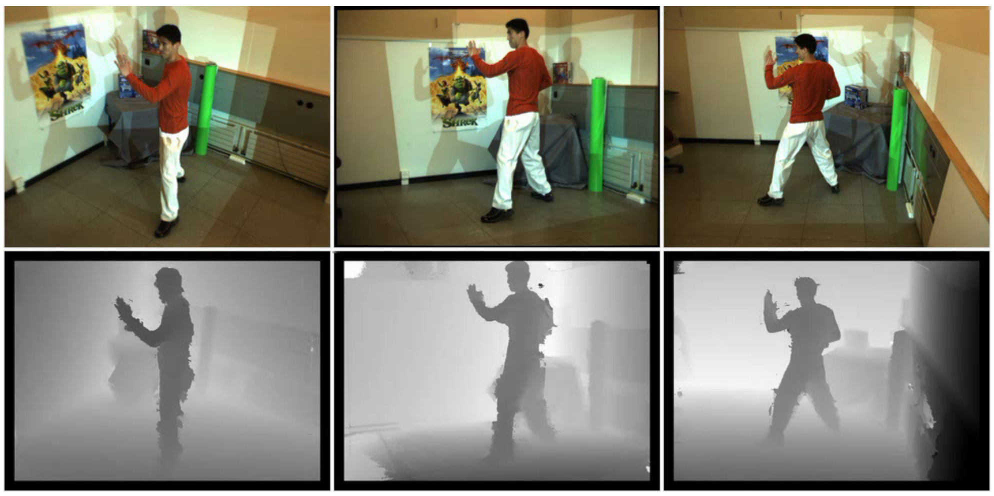


This video shows the final depth maps of the sequence acquired by all three bricks.

We still have some outliers at discontinuities. They are reduced during reconstruction of the 3d model of the scene.

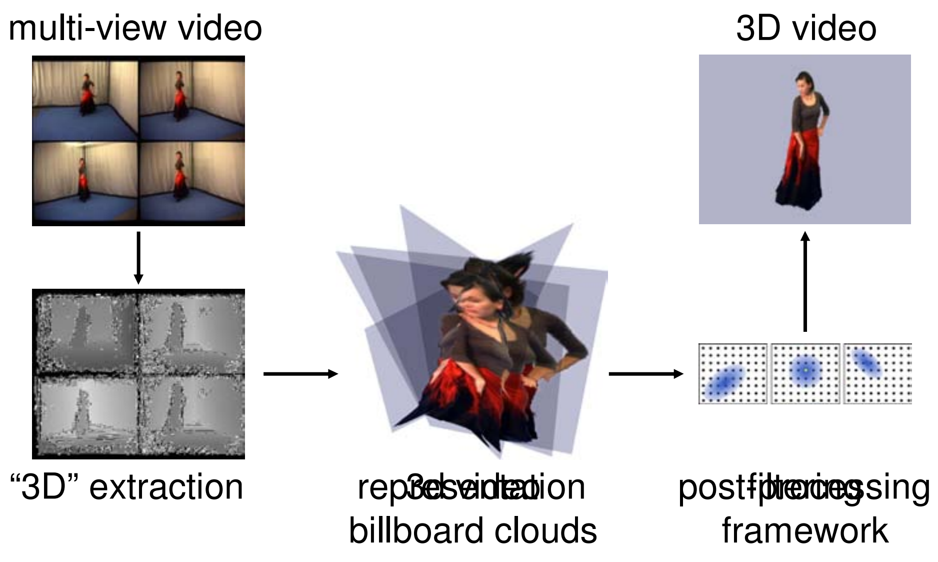


Depth Extraction – Results



And here are the colors and depths of all three bricks used in our original setup.

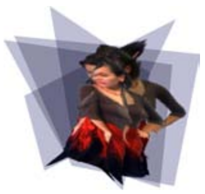
Sparse setups: Need for post-processing!



We will now explain a novel representation which is also introduced here at Eurographics in a talk by Michael Waschbuesch – the 3D video billboard clouds – and how you can exploit this for post-processing the data to achieve high-quality re-renderings.

3D Video Billboard Cloud

- One billboard from each input viewpoint



- Planar geometric proxy

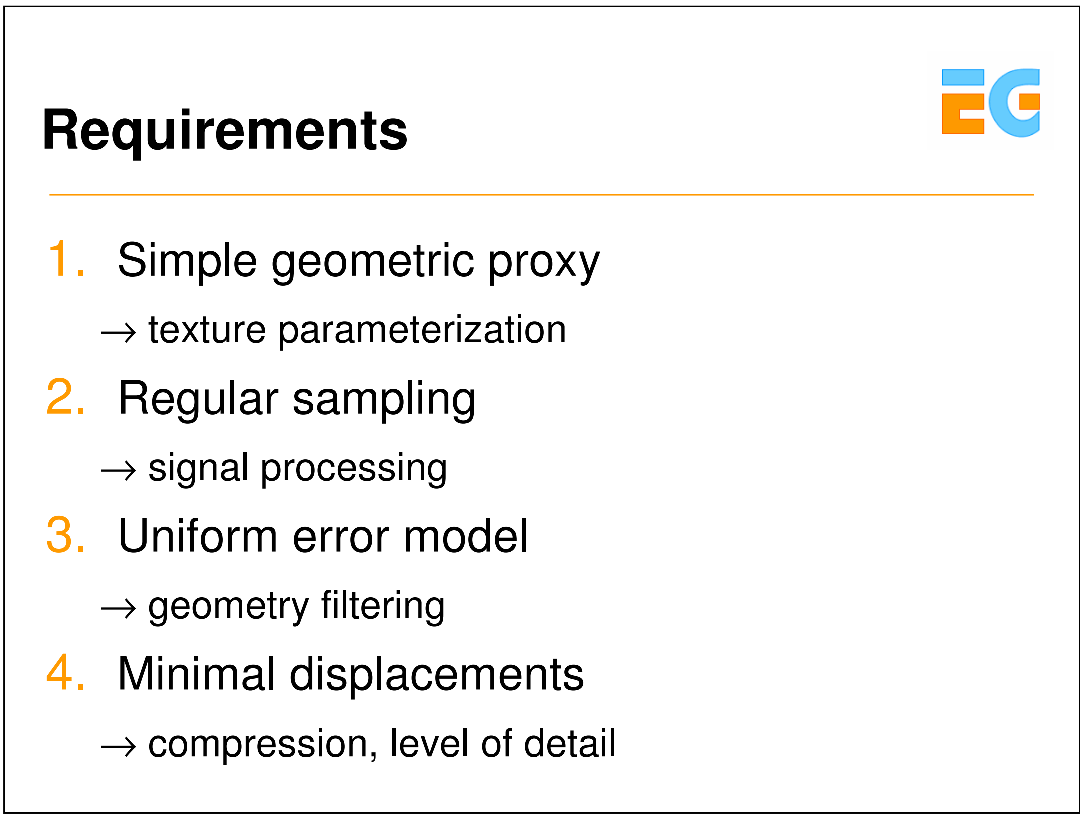
- Displacement map
  Mantler et al., Displacement-mapped Billboard Clouds, TR Vienna Uni. Of Technology, 2007

Waschbüsch et al.
3D video billboard clouds
Eurographics 2007

A 3D video billboard represents the 3D structure and texture of an object at a specific point in time as observed from a single viewpoint.

It consists of an arbitrarily placed and oriented texture-mapped rectangle or proxy approximating the real geometry of the object. Its associated textures are a displacement map for adding fine scale geometric detail, a color map modeling the surface appearance, and an alpha map holding a smooth alpha matte representing the object's boundary.

The latter is employed for seamless blending with the background of the scene.



Requirements

1. Simple geometric proxy
   → texture parameterization
2. Regular sampling
   → signal processing
3. Uniform error model
   → geometry filtering
4. Minimal displacements
   → compression, level of detail

We impose a set of requirements for an optimal 3D video billboard clouds representation:

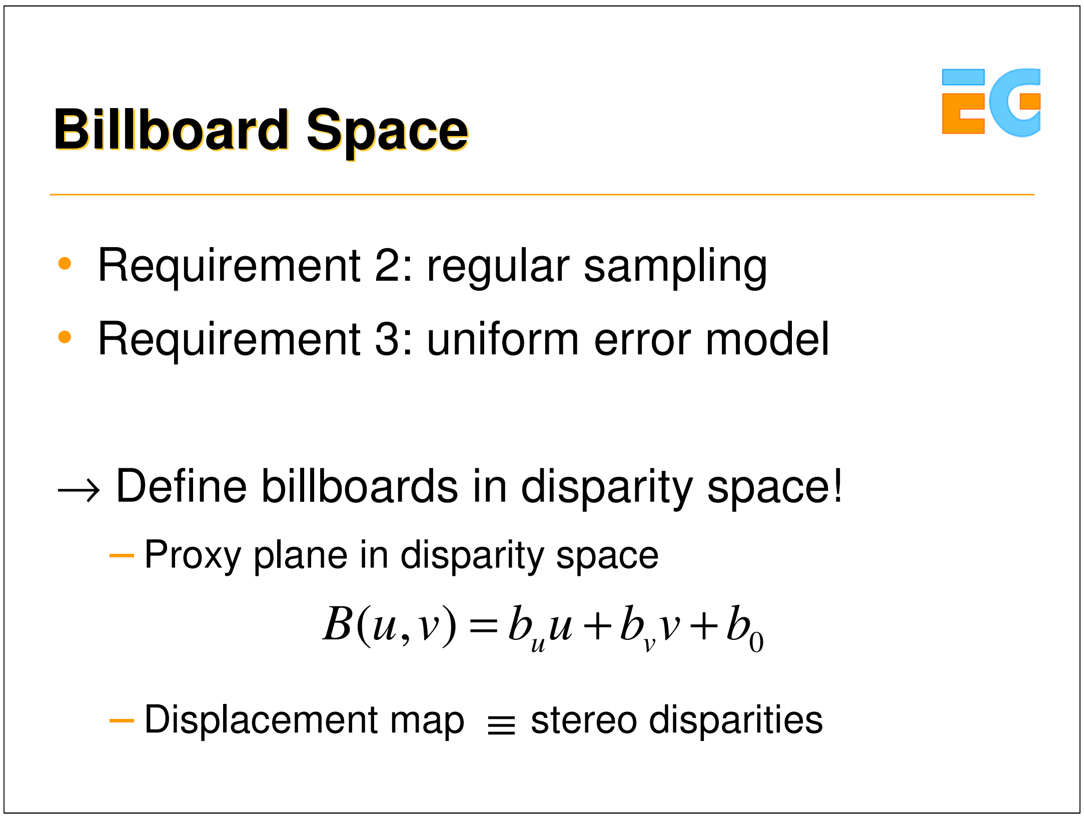
1) Simple geometric proxy. The geometric proxy should be as simple as possible, i.e. a rectangle. This permits an easy parameterization for texture mapping.

2) Regular sampling. By ensuring a regular sampling we can exploit standard signal processing methods for easy post-processing of the geometry without the need of resampling.

In particular, we would like to directly exploit the existing regular sampling from the acquisition cameras.

3) Uniform error model. 3D reconstruction introduces noise which is usually not uniform in world coordinates. The uncertainty of depth values reconstructed by triangulation increases with their absolute value. Our billboard representation should be defined in a space where the reconstruction error can be assumed to be uniform, independent from the distance of a surface from the camera. This allows for easy reconstruction of filters using a uniform, linear kernel for smoothing the acquired geometry.

4) Minimal displacements. A minimal displacement of the proxy to the real surface ensures a good approximation of the geometry and can improve future compression and level-of-detail algorithms.



Billboard Space

- Requirement 2: regular sampling
- Requirement 3: uniform error model

$\rightarrow$ Define billboards in disparity space!

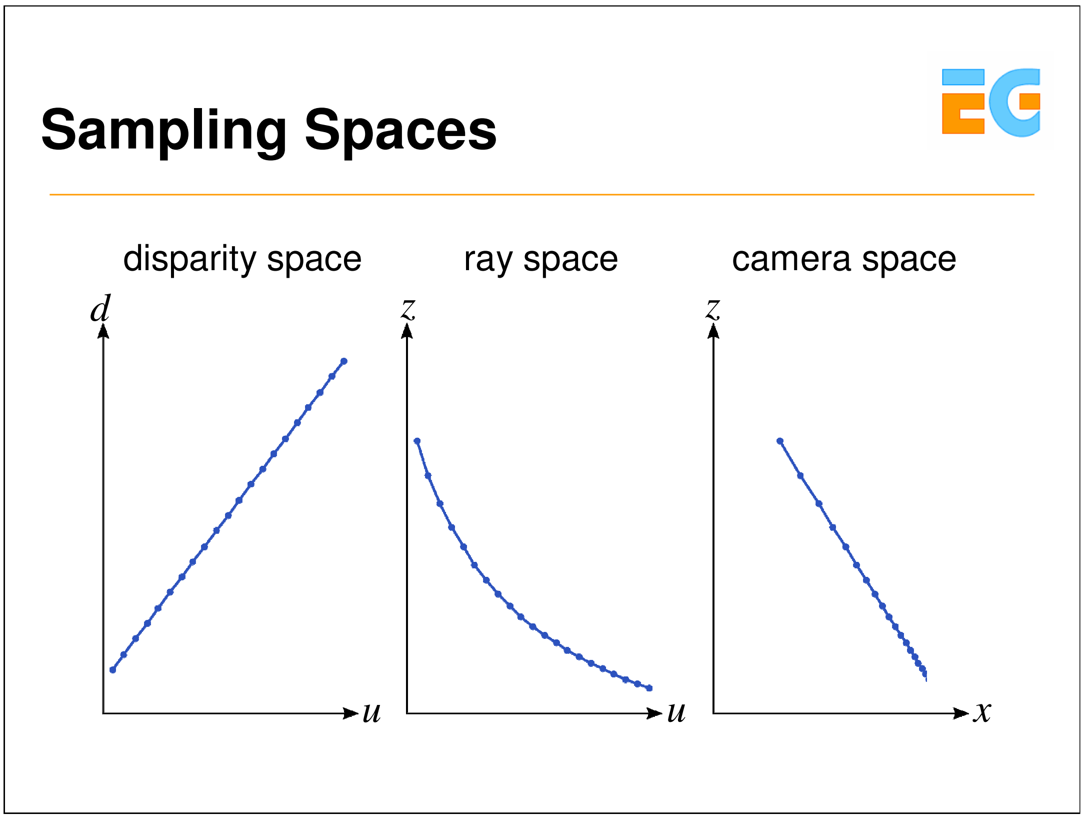
- Proxy plane in disparity space

$$B(u,v) = b_u u + b_v v + b_0$$

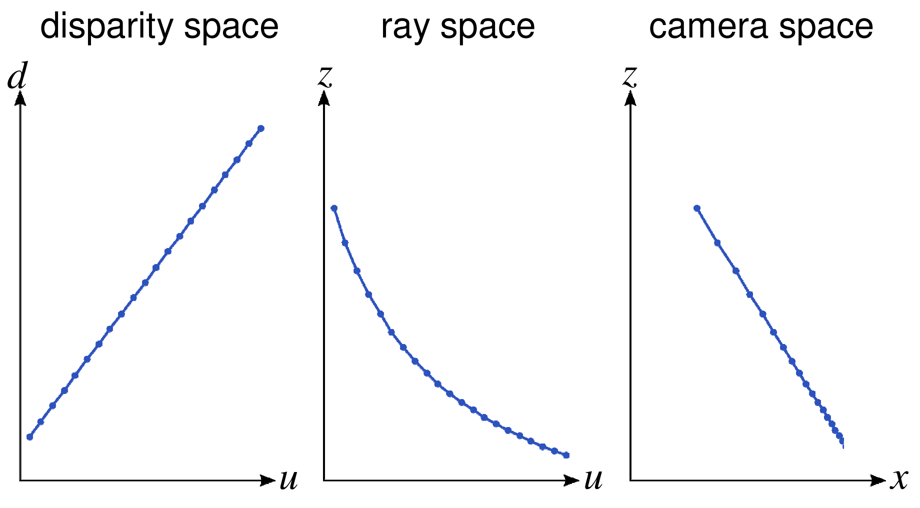
- Displacement map $\equiv$ stereo disparities

Requirement (1) can be guaranteed by definition.

To fulfill requirements (2) and (3) the billboards are not defined in conventional 3D space of the scene but in the so-called disparity space of the acquisition camera. There, the displacements are simply the stereo disparities.



Sampling Spaces

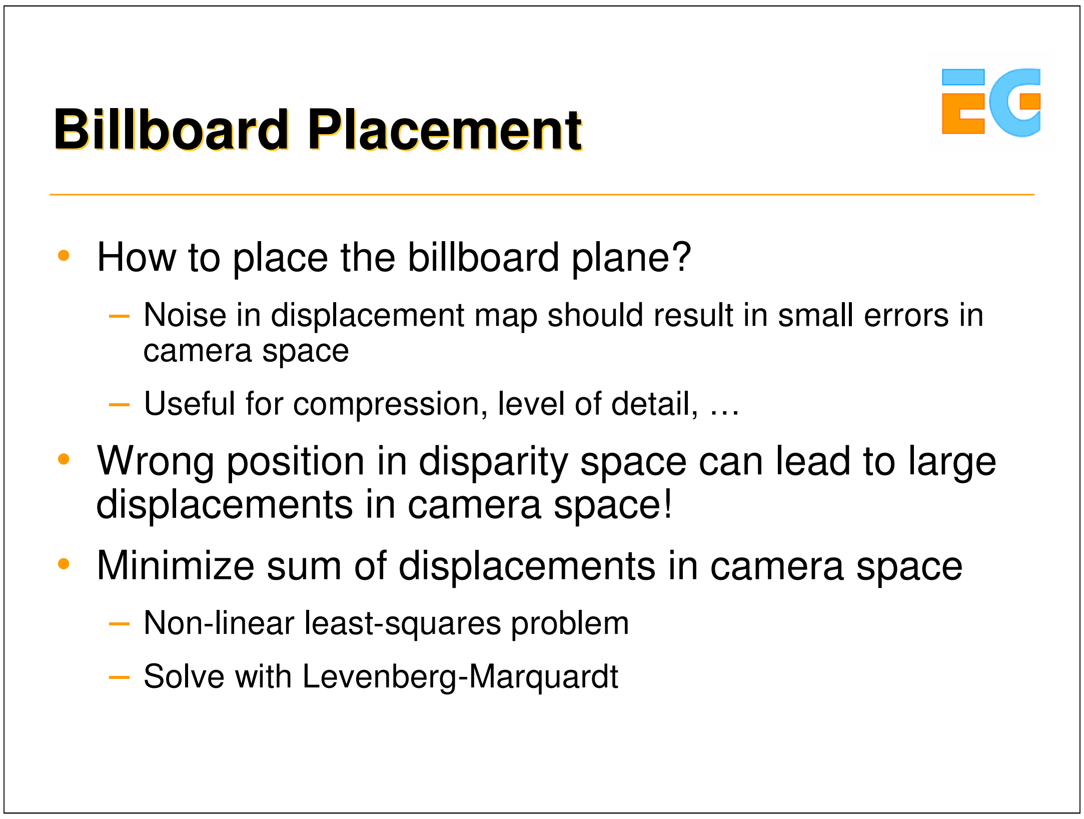


The transformation from camera to ray space is nonlinear, i.e. linear functions in camera space are not linear in ray space anymore. Hence, if we would define the billboard plane in ray space and use the depth values as displacements, it would not be planar in world coordinates and thus it would be difficult to use it as an approximation for the real geometry. On the other hand, if we would place it in camera space, the sampling would become irregular.

Instead, we define a disparity space of a camera as coordinates (ui, vi, zi') with zi' = 1/zi . If we use this representation and store the reciprocal of the z-coordinate from ray space, we can observe that planes in disparity space stay planar in
camera space.

Moreover, sampling in disparity space is identical to the regular sampling of the acquisition cameras. Thus, requirement (2) is fulfilled if we define the billboard planes in these coordinates.

In camera space it can be observed that the resulting uncertainty of the geometry is not constant anymore but depending on the absolute value of the disparity.



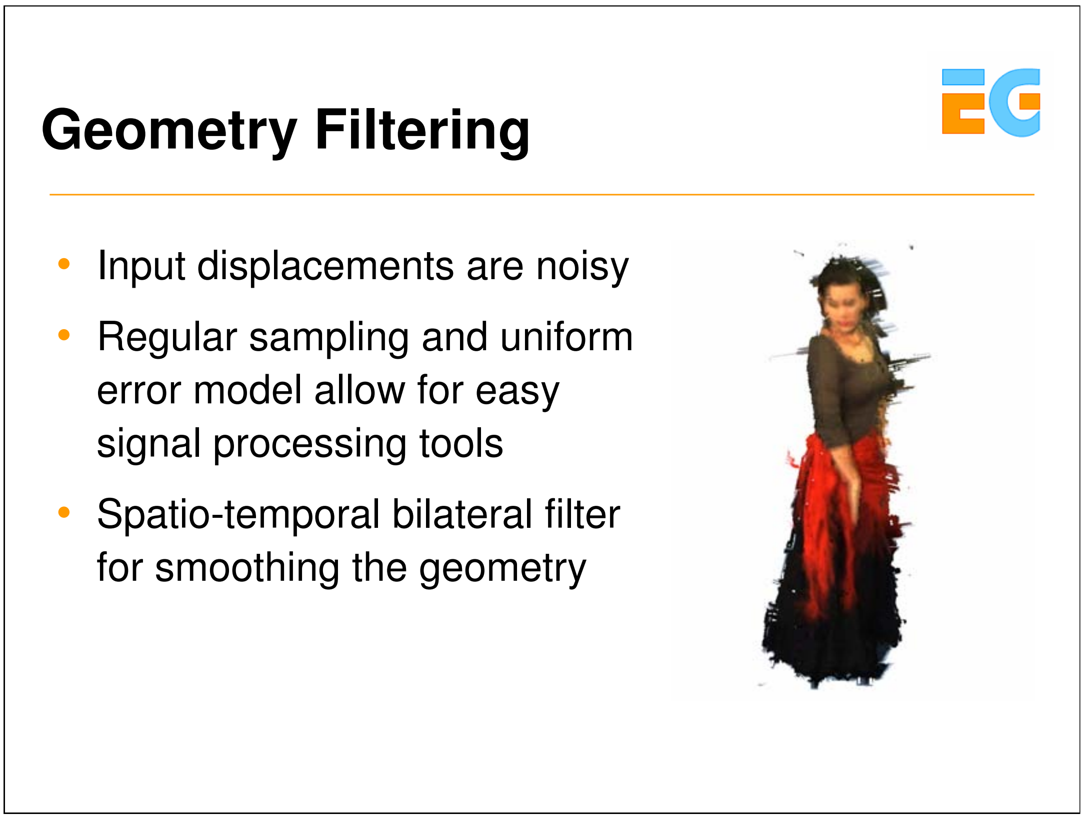
Billboard Placement

- How to place the billboard plane?
  - Noise in displacement map should result in small errors in camera space
  - Useful for compression, level of detail, …
- Wrong position in disparity space can lead to large displacements in camera space!
- Minimize sum of displacements in camera space
  - Non-linear least-squares problem
  - Solve with Levenberg-Marquardt

We are still free to choose the position and orientation of the billboard plane. A bad choice of these values can lead to arbitrarily large displacements in world coordinates. This becomes an important issue as soon as the values of the displacement map should be processed, e.g. for compression, level of detail, …

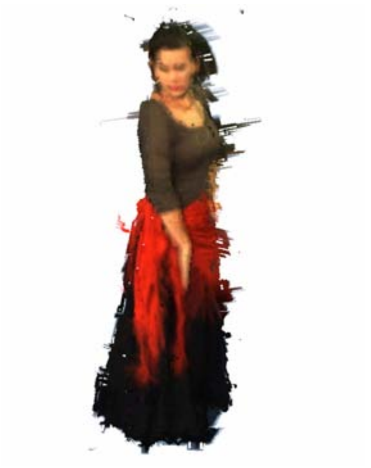
Hence, noise in displacement map should result in small errors in camera space

We minimize the sum of displacements in camera space – non-linear least-squares problem – and solve it with Levenberg-Marquardt algorithm.



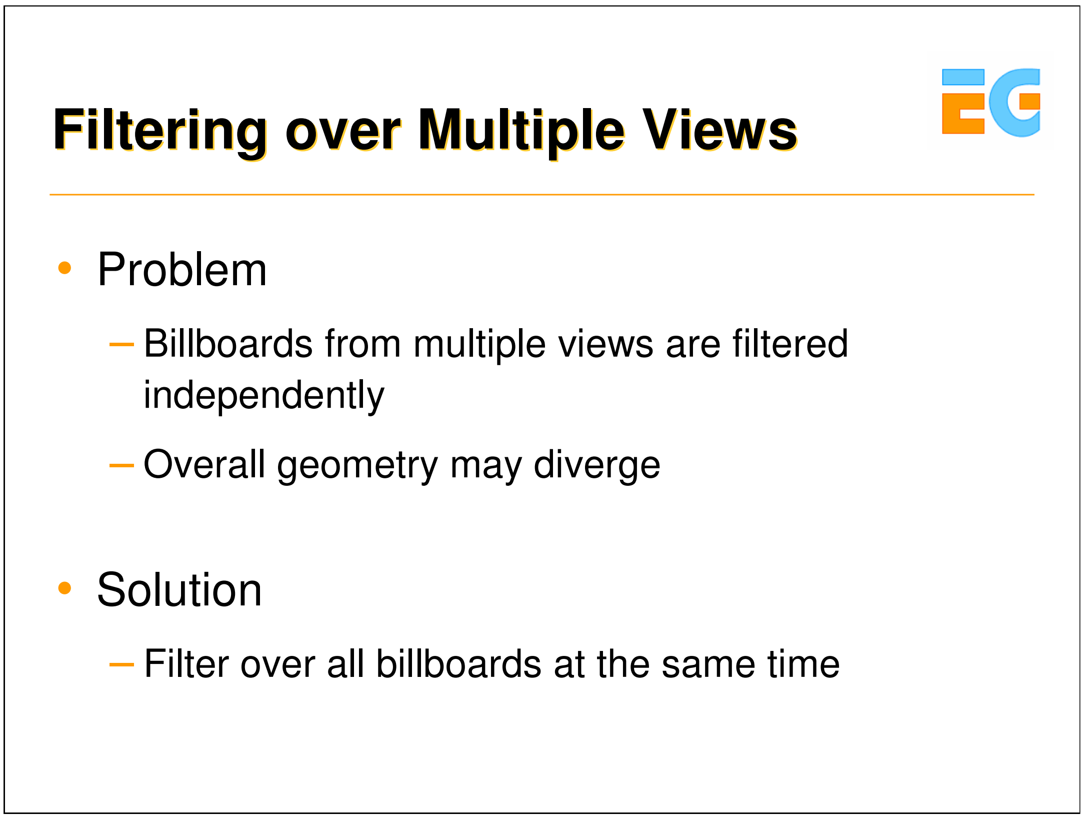
Geometry Filtering

- Input displacements are noisy
- Regular sampling and uniform error model allow for easy signal processing tools
- Spatio-temporal bilateral filter for smoothing the geometry



The displacement values generated by the acquisition system are subject to quantization errors, noise, and calibration inaccuracies, resulting in several kinds of artifacts in the re-rendered image: The object surfaces do not appear smooth and their geometry is very noisy, which is especially visible as flickering over time. Moreover, overlapping parts of surfaces from different scanning directions do not necessarily fit to each other.

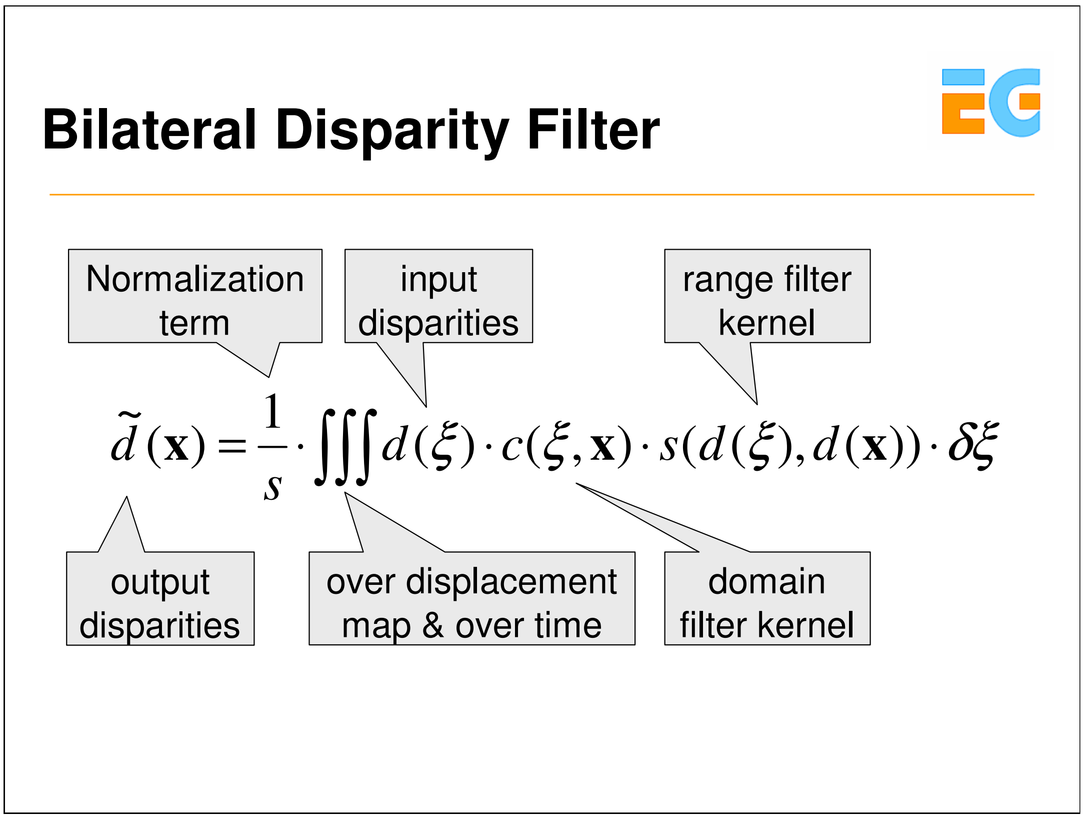
To improve this, we apply a four-dimensional smoothing filter yielding better spatial coherence within surfaces and between overlapping surfaces, and better coherence over time.



Filtering over Multiple Views

- Problem
  - Billboards from multiple views are filtered independently
  - Overall geometry may diverge
- Solution
  - Filter over all billboards at the same time

However, when filtering each acquisition view independently, corresponding surfaces reconstructed from different views would not fit to each other. Thus, we extend the filter by an additional domain by accumulating the data from all acquisition views.



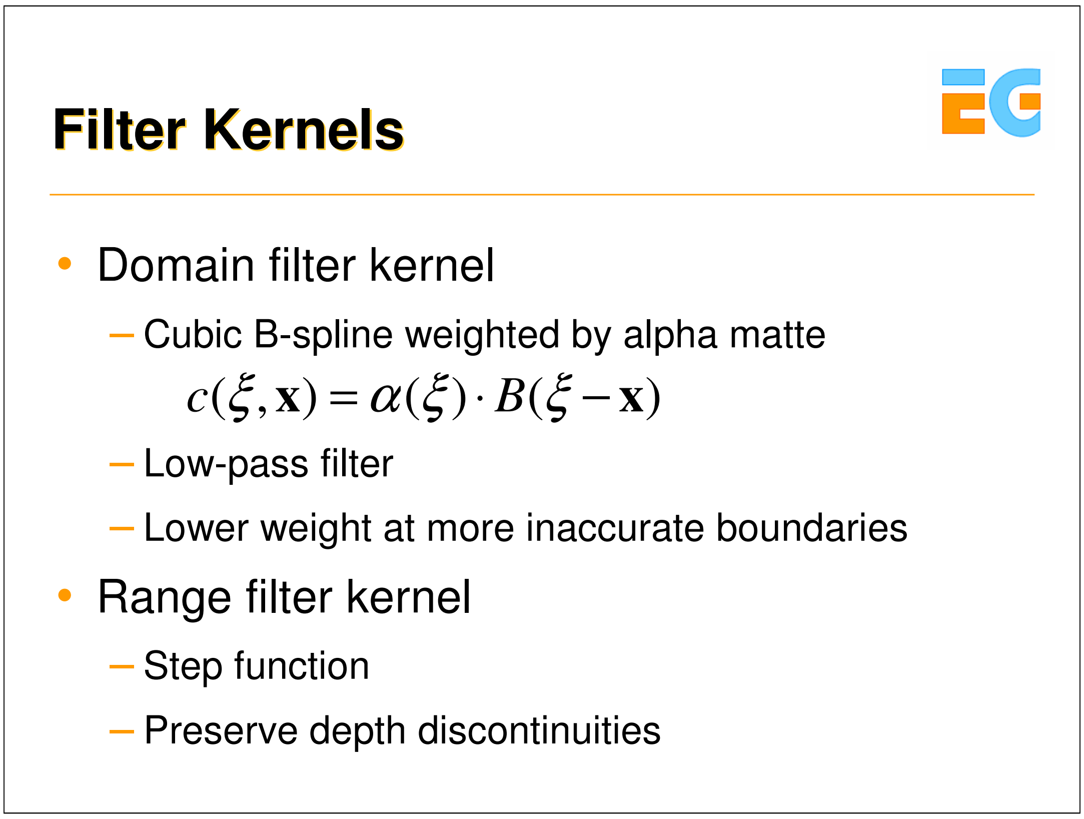
Bilateral Disparity Filter

$$\tilde{d}(\mathbf{x}) = \frac{1}{s} \cdot \iiint d(\xi) \cdot c(\xi, \mathbf{x}) \cdot s(d(\xi), d(\mathbf{x})) \cdot \delta\xi$$

Normalization term
input disparities
range filter kernel
output disparities
over displacement map & over time
domain filter kernel

All this is now put together into this formula, where you see how a bilateral filter is applied over the displacement map and over time.

Bilateral filter: The domain filter kernel c smoothes the disparities over space and time while the range filter kernel s tries to retain geometric discontinuities



Filter Kernels

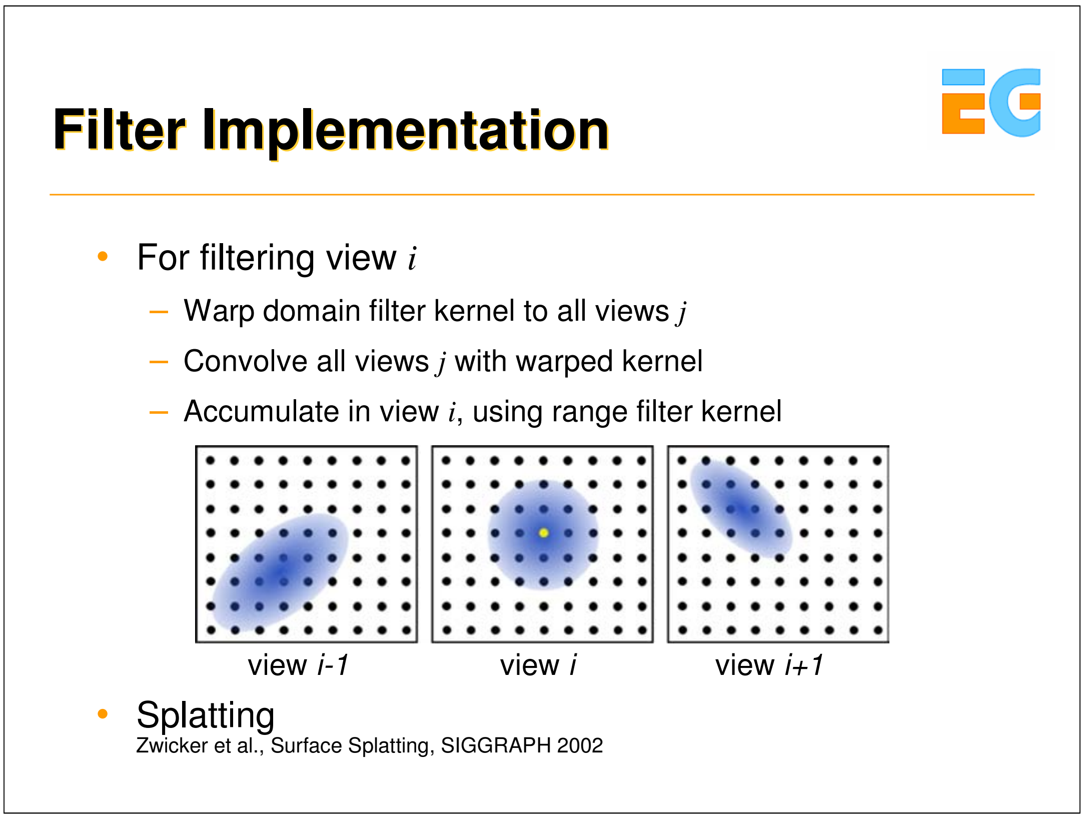
- Domain filter kernel
  - Cubic B-spline weighted by alpha matte

$$c(\xi, \mathbf{x}) = \alpha(\xi) \cdot B(\xi - \mathbf{x})$$

  - Low-pass filter
  - Lower weight at more inaccurate boundaries
- Range filter kernel
  - Step function
  - Preserve depth discontinuities

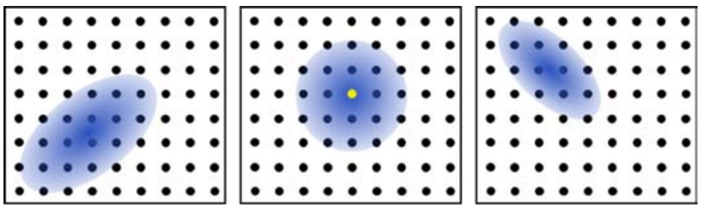
For c we use a cubic b-spline low-pass filter kernel B weighted by the alpha values of the current billboard.

The range filter kernel s does not only maintain discontinuities in the disparity maps but also ensures a correct handling of occlusions that occur during warping. We use a simple step function.



Filter Implementation

- For filtering view *i*
  - Warp domain filter kernel to all views *j*
  - Convolve all views *j* with warped kernel
  - Accumulate in view *i*, using range filter kernel

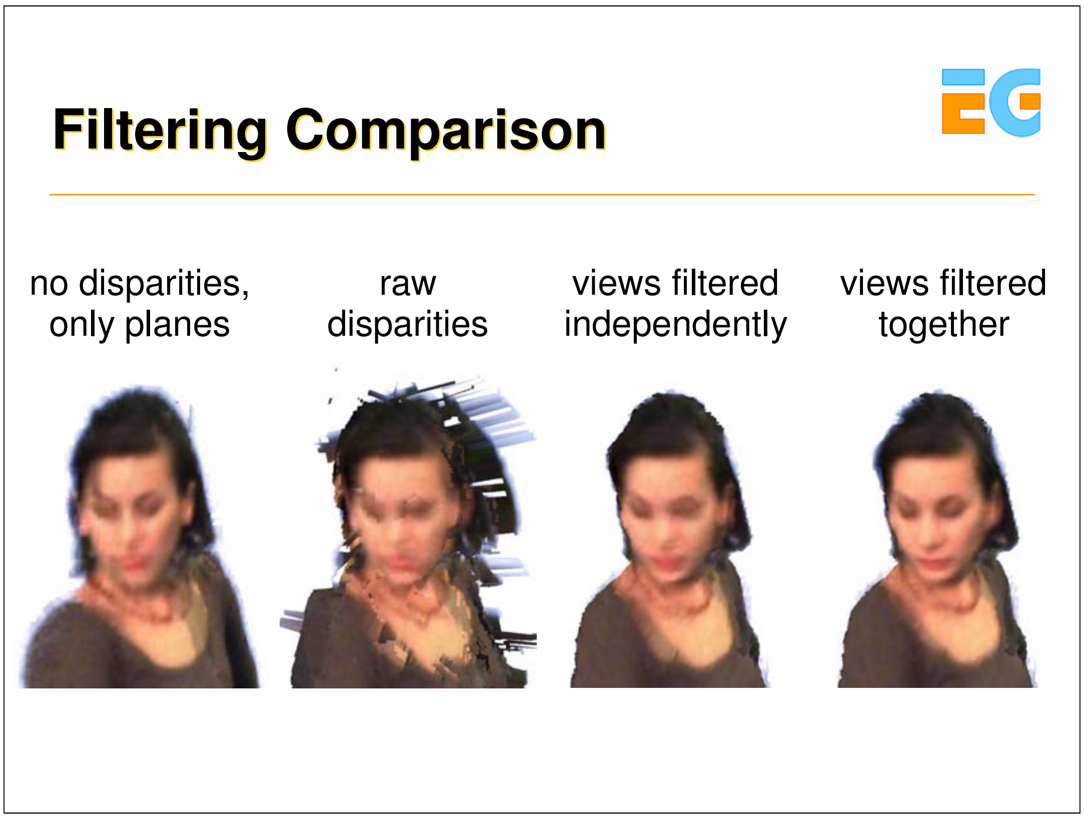


- Splatting
  Zwicker et al., Surface Splatting, SIGGRAPH 2002

Intuitively, for computing a disparity d˜i, this filter does not only compute a convolution in the current view i but it convolves all views with warped versions of the domain filter kernel c and accumulates all values weighted by the range filter kernel s.

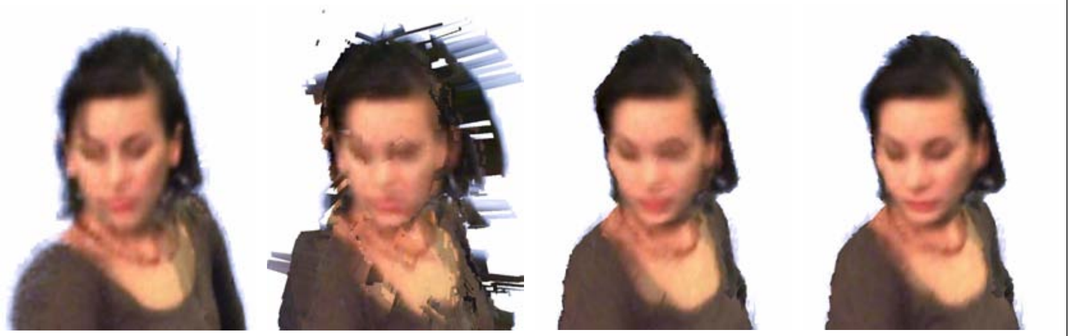
[Image: For computing the value of the yellow pixel in view i, the values all pixels in all views weighted by the warped domain filter kernel (blue) are accumulated.]

The implementation of the filter process is done via splatting. For filtering a view i, instead of projecting the filter kernel into all other views, we do the inverse and splat all views into view i. This has the advantage that we can use a uniform splatting kernel. Splatting can be performed efficiently in the GPU.

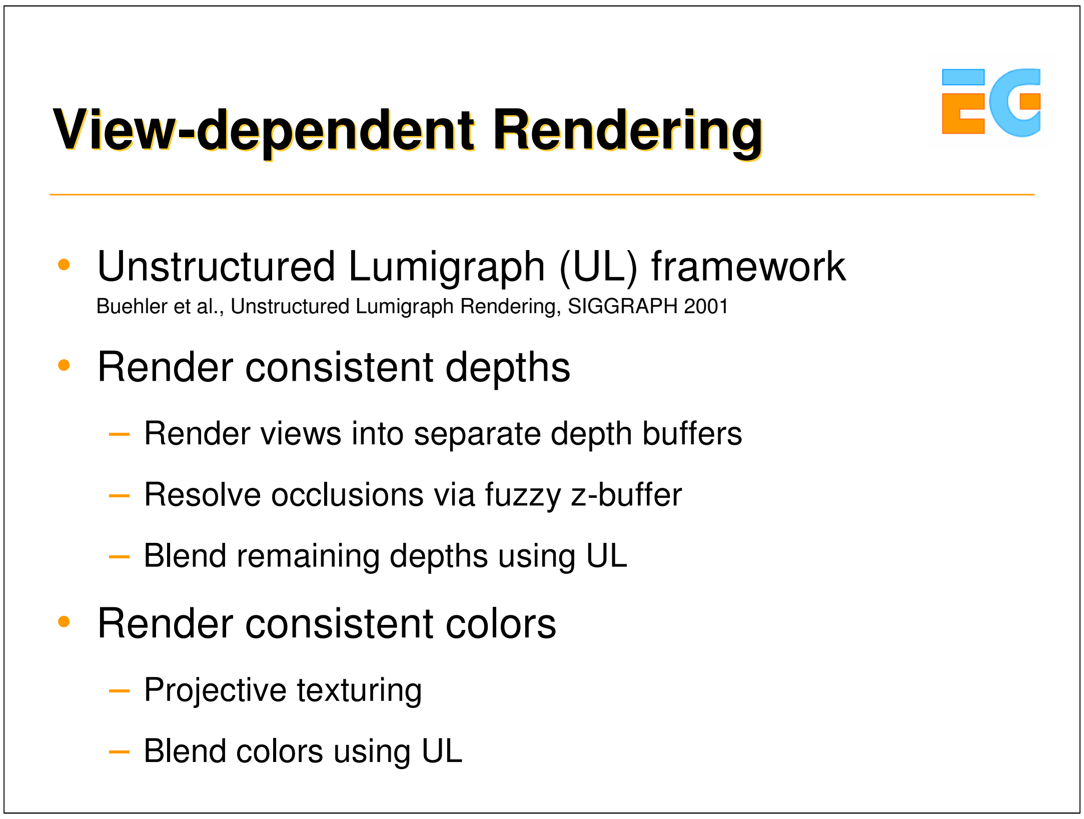


Filtering Comparison

no disparities, only planes | raw disparities | views filtered independently | views filtered together



Here is a comparison of different filtering techniques where you clearly see the difference.



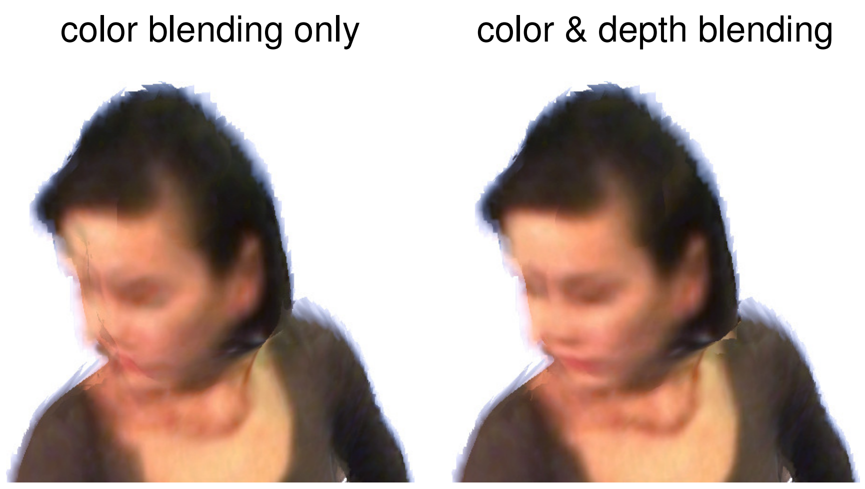
View-dependent Rendering

- Unstructured Lumigraph (UL) framework
  Buehler et al., Unstructured Lumigraph Rendering, SIGGRAPH 2001
- Render consistent depths
  - Render views into separate depth buffers
  - Resolve occlusions via fuzzy z-buffer
  - Blend remaining depths using UL
- Render consistent colors
  - Projective texturing
  - Blend colors using UL

We render the billboard clouds using the Unstructured Lumigraph Rendering framework.

In contrast to the original UL algorithm, our method does not only blend the colors but first reconstructs a consistent, view-dependent geometry of thescene, where each pixel has a uniquely assigned depth value. This results in much crisper renderings. If multiple fragments are rendered at the same pixel, its depth buffer value d is computed in a fragment program by blending all fragment depths di.

Rendering Comparison



Here is a comparison of rendering with only color blending and with color & depth blending.

Results – No Disparities

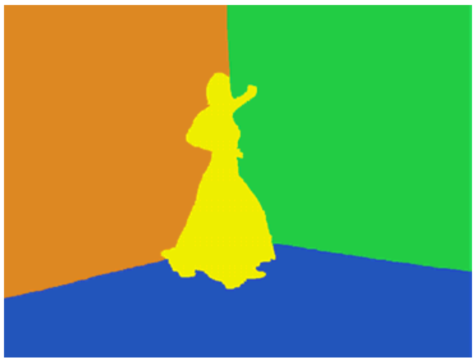
Results – Raw Disparities

Results – Filtered Disparities

Handling Scenes

- Segment scene using semi-automatic video cutout techniques

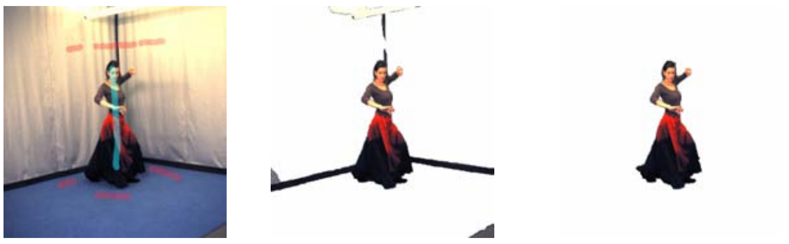
Li et al. Video Object Cut and Paste SIGGRAPH 2005
Wang et al., Interactive Video Cutout, SIGGRAPH 2005



To represent scenes and not only stand-alone objects, each view has to be decomposed into multiple billboards. We use a semi-automatic video cutout technique to segment the input videos into distinct objects. After the user has marked the objects in a single input frame of each view by applying a few brush strokes, a graph-cut optimization automatically computes the segments over time. The segment boundaries are refined by a Bayesian matting algorithm [Chuang et al., A bayesian approach to digital matting. CVPR 2001] yielding alpha mattes for the billboards.

Graph Cut Segmentation

- Graph cut on color and depth maps
  - Colors for accurate boundaries
  - Depths for increased stability



input colors only colors & depths

As well we include the depths in this process which yields significantly higher stability.

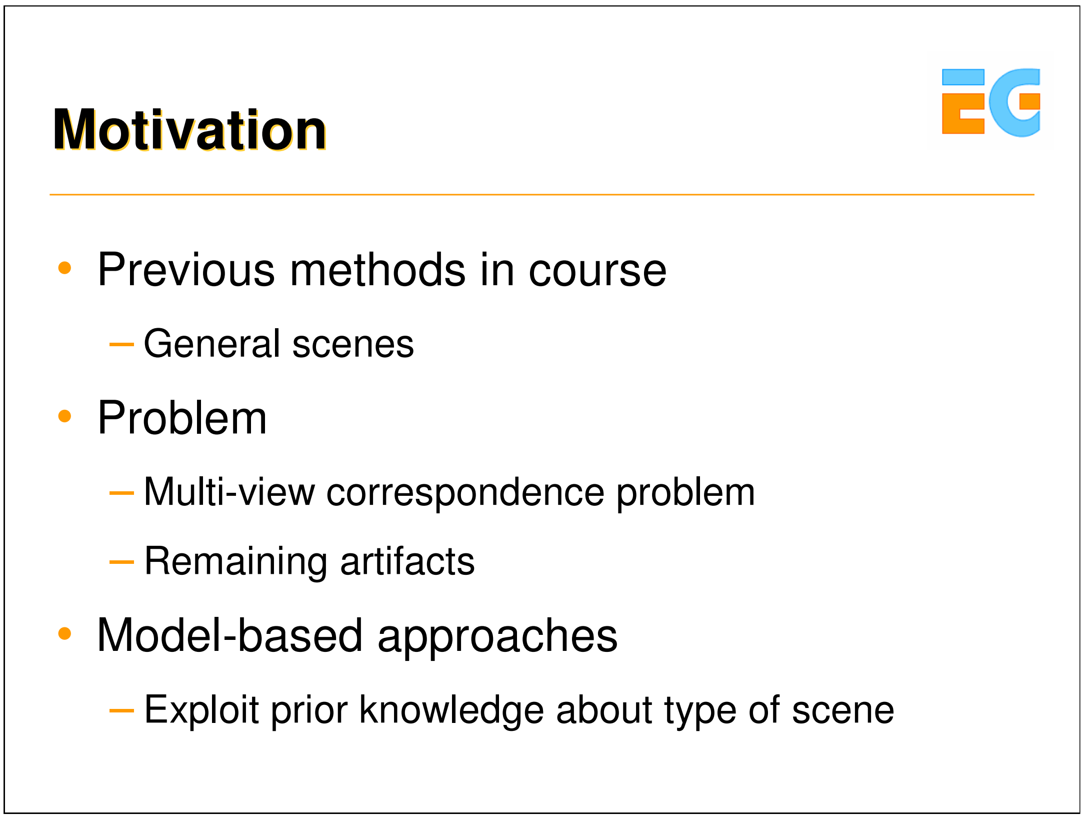
Results with Complete Scenes

Final result.

Model-based 3D Video I (25 min)

Christian Theobalt

Stanford University



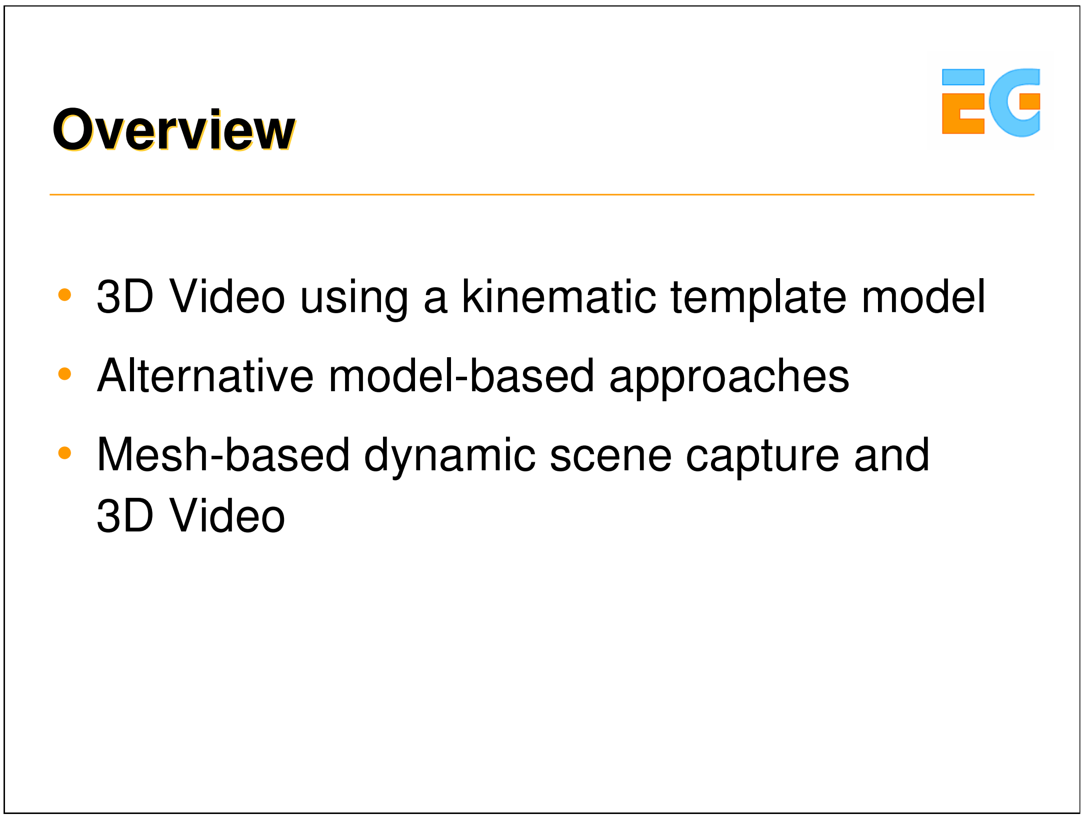
Motivation

- Previous methods in course
  - General scenes
- Problem
  - Multi-view correspondence problem
  - Remaining artifacts
- Model-based approaches
  - Exploit prior knowledge about type of scene

The approaches presented so far in the course did either not at all or just to a small extend exploit prior knowledge about the type of scene during reconstruction. This has the great benefit that arbitrary types of scenes can be processed. However, the multi-view reconstruction (and thus the multi-view correspondence problem) in this case is particularly hard and thus there may be noticeable artifacts.

For certain types of scenes it may therefore be beneficial to exploit prior knowledge about the type of subjects in the scene being reconstructed. Although by this means generality may be sacrificed, it may still be beneficial in terms of reconstruction quality.

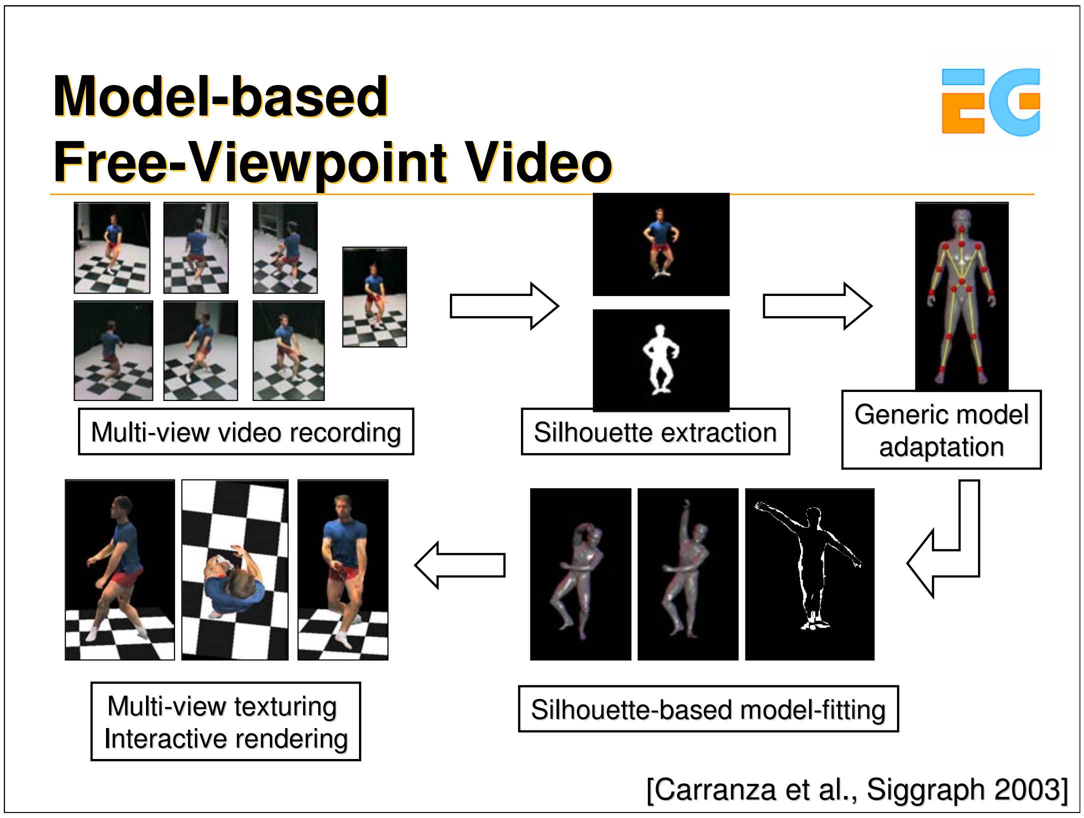
The approaches in this part of the course utilize prior shape and motion models for 3D video reconstruction. While the approaches in the first part of this section mainly deal with human actors, towards the end some more recent approaches are shown that can handle arbitrarily dressed people and even a broader category of subjects.



Overview

- 3D Video using a kinematic template model
- Alternative model-based approaches
- Mesh-based dynamic scene capture and 3D Video

The section on model-based 3D Video is subdivided into three main parts summarized above.



Model-based Free-Viewpoint Video

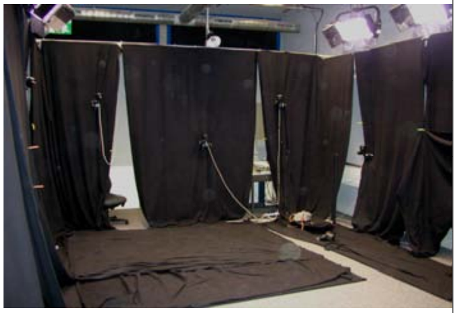
[Carranza et al., Siggraph 2003]

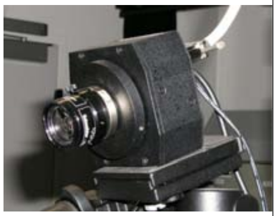
In free-viewpoint video the viewer is given the possibility to freely change his viewpoint onto a 3D rendition of a dynamic real-world scene. In order to generate a free-viewpoint video, the problems of acquisition of input data, reconstruction of a dynamic scene descriptions, and rendering in real-time have to be solved simultaneously.

This slide illustrates the workflow between algorithmic components of a model-based system to reconstruct and render free-viewpoint videos of human actors [3]. Inputs to our system are multiple frame-synchronized video streams showing a moving person that have been captured with calibrated video cameras. Image silhouettes of the person in the foreground are extracted from each video frame. A generic human body model consisting of 16 segments, a triangle mesh surface geometry with roughly 21000 triangles, and a kinematic skeleton made of 17 joints is employed to represent the time-varying appearance of the human. An analysis-by-synthesis approach based on silhouette-overlap is used to adapt the model in shape and proportions to the actor, and to determine pose parameters for each time step of video. Real-time renditions of the captured scene are generated by projectively texturing the moving body model from the input video frames that are appropriately blended.

Acquisition

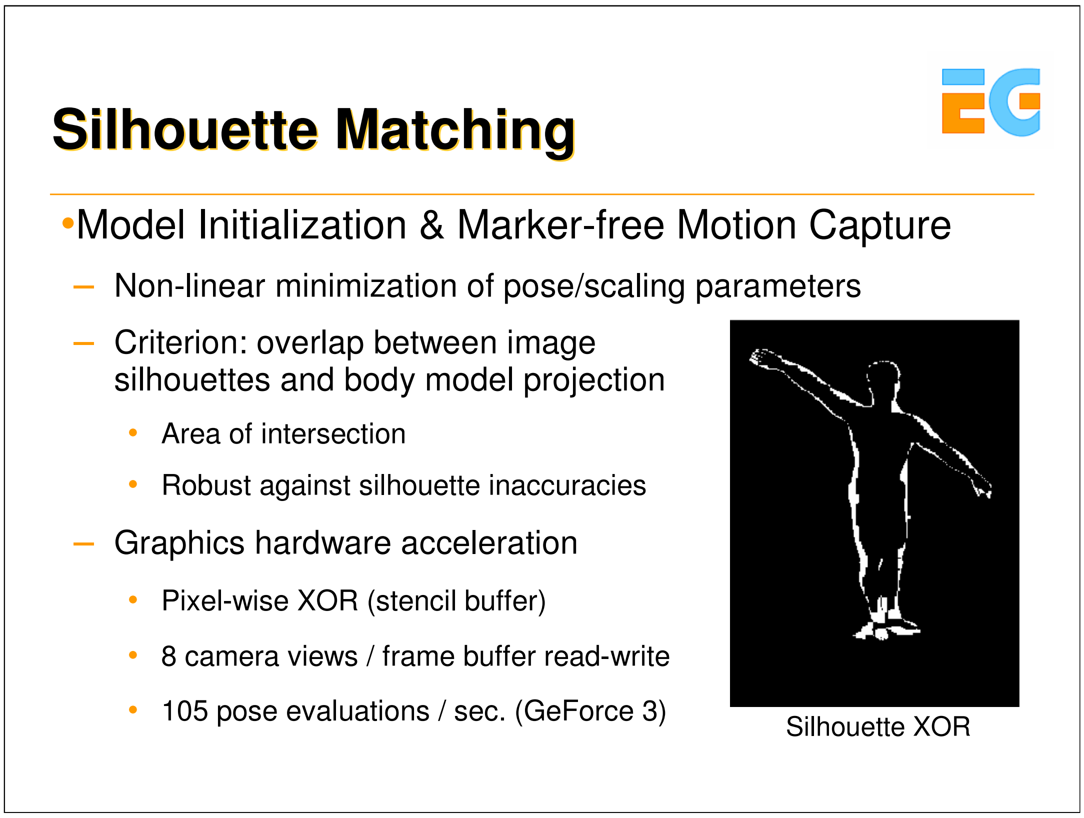
- 8 synced CCD cameras; currently:
  - Currently: 1004x1004,25fps, 12 bit
  - old: 320x240, 15 fps
- Calibrated
- Controlled lighting





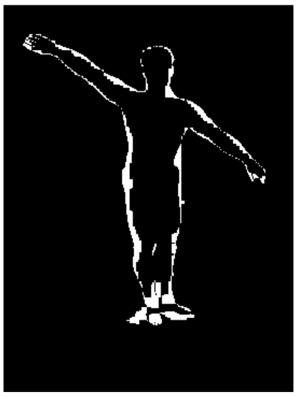
Multi-view video footage is recorded in a 3D video acquisition studio. This studio features 8 synchronized calibrated video cameras. Currently, we employ a system with eight CameraLink 1-Megapixel video cameras. In a previous version of the studio, we used 8 IEEE1394 video cameras with VGA resolution. Some of the older footage presented in this course has been recorded with our old setup.

In the remainder of the course, we will also show important hardware and software extensions to the studio in case they are relevant for a specific topic.



Silhouette Matching

- Model Initialization & Marker-free Motion Capture
  - Non-linear minimization of pose/scaling parameters
  - Criterion: overlap between image silhouettes and body model projection
    - Area of intersection
    - Robust against silhouette inaccuracies
  - Graphics hardware acceleration
    - Pixel-wise XOR (stencil buffer)
    - 8 camera views / frame buffer read-write
    - 105 pose evaluations / sec. (GeForce 3)

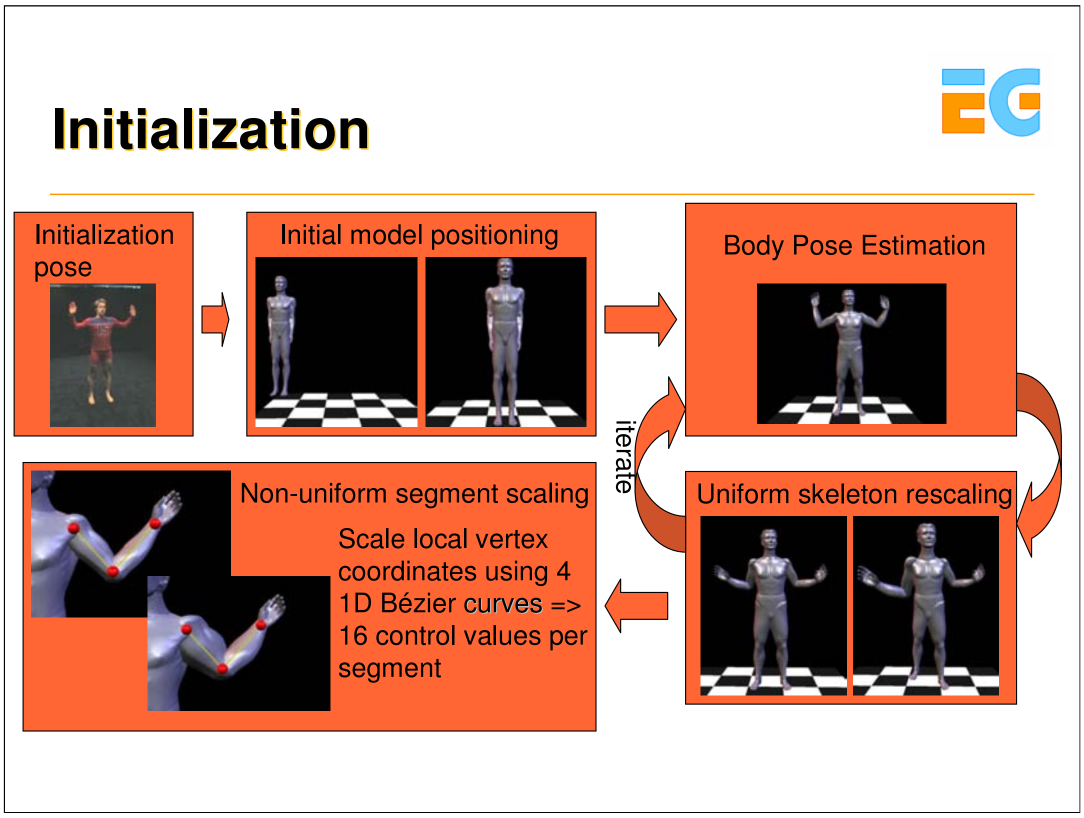

Silhouette XOR

Besides 35 parameters to specify the pose, our body model features for each segment a uniform scaling parameter, as well as 16 Bézier parameters for fine-tuning the appearance of the surface. Our analysis-by-synthesis approach employs the overlap between the projected model silhouettes and the image silhouettes in all camera views for two purposes:

1) Initialization: The geometry of the model is adapted to optimally represent the appearance of the real-world equivalent.

2) Marker-free Motion Capture: after the model has been customized, its pose is matched to the pose of the actor at each time step of the input video footage.

Both tasks require non-linear optimizations in the model parameters, the first one being performed in the pose and scaling parameters simultaneously, the second one being performed in the pose parameters only. The error function to be minimized computes the non-intersecting areas between the projected model and input silhouettes in all camera views. Conveniently, an estimate for this is obtained via the binary XOR between image and model silhouettes in all views. It can be efficiently evaluated on state-of-the-art graphics hardware using the stencil buffer. On a PC featuring an Intel Xeon 1.8 GHz CPU and an Nvidia GeForce3 GPU, we can perform 105 energy function evaluations using 8 cameras and a frame size of 320x240 pixels.



Initialization

Initialization pose
Initial model positioning
Body Pose Estimation
iterate
Uniform skeleton rescaling
Non-uniform segment scaling
Scale local vertex coordinates using 4 1D Bézier curves => 16 control values per segment

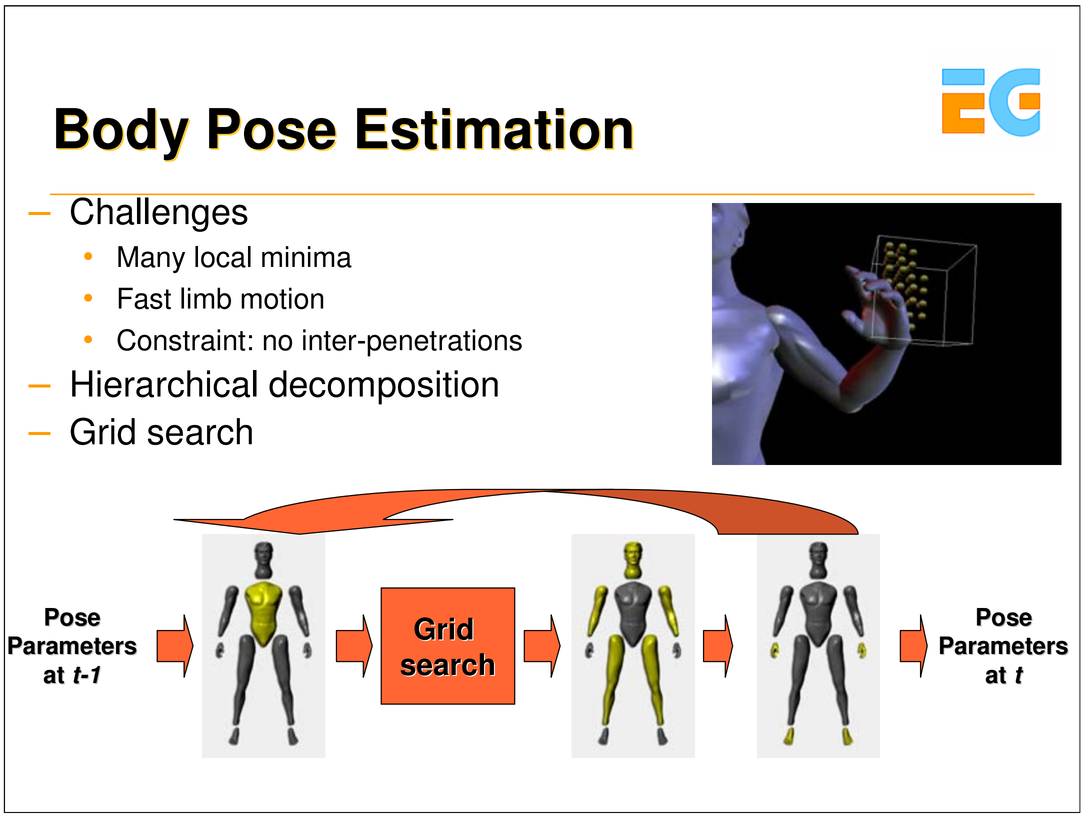
The shape and proportions of the body model are adapted to the shape of the actor in an iterative optimization procedure. Inputs are silhouette images of the actor striking a dedicated initialization pose. In a first step the position and orientation of the torso segment in 3D space is determined. The space of pose parameters for the torso is sampled regularly to find an optimal starting point for a subsequent downhill optimization that determines the final torso pose.

Thereafter, the method iterates between determining a new set of pose parameters for the whole model (explained on the next slide), and determining a new set of uniform scaling parameters for each segment.

Finally, optimal parameters for the 1D Bézier scaling functions are found that bring the segment outlines into optimal accordance with the multi-view silhouettes. On the hands and feet no Bézier scaling is performed.

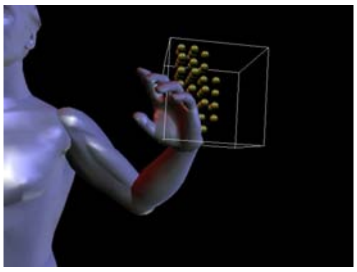
The estimation of the optimal scaling parameters is performed hierarchically. It starts with the root of the skeleton (located in the torso) and proceeds layer-by-layer further down the skeleton hierarchy until the terminating nodes (head, hands and feet are reached).

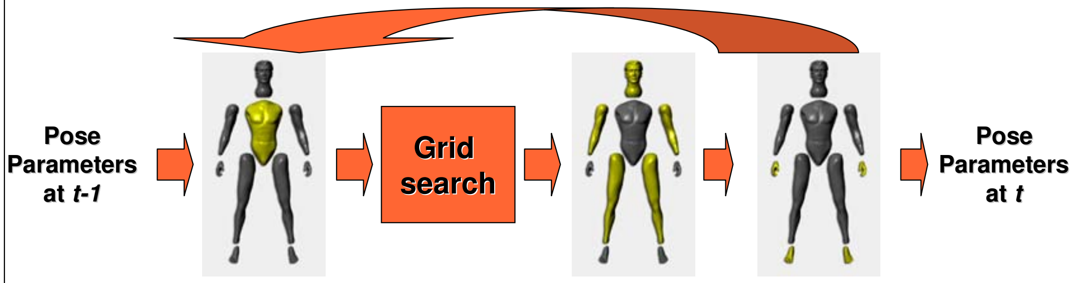
For numerical minimization we employ a standard downhill direction set method such as Powell's method. The shape parameters of the model remain fixed for the duration of a free-viewpoint video.



Body Pose Estimation

- Challenges
  - Many local minima
  - Fast limb motion
  - Constraint: no inter-penetrations
- Hierarchical decomposition
- Grid search





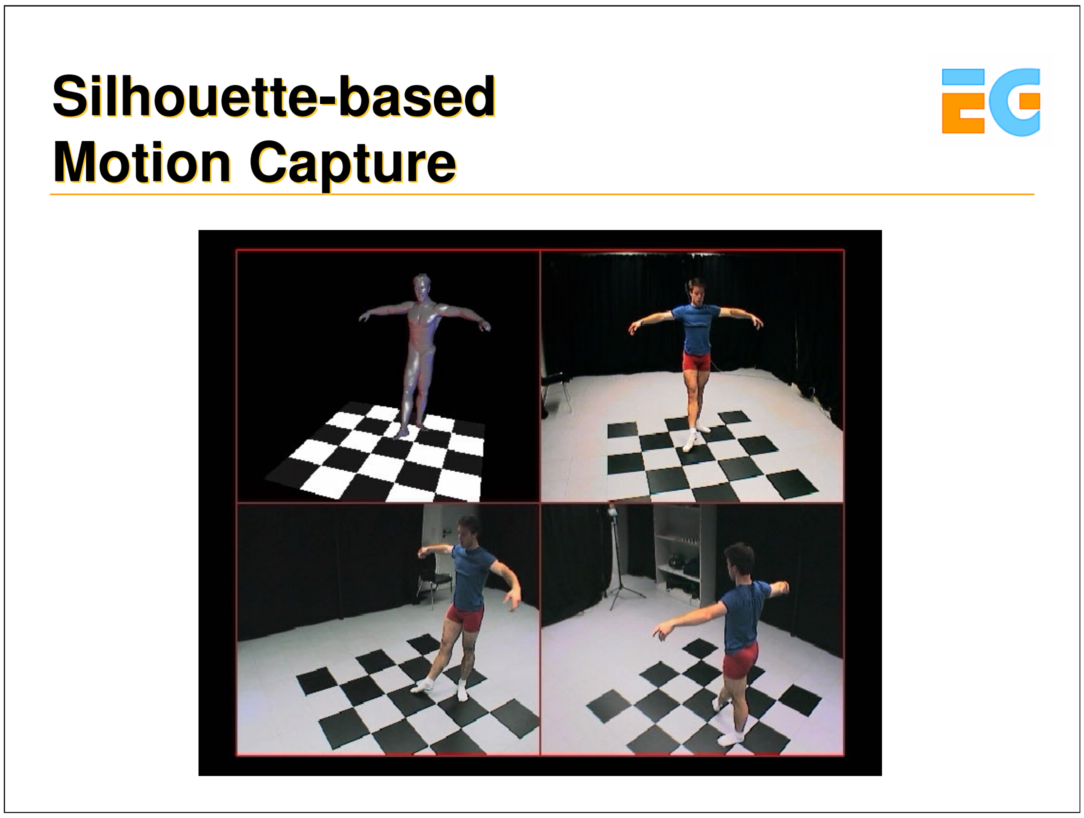
Only with a completely passive marker-free motion capture approach the same video material can be used for motion and texture estimation. Determining the 35 pose parameters for each time step of video is a challenging problem since the non-convex energy-functional exhibits many local minima. Potentially rapid limb motion and constraints on allowable body poses require the parameter search space itself to be constrained. Only this way robust convergence to the correct solution can be assured.

We initialize the optimization search for one time step t with the parameters found at t-1. The problem's search space is constrained by performing a hierarchical optimization that exploits structural knowledge about the body. The poses of individual kinematic sub-chains in the skeleton are determined separately. Body segments are considered in descending order with respect to the skeleton hierarchy and their respective influence on the silhouette appearance. In consequence, we first determine the torso pose, thereafter the poses of arms, legs, and head, and finally the parameters for the hands and feet.

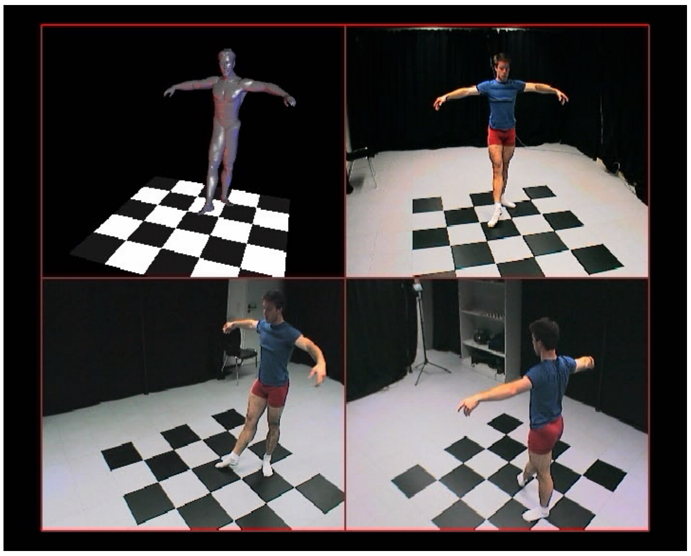
For arms and legs we employ a custom 4-degree-of-freedom parameterization.

In order to handle rapid body motion the pose determination for each limb is preceded with a regular grid search in the 4D pose space. The best grid point found is used as starting point for the subsequent downhill optimization. Interpenetrating limb poses are also discarded during grid sampling.

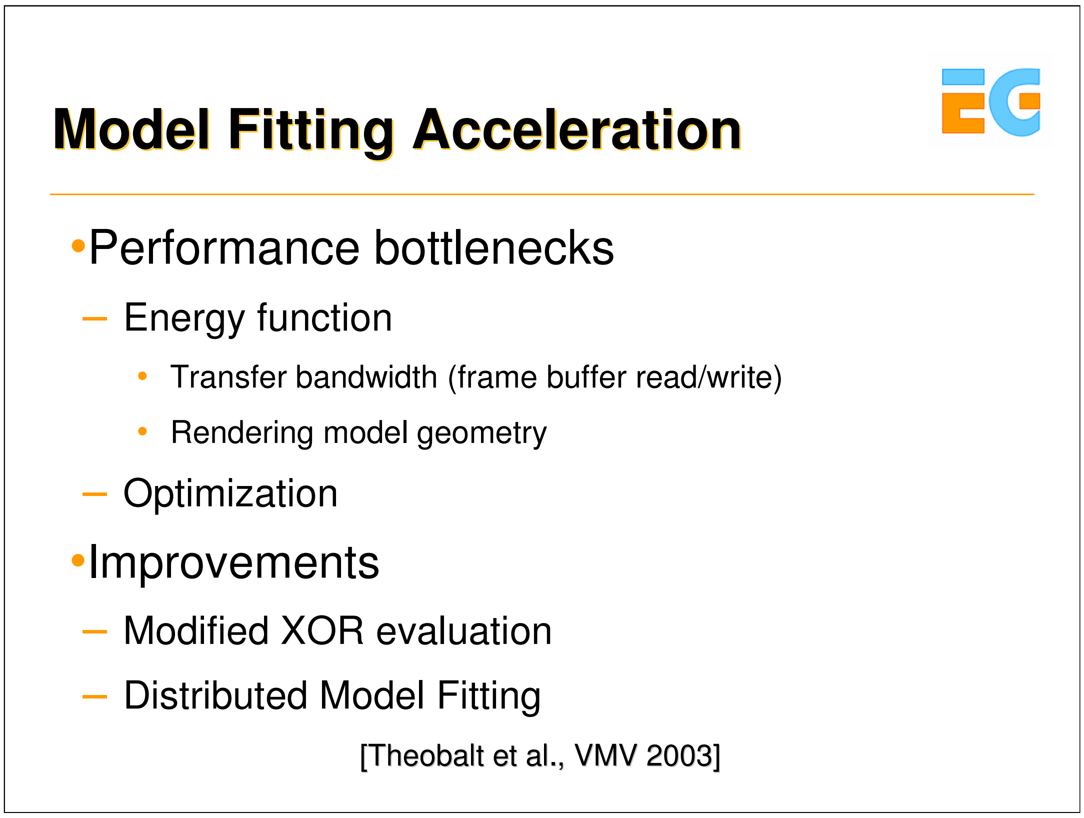
Optionally, the complete pose estimation scheme is iterated.



Silhouette-based Motion Capture



The video shows three of the input camera views and the body model correctly following the motion of the dancer.



Model Fitting Acceleration

- Performance bottlenecks
  - Energy function
    - Transfer bandwidth (frame buffer read/write)
    - Rendering model geometry
  - Optimization
- Improvements
  - Modified XOR evaluation
  - Distributed Model Fitting

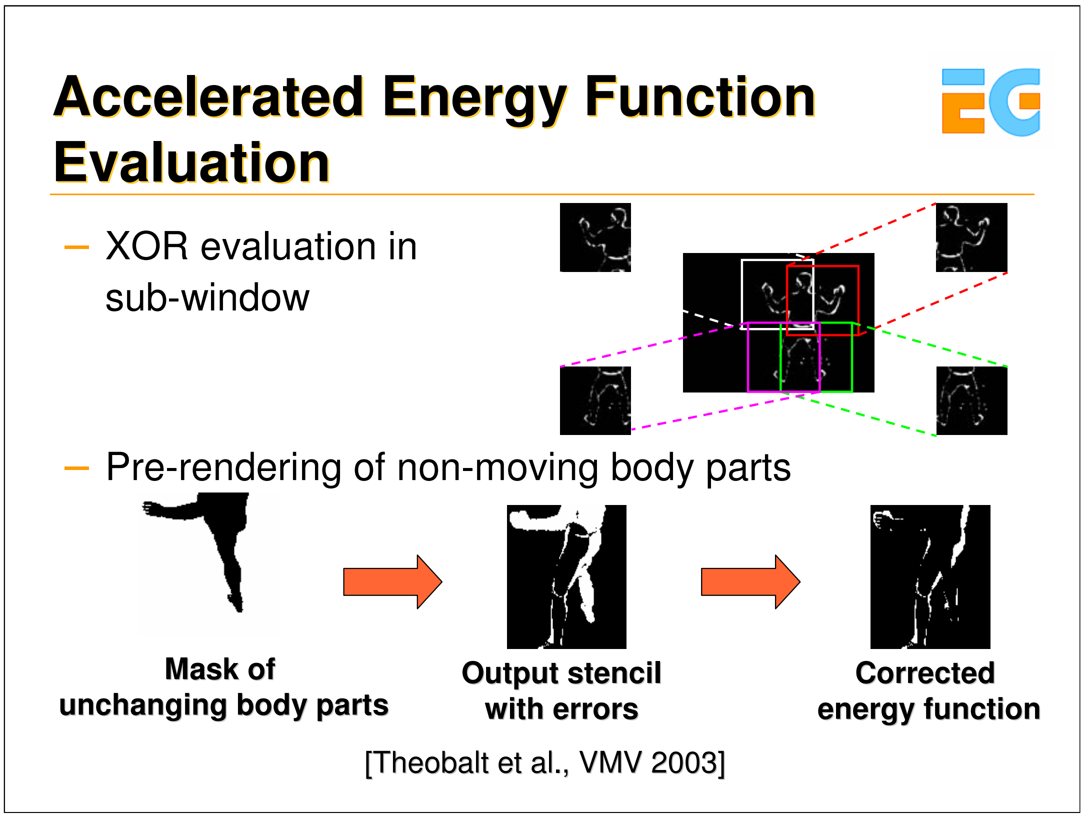
[Theobalt et al., VMV 2003]

The performance of the model fitting method is limited by two factors: the time needed to evaluate the XOR energy function and the runtime of the numerical minimizer itself.

The performance of the energy function evaluation on the GPU is constrained by the overhead inflicted by the necessary frame-buffer read/write operations, as well as the overhead to render the model geometry.

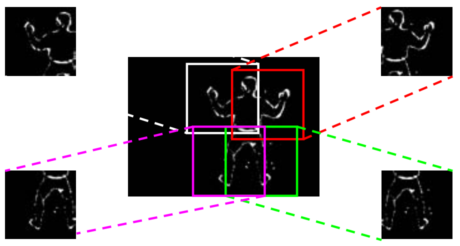
The performance of the pose determination procedure is constrained by the fact that on a single PC one can only optimize the pose for one body segment at a time. Given more CPUs, however, the parameters for independent segments on the same level of the skeleton hierarchy could be efficiently estimated in parallel.

Thus, we have enhanced the GPU-based XOR computation in order to capitalize on the compartmentalized nature of the pose determination problem [1]. The implicit parallel structure of the problem also suggests a distributed implementation of the motion capture sub-system as a whole.

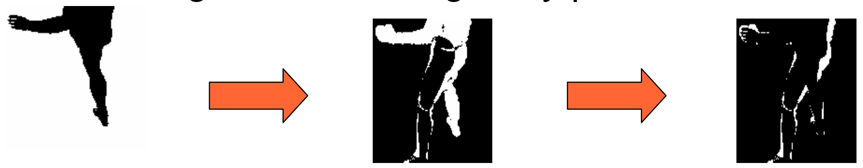


Accelerated Energy Function Evaluation

– XOR evaluation in sub-window



– Pre-rendering of non-moving body parts

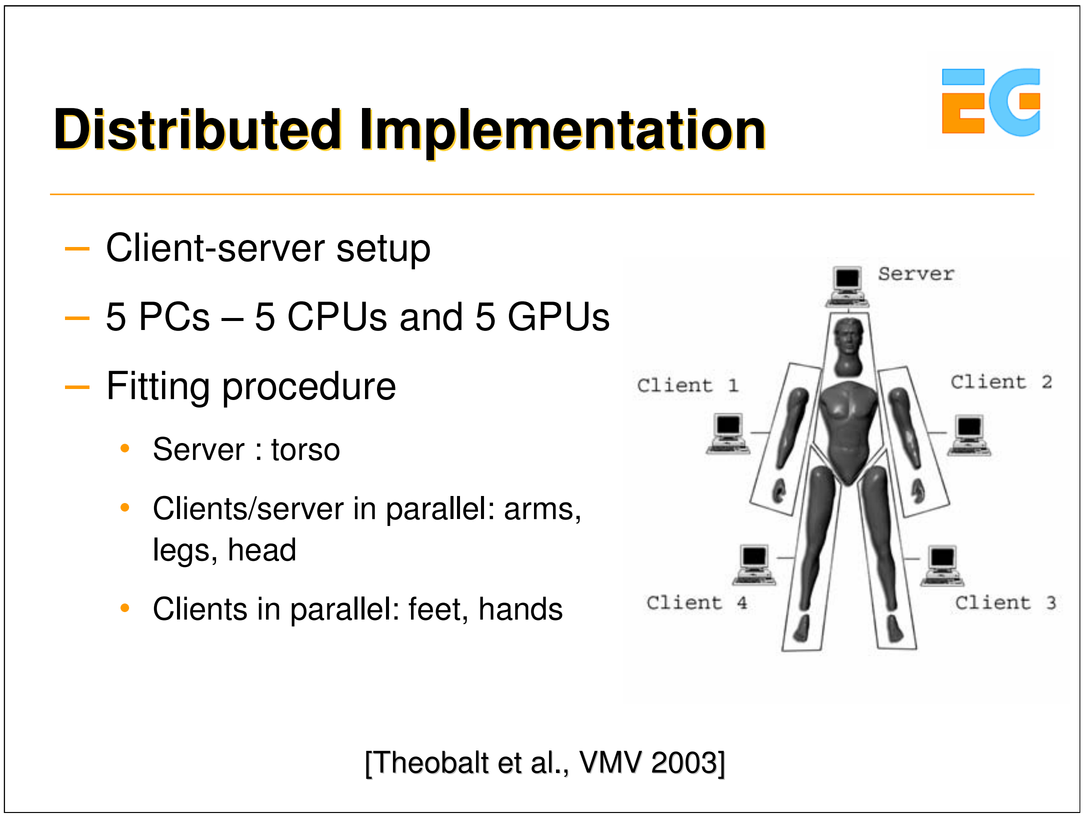

Mask of unchanging body parts
Output stencil with errors
Corrected energy function

[Theobalt et al., VMV 2003]

We modify the error function evaluation in two ways in order to exploit the compartmentalized nature of the pose determination problem:

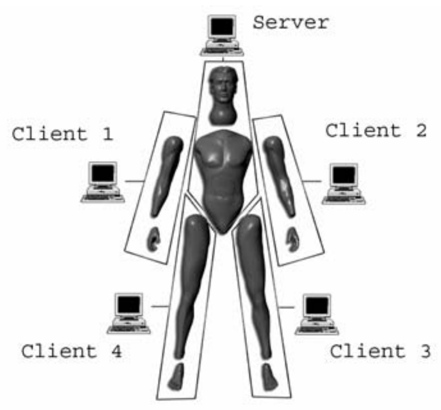
1) Instead of rendering the frames in full size, the rendering window for each camera views is confined to a limited area around the image plane location of that kinematic sub-chain which is currently optimized. By this means the amount of data that has to be transferred during frame-buffer read/write is significantly reduced.

2) The rendering overhead for the body model can be reduced if only the geometry of those body parts is displayed whose pose is currently optimized. The additional error in the XOR value in each camera view that is inflicted by not showing large parts of the model has to be compensated. We do this by applying an image-mask of unchanging body parts that is generated prior to the optimization. The bottom figure illustrates the process for one camera view.



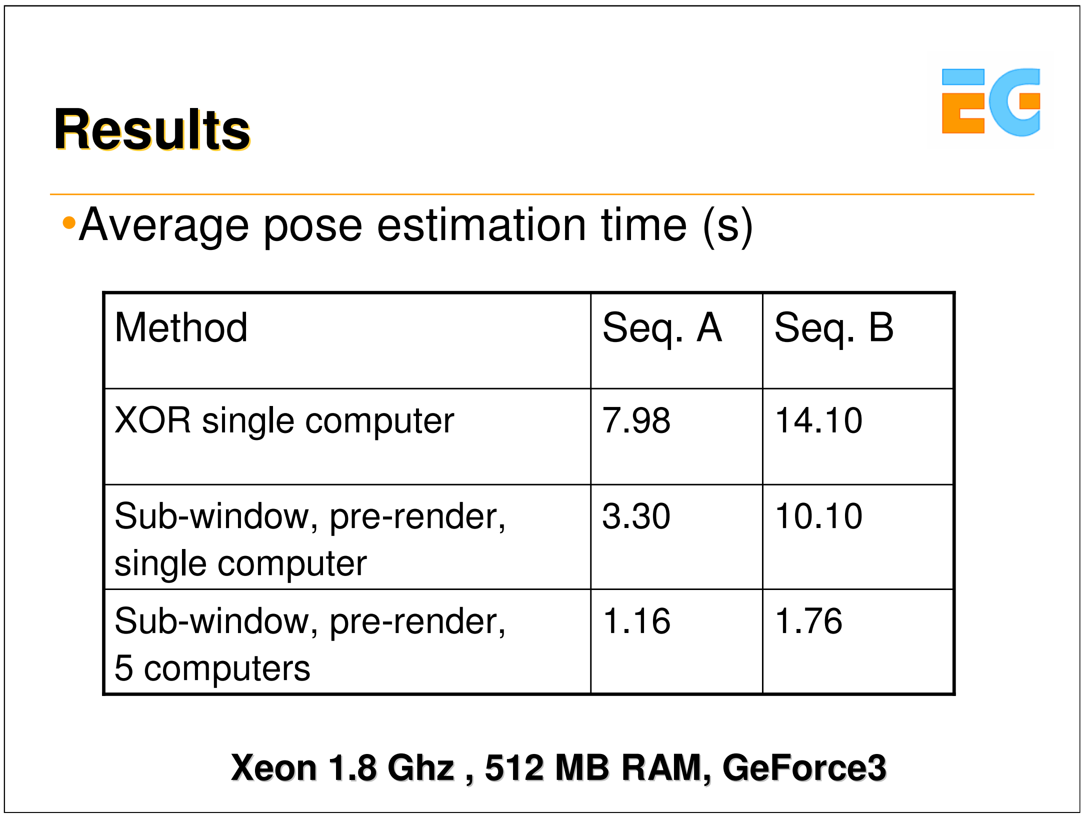
Distributed Implementation

- Client-server setup
- 5 PCs – 5 CPUs and 5 GPUs
- Fitting procedure
  - Server : torso
  - Clients/server in parallel: arms, legs, head
  - Clients in parallel: feet, hands



[Theobalt et al., VMV 2003]

Our distributed pose determination system is a client-server setup using 5 PCs, i.e. 5 CPUs and 5 GPUs. The hierarchical pose estimation procedure first determines the parameters of the torso on the server. The result is transferred to all clients. Now, the server and the clients determine the parameters of legs, arms and head in parallel. The results are broadcasted via the server before, in a final step, the poses of hands and feet are determined.



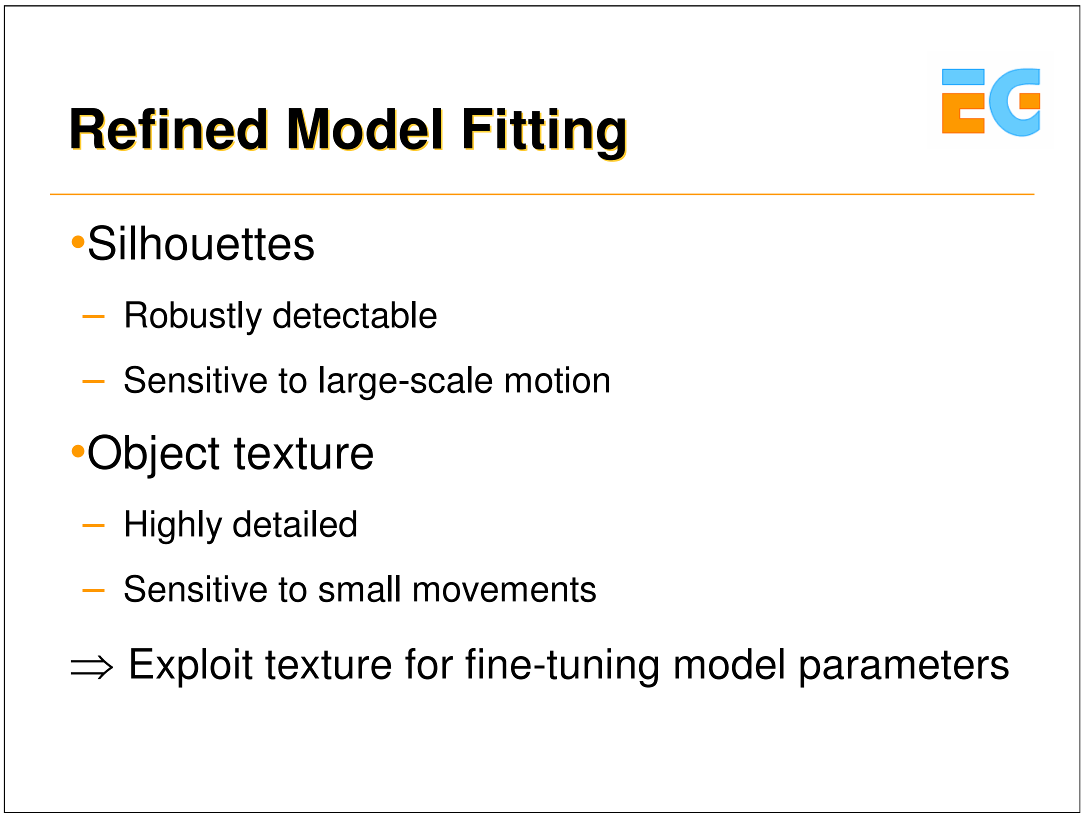
Results

- Average pose estimation time (s)

Method	Seq. A	Seq. B
XOR single computer	7.98	14.10
Sub-window, pre-render, single computer	3.30	10.10
Sub-window, pre-render, 5 computers	1.16	1.76

Xeon 1.8 Ghz , 512 MB RAM, GeForce3

The table shows average times needed for determining the pose of the body model at a single time step with the different algorithmic alternatives for silhouette fitting. The results we obtained with two test sequences are shown. Seq.A exhibits mostly slow body motion, whereas Seq.B shows an expressive jazz dance performance. Motion capture with the non-accelerated XOR computation on a single computer takes between 8s and 14s. A significant speed-up is obtained if we apply the enhanced energy function evaluation with sub-window constraint and body-part pre-rendering. By employing our distributed implementation we achieve average fitting times close to 1s.



Refined Model Fitting

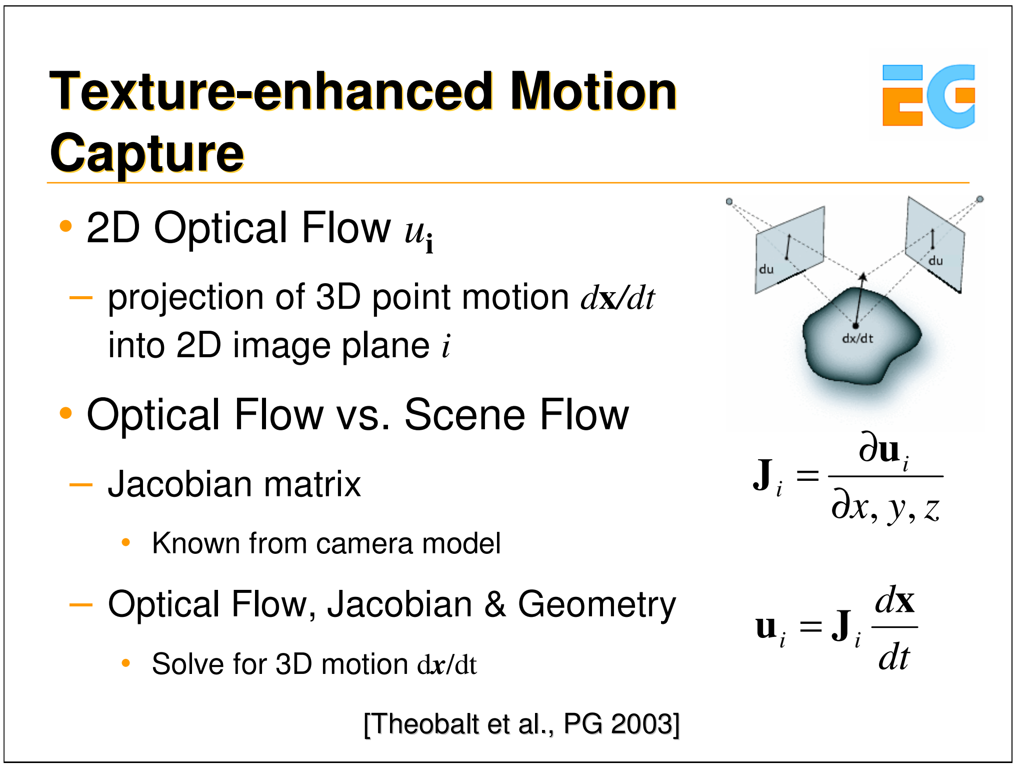
- Silhouettes
  - Robustly detectable
  - Sensitive to large-scale motion
- Object texture
  - Highly detailed
  - Sensitive to small movements

$\Rightarrow$ Exploit texture for fine-tuning model parameters

The silhouette-based analysis-by-synthesis approach described on the previous slides enables us to reconstruct a dynamic scene description without having to modify the input video footage in any form, e.g. with optical markings in the scene. This is a necessary precondition if texture information is taken from the video streams as well.

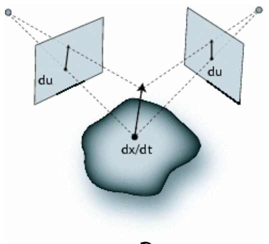
Image silhouettes can be computed robustly and our motion capture approach is fairly insensitive to measurement noise in the image data. However, although a silhouette-based approach robustly captures poses on a large scale, the exact pose of some body parts is hard to infer exactly. Slight pose inaccuracies can often be observed for those segments whose shape exhibits very few features on the silhouette outline that guide the optimization towards the correct parameters, such as the head.

We thus propose to enhance the original motion capture method into a hybrid approach that jointly uses silhouettes and texture information from the video frames for pose determination. The texture information enables the correction of slight pose inaccuracies that exist in the silhouette fit.



Texture-enhanced Motion Capture

- 2D Optical Flow $u_\mathbf{i}$
  - projection of 3D point motion $d\mathbf{x}/dt$ into 2D image plane i
- Optical Flow vs. Scene Flow
  - Jacobian matrix
    - Known from camera model
  - Optical Flow, Jacobian & Geometry
    - Solve for 3D motion $d\mathbf{x}/dt$



$$\mathbf{J}_i = \frac{\partial \mathbf{u}_i}{\partial x, y, z}$$

$$\mathbf{u}_i = \mathbf{J}_i \frac{d\mathbf{x}}{dt}$$

[Theobalt et al., PG 2003]

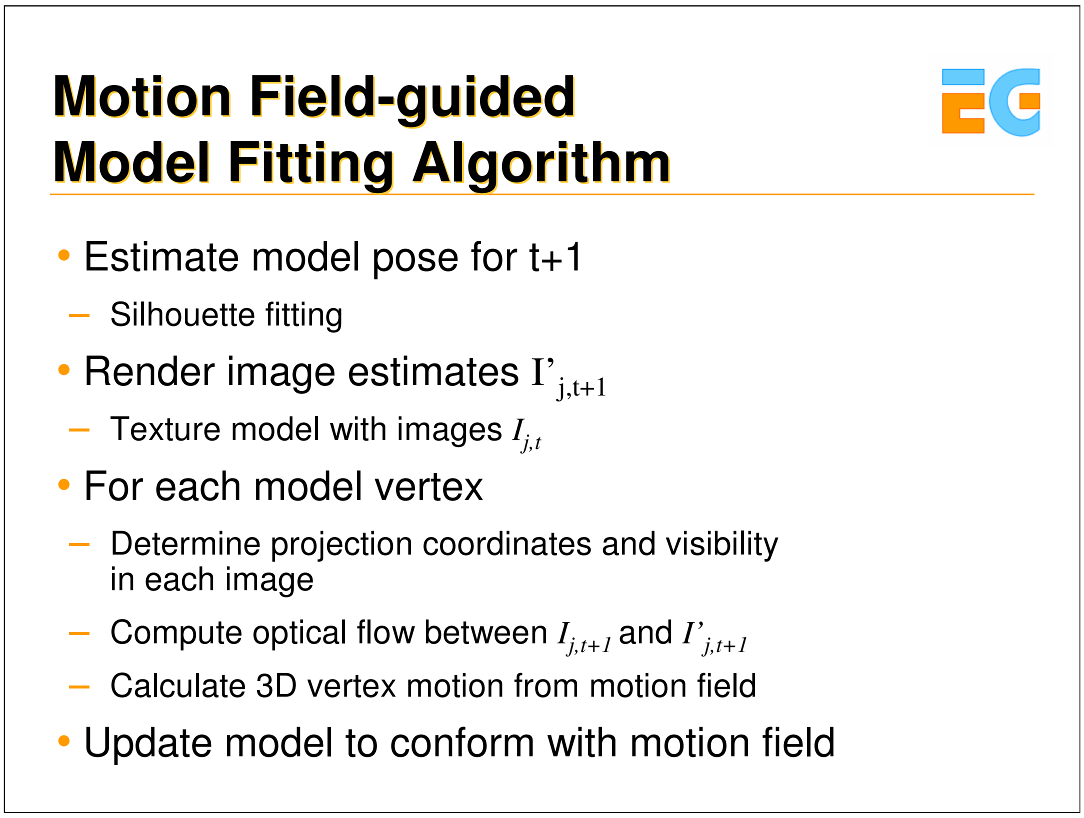
Our texture-enhanced motion capture method employs the ***scene flow***, the 3D equivalent of the 2D optical flow in the image plane, to compute corrective pose updates.

The 2D optical flow is the projection of the 3D motion field of a dynamic scene into the image plane of a recording camera.

A sea of algorithms has been proposed in the computer vision literature to compute this 2D flow field between two subsequent depictions of the scene. In our implementation we employ the method by Lukas and Kanade (see [2] for references to the original literature).

The scene flow corresponding to a set of optical flows in multiple camera views is the set of 3D motion vectors, one for each point in the scene, whose projections are the 2D optical flows. The differential relationship between the optical flow and the scene flow is described by a Jacobian matrix whose entries can be determined from the matrix of a calibrated camera.

Given a set of optical flows from multiple calibrated camera views, and full knowledge about the 3D geometry, it is possible to infer the scene flow vector for each point on the geometry by solving a linear equation system.



Motion Field-guided Model Fitting Algorithm

- Estimate model pose for t+1
  - Silhouette fitting
- Render image estimates $I'_{j,t+1}$
  - Texture model with images $I_{j,t}$
- For each model vertex
  - Determine projection coordinates and visibility in each image
  - Compute optical flow between $I_{j,t+1}$ and $I'_{j,t+1}$
  - Calculate 3D vertex motion from motion field
- Update model to conform with motion field

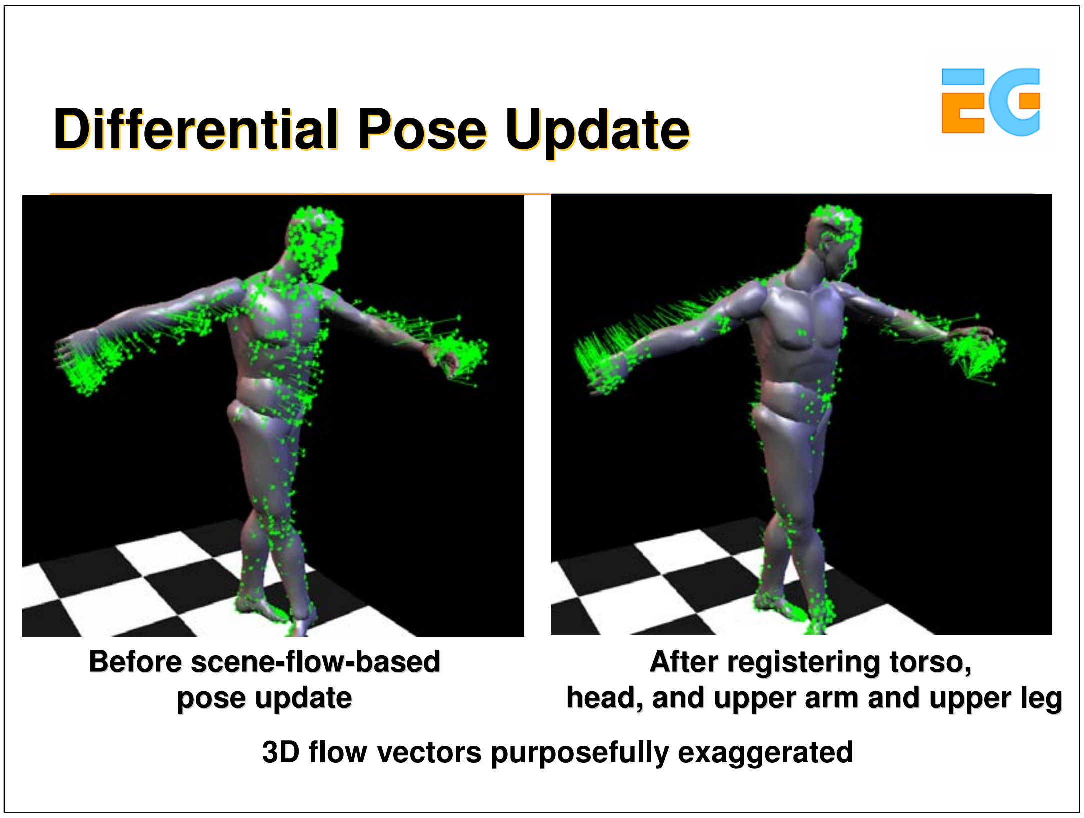
In a predictor-corrector-scheme, we employ the scene flow determination method for computing differential pose updates that correct inaccuracies in the silhouette-fit.

In a first step, an estimate of the pose parameters at time t+1 is computed using the original silhouette-based motion capture technique.

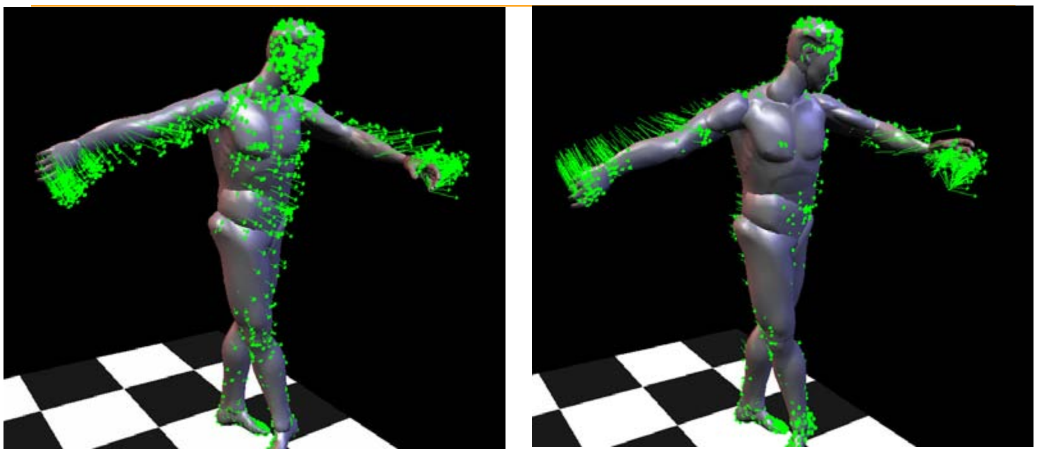
Using this first pose estimate, we predict the appearance of the model by rendering and projectively texturing it with the camera images from the previous time step $I_{j,t}$. This way, we generate novel images $I'_{j,t+1}$ showing the predicted model appearance at time t+1 in each camera view *j*.

For each vertex it is determined in which cameras it is visible. For each camera that sees the vertex, the optical flow between its projection into $I'_{j,t+1}$ and $I_{j,t+1}$, is computed. Using all the 2D flow vectors found for this vertex, the corresponding 3D scene flow vector is computed using the method outlined on the previous slide.

This way, a scene flow field is generated which describes for each vertex a position update that brings the model into a pose that is photo-consistent with all camera images. The corrective flow field is translated into a set of corrective pose parameters for the model by means of a shape registration method originally proposed by Horn (see [2]).



Differential Pose Update



Before scene-flow-based pose update

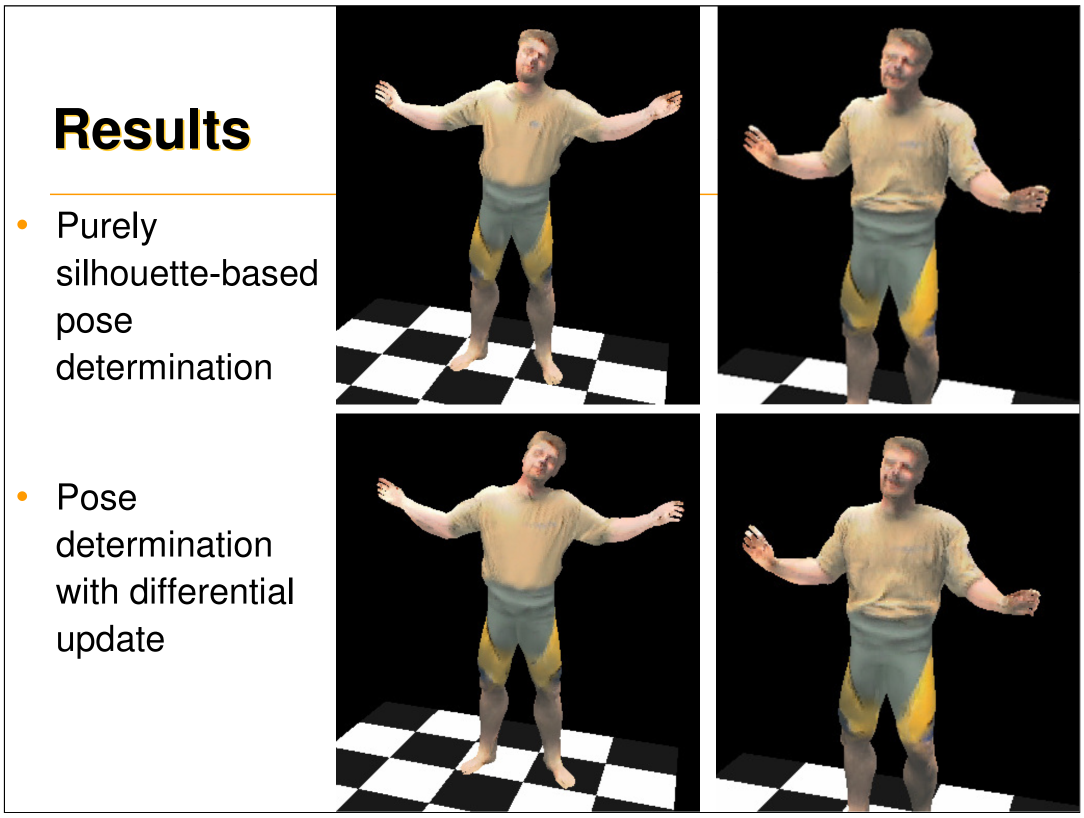
After registering torso, head, and upper arm and upper leg

3D flow vectors purposefully exaggerated

The figure on the left shows the body model in the pose that was found via silhouette–fitting.

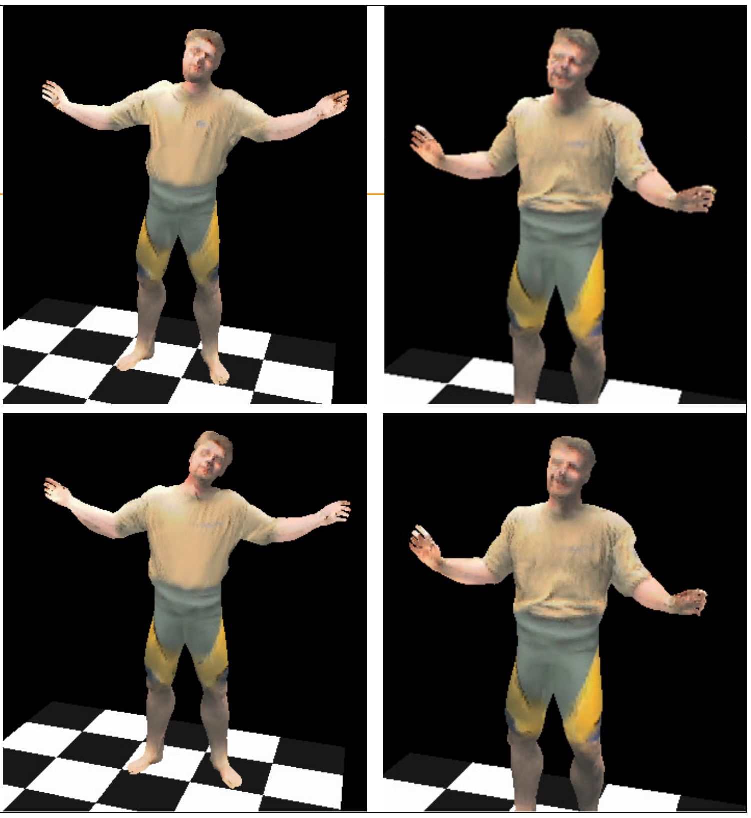
The green arrows are corrective scene flow vectors that have been computed for all vertices in the body model using the predictor-corrector scheme described on the previous slide. The length of the flow vectors has been purposefully exaggerated in order to better visualize the flow field.

The figure on the right shows the pose of the model after the corrective pose update parameters for the torso, the head, the upper arms and the upper legs have been applied to the skeleton.



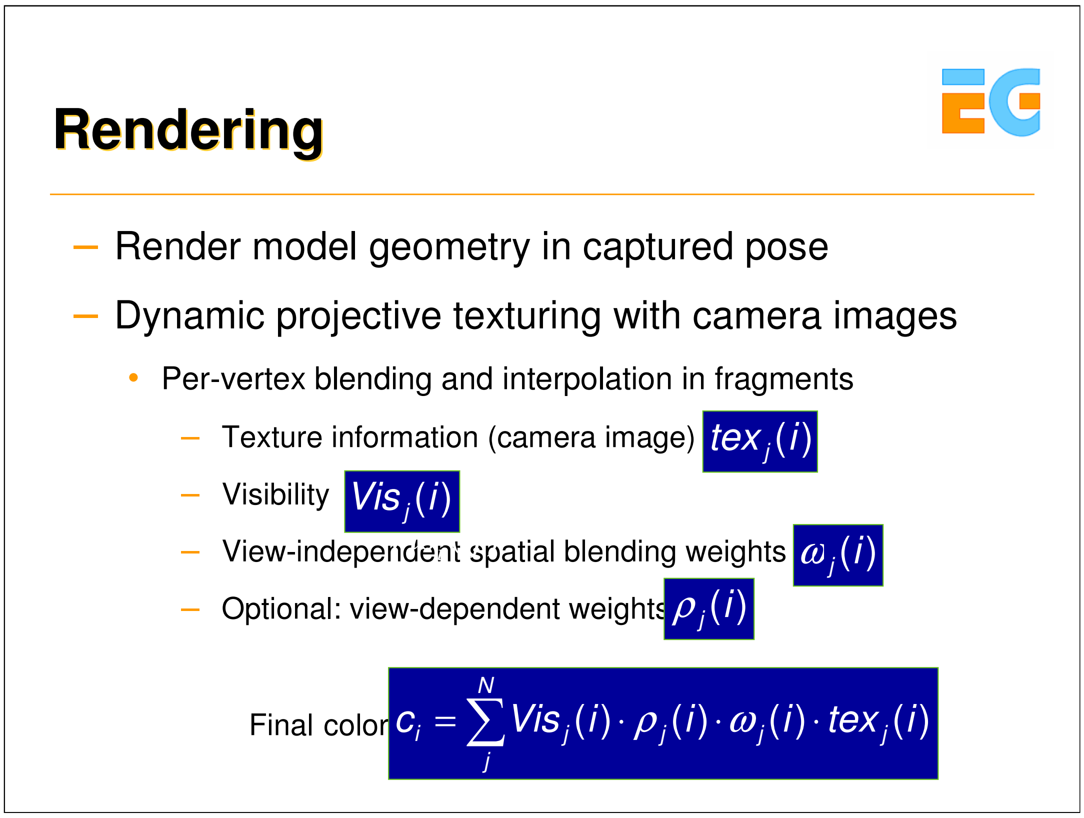
Results

- Purely silhouette-based pose determination
- Pose determination with differential update



The four figures illustrate the visual improvements achievable with the texture-enhanced motion capture method. The top row shows screen-shots of free-viewpoint videos that have been reconstructed with pure silhouette-fitting.

The bottom row shows renditions from the same virtual camera views but with the scene-flow-based pose correction applied. Improvements are mainly visible in the face and on the torso. Block artifacts in the images are due to the limited camera resolution of 320x240 pixels.



Rendering

- Render model geometry in captured pose
- Dynamic projective texturing with camera images
  - Per-vertex blending and interpolation in fragments
    - Texture information (camera image) $tex_j(i)$
    - Visibility $Vis_j(i)$
    - View-independent spatial blending weights $\omega_j(i)$
    - Optional: view-dependent weights $\rho_j(i)$

Final color $c_i = \sum_j^N Vis_j(i) \cdot \rho_j(i) \cdot \omega_j(i) \cdot tex_j(i)$

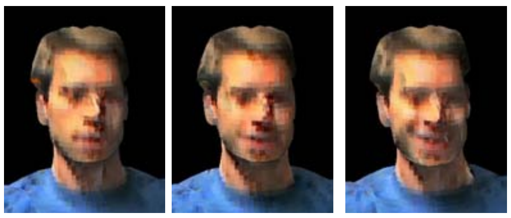
The reconstructed 3D videos are rendered by displaying the body model in the sequence of captured poses, and projectively texturing it with the video frames at each time step of video.

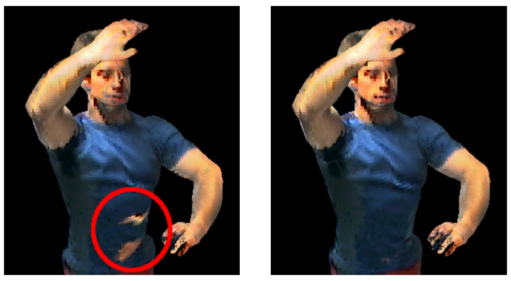
For vertex i in the model we compute the final color c_i by appropriately weighting and summing the color contributions from each input camera view $tex_j(i)$.$Vis_j(i)$ is the visibility of vertex i in camera view j, $\omega_j(i)$ is a spatial blending weight depending on the relative orientation of the vertex with respect to the input camera. We can assume that the reflectance of most garments is close to lambertian. It is thus valid to compute the spatial texture blending weights only based on the orientation of a vertex with respect to the input cameras. The camera which sees a surface element most head-on is assigned the highest blending weight.

However, in order to model view-dependent reflectance effects appearing in some garments, an optional view-dependent rescaling factor $\rho_j(i)$ can be included into the color computation. It weights the camera contributions depending on the relation between outgoing viewing direction and camera viewing direction. The per-vertex blending weights are interpolated in the fragment stage of the GPU.

Video Texturing

- Time-varying texture
  - Cloth folds
  - Local shadows
  - Facial expressions
- Detail preservation
  - Small inaccuracies between geometry and silhouettes
  - Soft shadow visibility
  - Camera image dilation
  - Controllable spatial blending weights

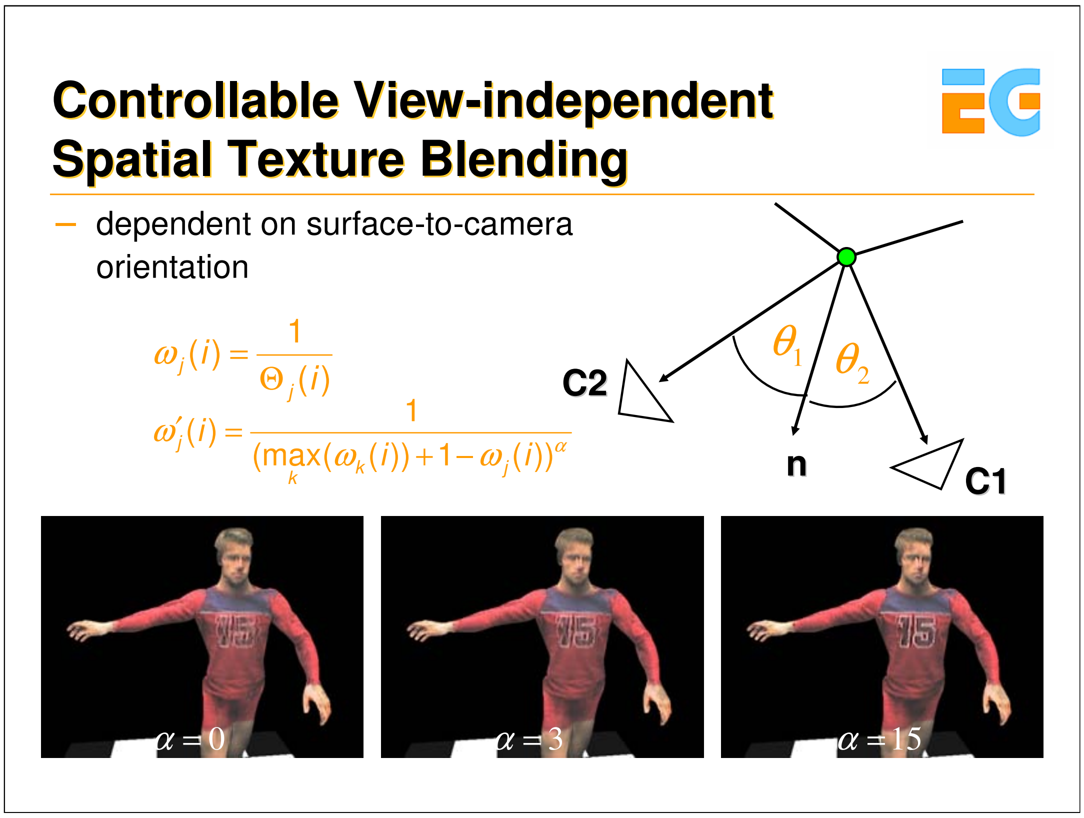




By generating a novel texture for each time step of video, subtle dynamic details in surface appearance, such as wrinkles in clothing, local shadows and facial expressions are faithfully reproduced (see three Figs. In top row, block artifacts are due to the limited camera resolution of 320x240).

Although our automatic model initialization approach generates a body geometry that matches the appearance of the actor very well, small geometry misalignments between the virtual and the real human may still exist. In These mismatches may lead to erroneous projections of parts of the texture belonging to occluding geometry onto actually more distant parts of the body (bottom left Fig.). Furthermore, seams originating from projected silhouette boundaries may appear on the body.

We employ three techniques to counter these effects. Firstly, we compute the visibility of each vertex from a set of slightly displaced camera views (soft shadow visibility). This way, projection artifacts at occlusion boundaries are prevented (bottom right Fig.). Secondly, we dilate the segmented video frames at the silhouette boundaries to prevent the seams. Thirdly, we apply a special method to compute controllable spatial texture blending weights which is illustrated on the next slide.

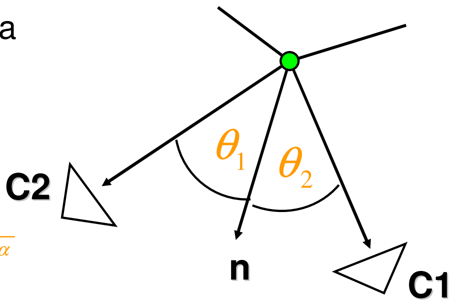


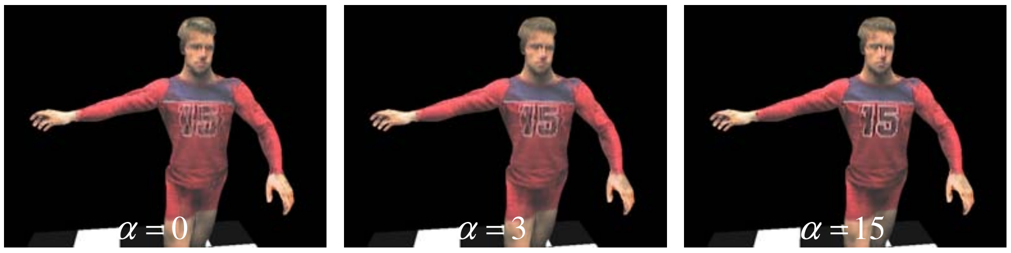
Controllable View-independent Spatial Texture Blending

– dependent on surface-to-camera orientation

$$\omega_j(i) = \frac{1}{\Theta_j(i)}$$

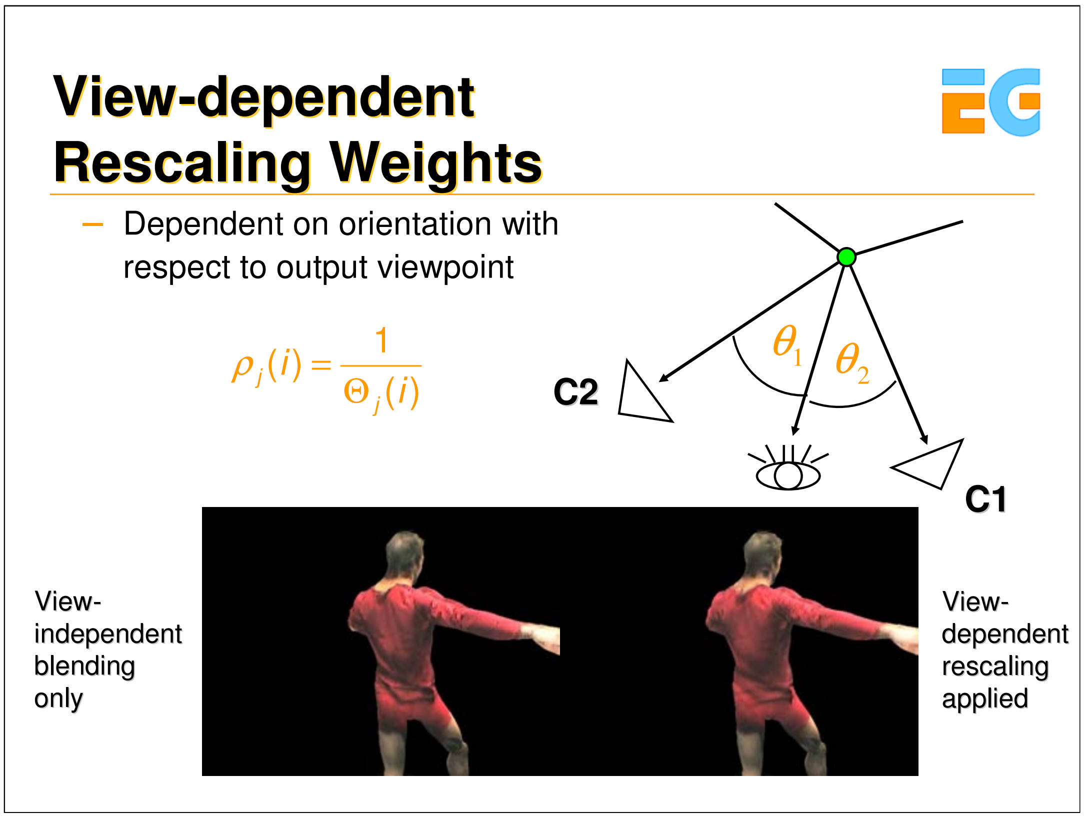
$$\omega'_j(i) = \frac{1}{(\max_k(\omega_k(i)) + 1 - \omega_j(i))^{\alpha}}$$





The easiest way to compute a view-independent spatial blending weight for vertex *i* and camera *j*, $\omega_j(i)$, is to apply the reciprocal of the angle $\theta_j(i)$ between the vertex normal and the viewing vector towards the camera.

We propose alternative spatial blending weights, $\omega'_j(i)$, that give a better control over the influence of each camera on the appearance of the final texture. The alternative weight computation assigns a proportionally high weight to the camera which sees the vertex most head-on. The sharpness value α controls the amount by which the influence of the best camera is exaggerated. In the limit, $\alpha \rightarrow \infty$, only the best camera is contributing to the color of the vertex. The three figures on the bottom of the slide illustrate the influence of the sharpness value on the final renditions.



View-dependent Rescaling Weights

– Dependent on orientation with respect to output viewpoint

$$\rho_j(i) = \frac{1}{\Theta_j(i)}$$

C2
θ_1
θ_2
C1

View-independent blending only

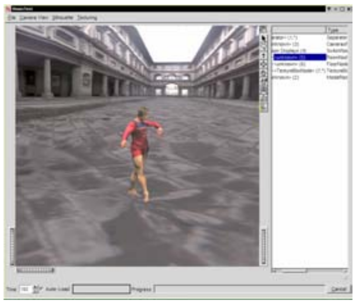
View-dependent rescaling applied

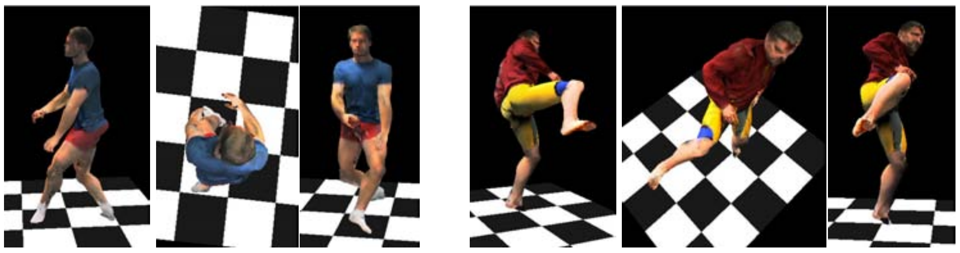
Optionally, a weight $\rho_j(i)$ can be computed that view-dependently rescales the view-independent blending weights.

The weight $\rho_j(i)$ for vertex i and camera j is the reciprocal of the angle $\Theta_j(i)$ between the viewing direction towards the camera and the current outgoing viewing direction.

Free-Viewpoint Video Viewer

– Rendering: 30 fps (GeForce3)

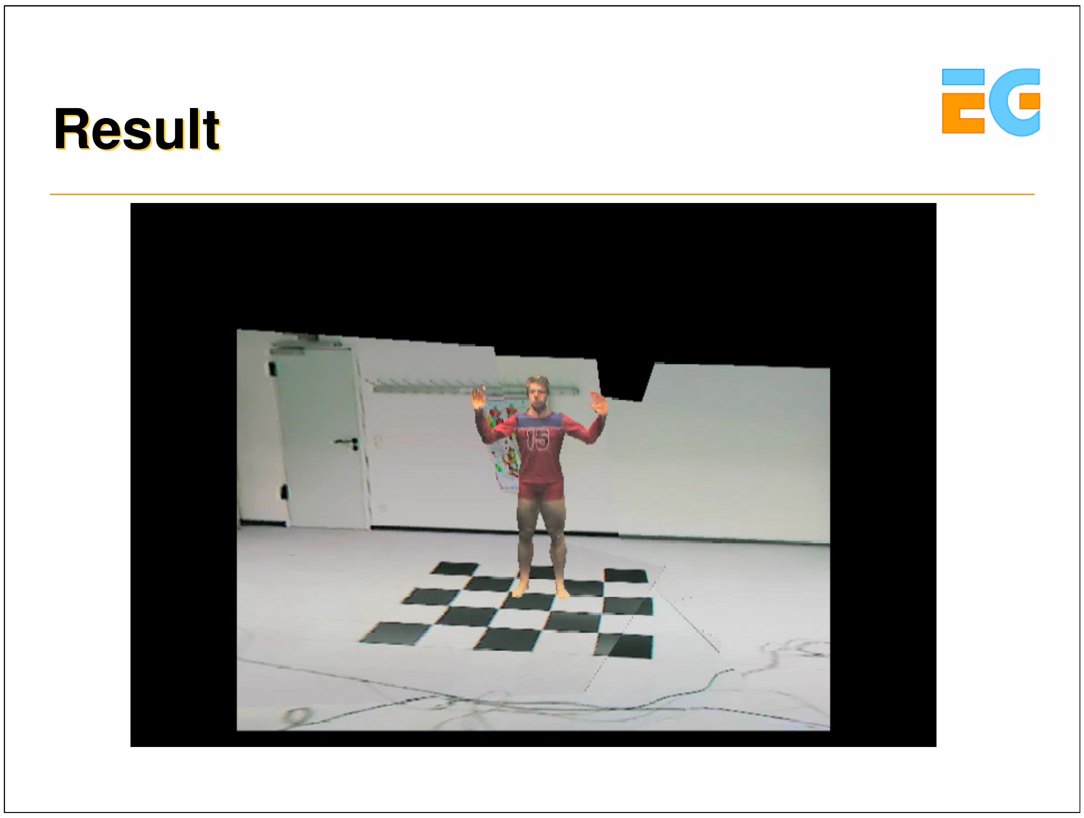




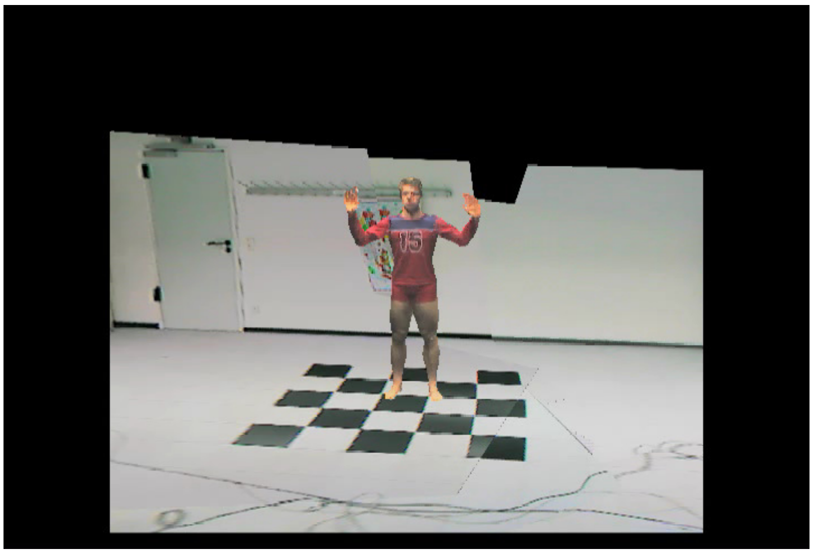
The top right image shows a screen-shot of our renderer which displays free-viewpoint videos at 30 fps on an Nvidia GeForce3 GPU.

The bottom row shows screen-shots of free-viewpoint videos that have been rendered from virtual camera views different from any input camera view.

The input video footage for these sequences (as well as all the other sequences shown on previous slides) has been captured with 8 video cameras that provide an image resolution of 320x240 pixels each.

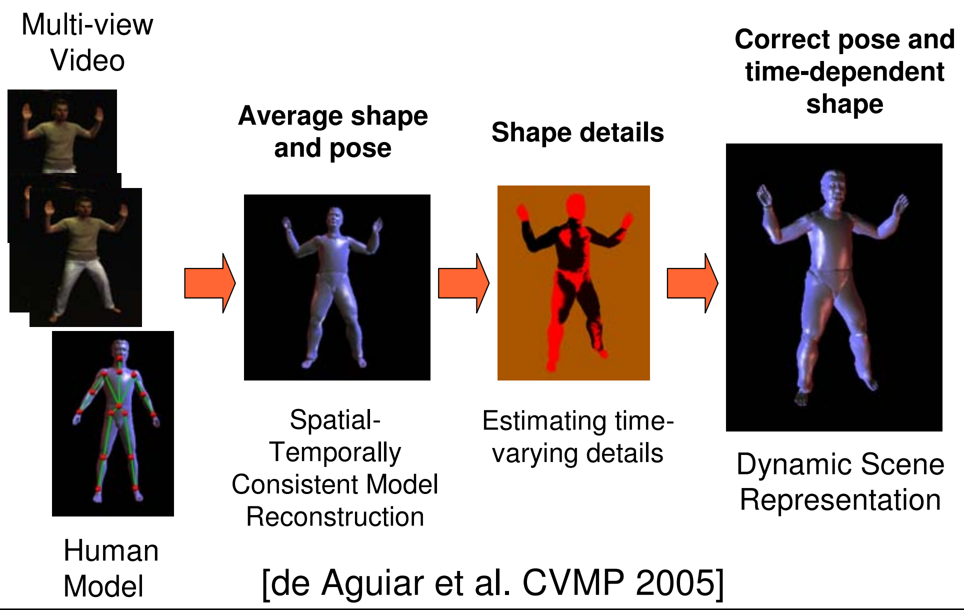


Result



A free-viewpoint video of ballet dancer that we have reconstructed with our method.

Spatio-temporal Shape Adaptation


Multi-view Video
Human Model
Average shape and pose
Spatial-Temporally Consistent Model Reconstruction
Shape details
Estimating time-varying details
Correct pose and time-dependent shape
Dynamic Scene Representation

[de Aguiar et al. CVMP 2005]

In our original pipeline, we adapted the kinematic template model to the shape of the actor in one reference pose only. A natural extension to this approach is to try to recover spatio-temporally varying scene geometry in order to handle at least the coarsest per-time-step deformations.

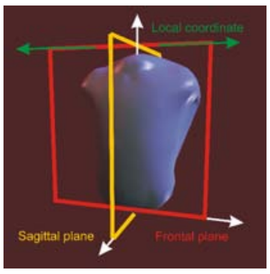
This has been explored in [4]

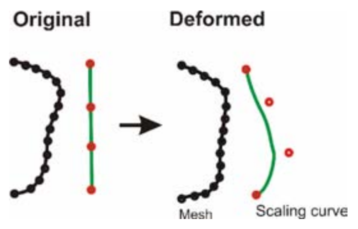
Our algorithm consists of two main steps. In the first step, a spatio-temporally consistent shape model is reconstructed by adapting the scaling parameters of the template to all video frames instead of just the reference one.

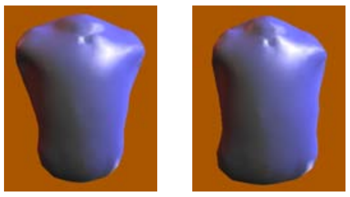
In a second step, per-time step multi-view photo-inconsistent regions are identified and their shape adjusted to recover per-time step deformations on isolated surface parts. These two main steps of the deformation method are detailed on the following slides.

STEP1: STC Model Reconstruction

- Find spline/scaling parameters that match the shape of the person „on average“ over several frames
- Iterate fitting and shape refinement over whole sequence several times
- Spatio-temporally consistent model







As briefly outlined earlier, the idea is to first reconstruct a spatio-temporally consistent shape model. This model is still a model with static shape parameters. The main difference to the original pipeline is that these parameters are now reconstructed by fitting the model to several time steps instead of just one time step. In consequence, pose estimation and shape refinement are iterated several times to yield the shape-adapted geometry.

STEP 2: Per-time Step Shape Refinement

- Diffuse surface → Identify photo-inconsistent regions [Fua and Leclerc, 1995]

- Deform vertices to
  - Increase photo-consistency
  - Preserve silhouette-consistency
  - Preserve model quality
- Find vertex displacements r_j for each vertex v_j minimizing

$$\begin{aligned} E(v_j, \vec{r_j}) = \quad & w_I E_I(v_j + \vec{r_j}) + w_S E_S(v_j + \vec{r_j}) + \\ & w_D E_D(v_j + \vec{r_j}) + w_P E_P(v_j, v_j + \vec{r_j}) \end{aligned}$$

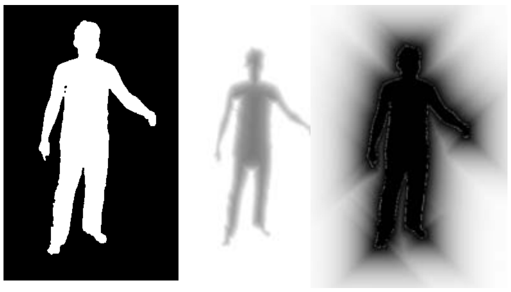
In step2, the spatio-temporally consistent model is deformed to recover per-time-step shape variations. To do so, for a set of seed vertices, optimal displacements are computed such that a multi-view consistency criterion is optimized. After multi-view consistency is optimized, the non-multi-view-consistent regions are smoothly deformed. For details on this deformation, please refer to [4]. The multi-view consistency criterion is represented by an energy functional comprising of 4 different error terms illustrated on the following slide.

STEP 2: Per-time Step Shape Refinement

- Energy terms

$w_I E_I(v_j + \vec{r_j})$ Photo-consistency

$w_S E_S(v_j + \vec{r_j})$ Silhouette distance

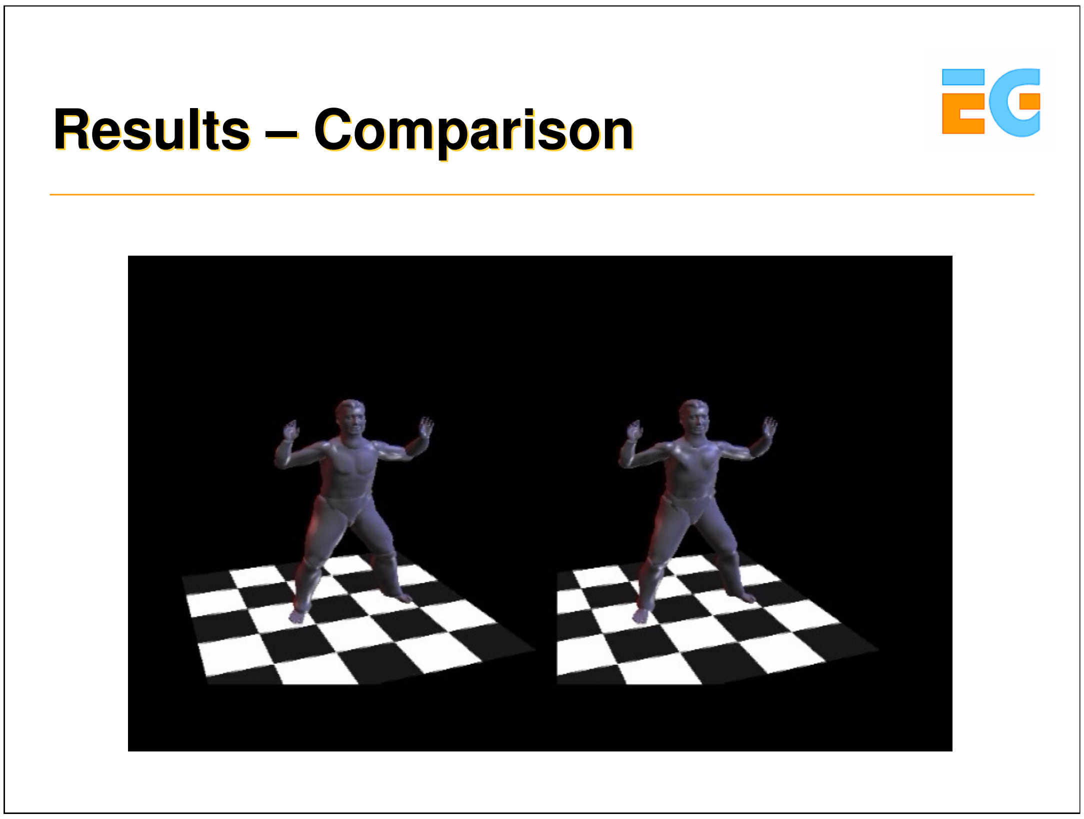


$w_D E_D(v_j + \vec{r_j})$ Triangle distortion

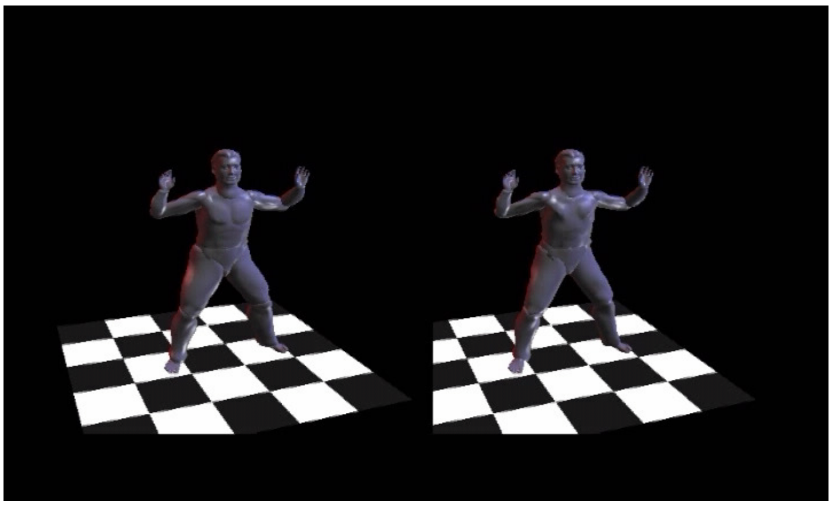
$w_P E_P(v_j, v_j + \vec{r_j})$ Change of visible camera set

The four error terms being minimized while solving for the optimal displacements for the seed vertices are shown here.

The first one measures the multi-view color consistency across all cameras in which a vertex is visible. Silhouette consistency is measured in terms of the distance (inner and outer distance field) to the silhouette boundary. Displacements that move a vertex closer to the silhouette boundary are favored.. Triangle distortion is taken into account as well to preserve shape integrity. Finally, strong displacements leading to changes in the set of cameras that see a vertex are also penalized.



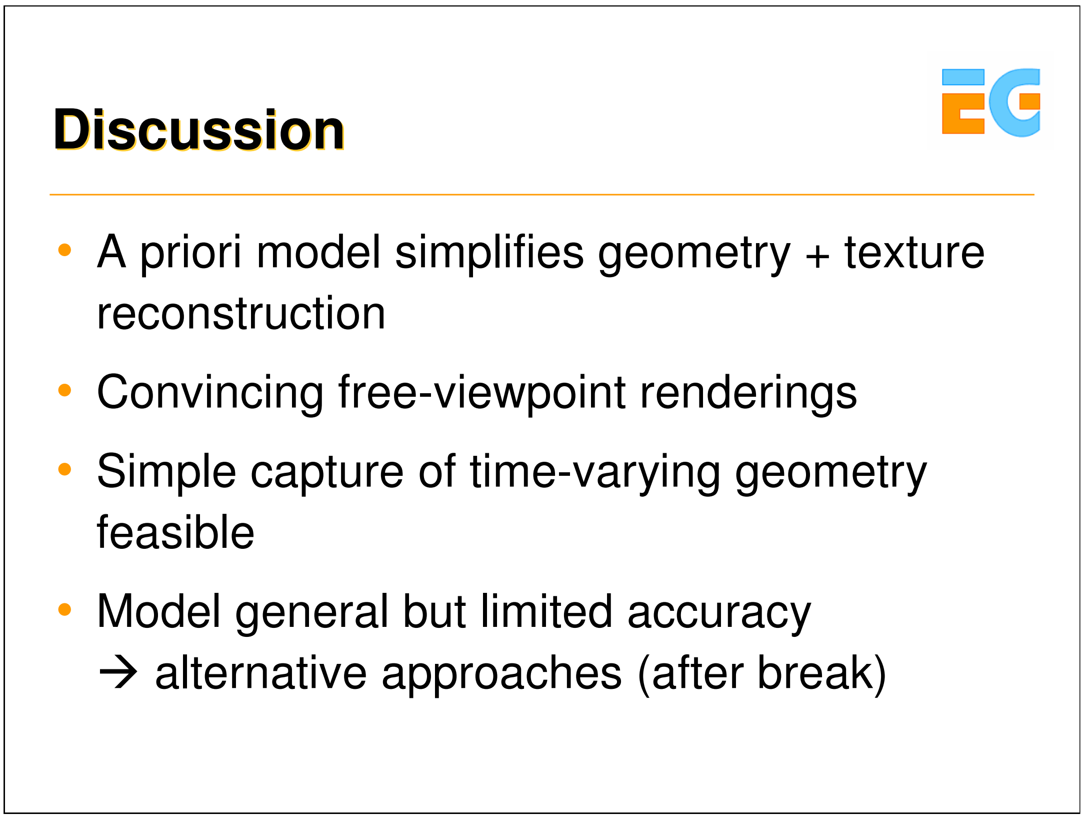
Results – Comparison



The video on this slide shows a side-by-side comparison between the captured human shape using the original pipeline, i.e. without spatio-temporal shape adaptation and the new spatio-temporally adapted human model.

The per-time step shape adaptation helps prevent certain texturing artifacts that are due to starkly incorrect shape. These inaccuracies may not be fully corrected by texture-based means only.

However, the underlying template geometry still imposes limits on how well per-time-step deformations can be captured. In the following, we will discuss novel directions in 3D video which allow us to capture more detailed dynamic model representations from input footage.



Discussion

- A priori model simplifies geometry + texture reconstruction
- Convincing free-viewpoint renderings
- Simple capture of time-varying geometry feasible
- Model general but limited accuracy
  → alternative approaches (after break)

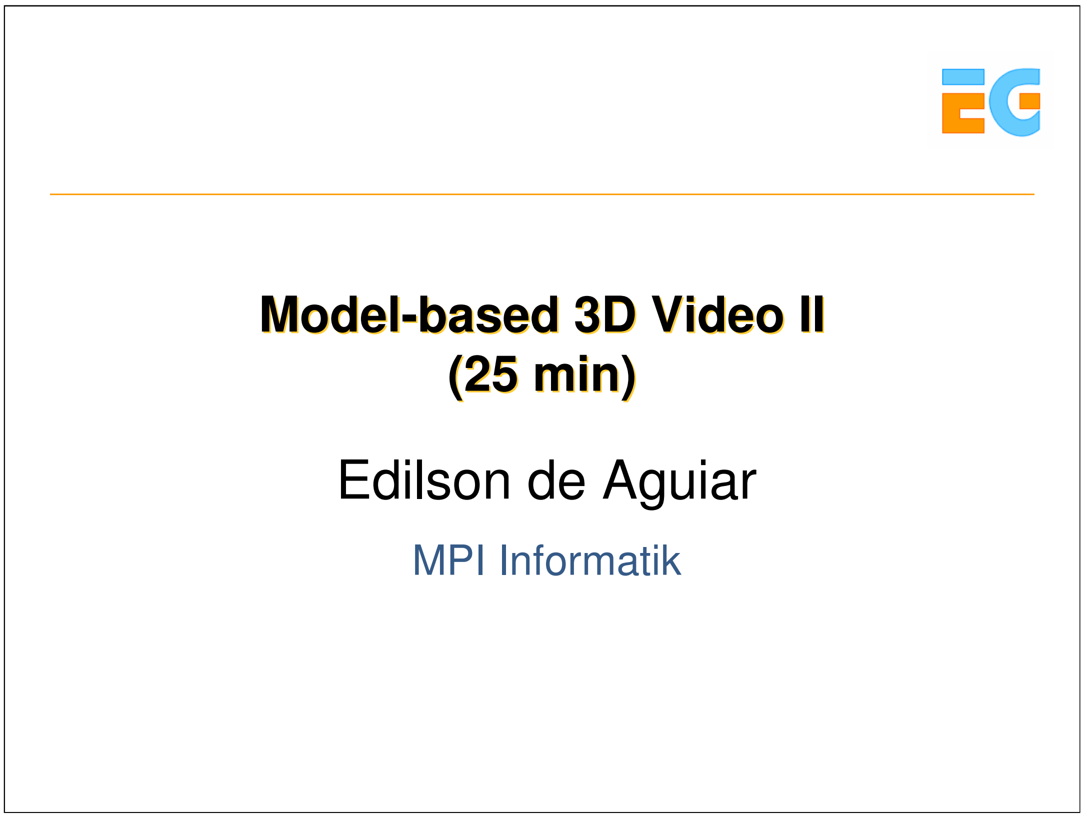
References

[1] C. Theobalt, J. Carranza, M. Magnor, H.P. Seidel, *A Parallel Framework for Silhouette-based Human Motion Capture*, in Proc. of Vision, Modeling and Visualization, p.207-214, Munich, Germany, 2003.

[2] C. Theobalt, J. Carranza, M. Magnor, H.P. Seidel, *Enhancing Silhouette-based Human Motion Capture with 3D Motion Fields*, Proc. of Pacific Graphics 2003,p.185-193, Canmore, Canada.

[3] J. Carranza, C. Theobalt. M. Magnor, H.P. Seidel, *Free-Viewpoint Video of Human Actors*. in Proc. of ACM SIGGRAPH 2003, p.569-577, San Diego, CA.

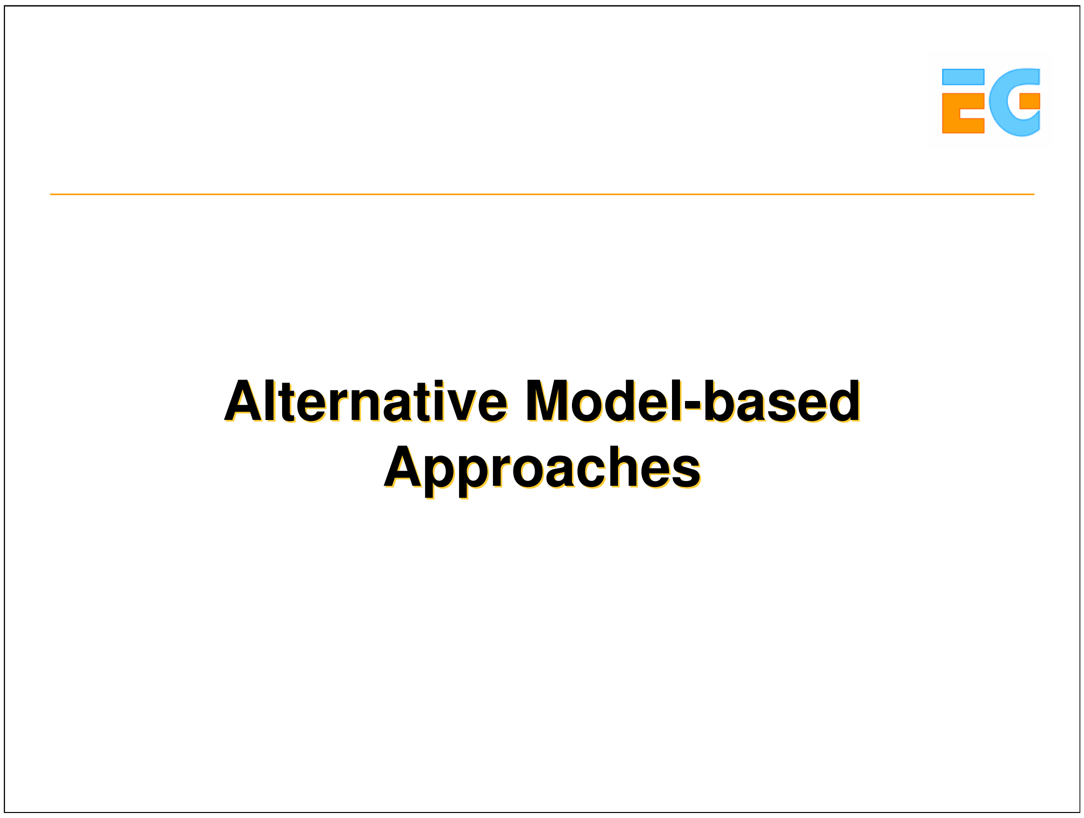
[4] E. de Aguiar, C. Theobalt, M. Magnor, H.-P. Seidel, *Reconstructing Human Shape and Motion from Multi-view video.* 2nd European Conference on Visual Media Production (CVMP), p. 42-49. London, UK. 2005.



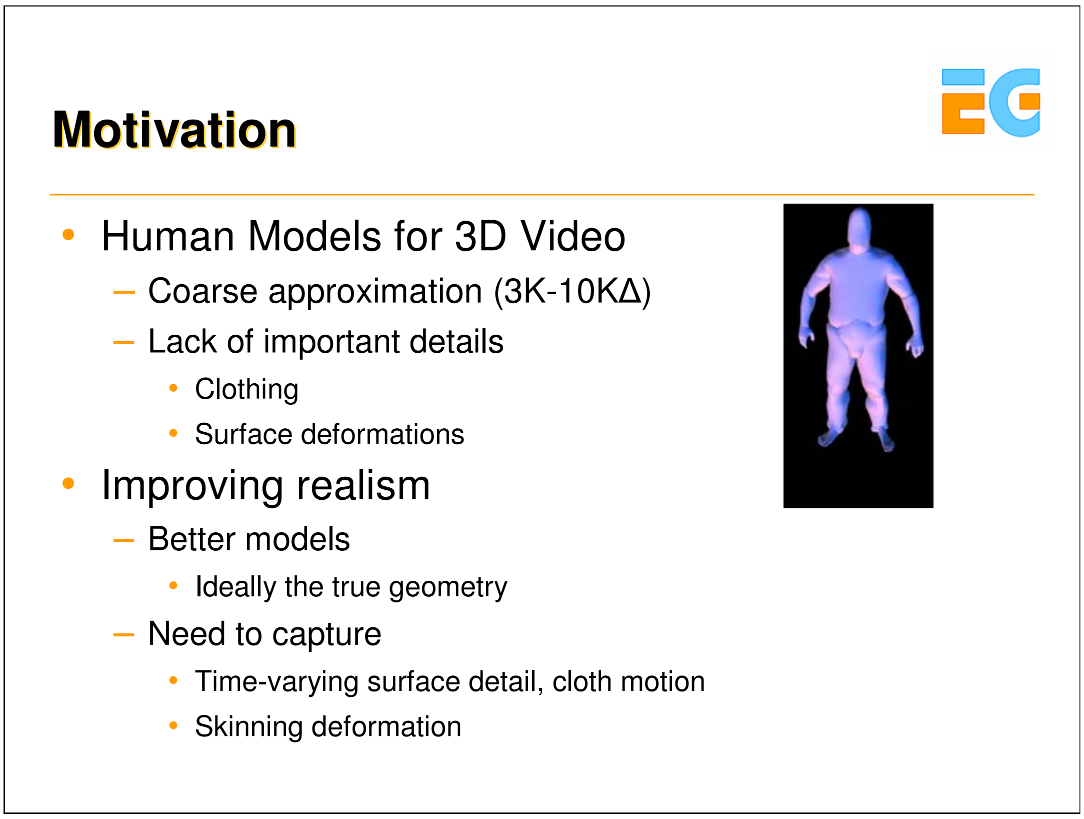
Model-based 3D Video II
(25 min)

Edilson de Aguiar

MPI Informatik



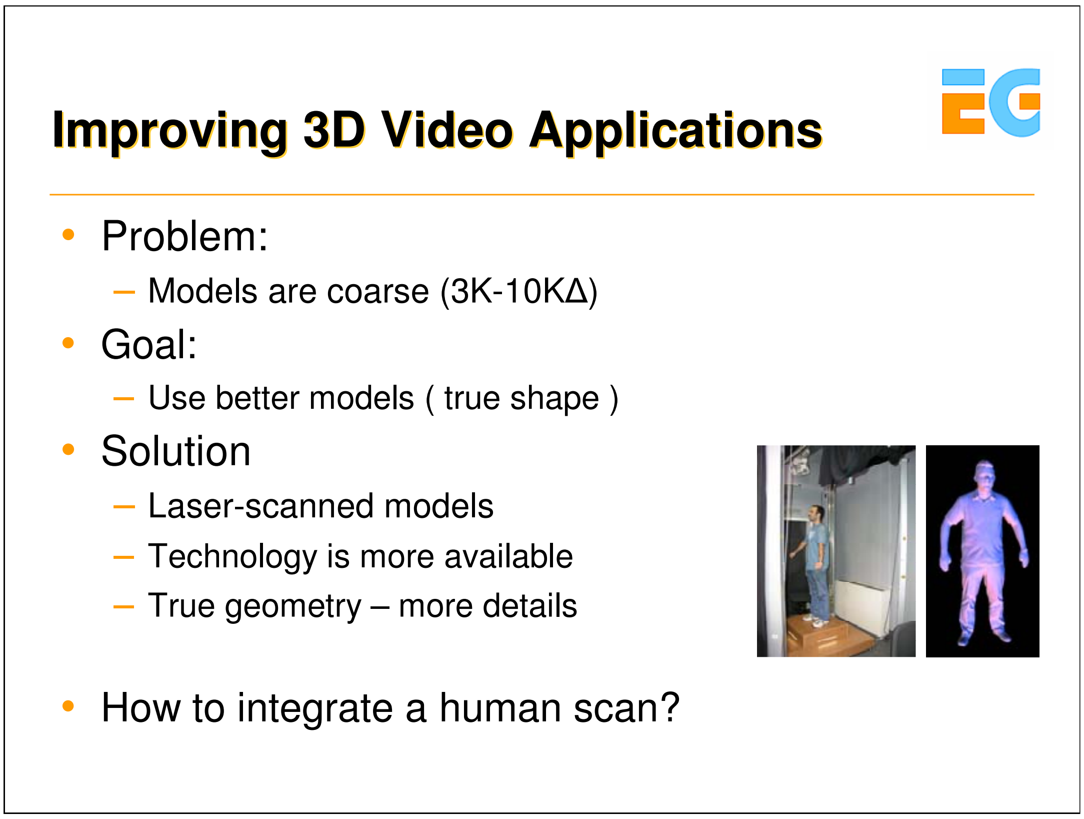
Alternative Model-based Approaches



Motivation

- Human Models for 3D Video
  - Coarse approximation (3K-10KΔ)
  - Lack of important details
    - Clothing
    - Surface deformations
- Improving realism
  - Better models
    - Ideally the true geometry
  - Need to capture
    - Time-varying surface detail, cloth motion
    - Skinning deformation

The previously presented template model for 3D video, even after reconstruction of some pert-time step deformation, was only a coarse approximation of the true geometry of the subject performing. Important details, like the appearance of the apparel that the actor is wearing or the way the skin deforms over time are not easily reproduced. To improve realism, a better representation for the subject is needed. Ideally, one would want to use the true shape of the subject as a model, since it contains all important details. In the following we will show some algorithmic ways to come closer to such a better model.

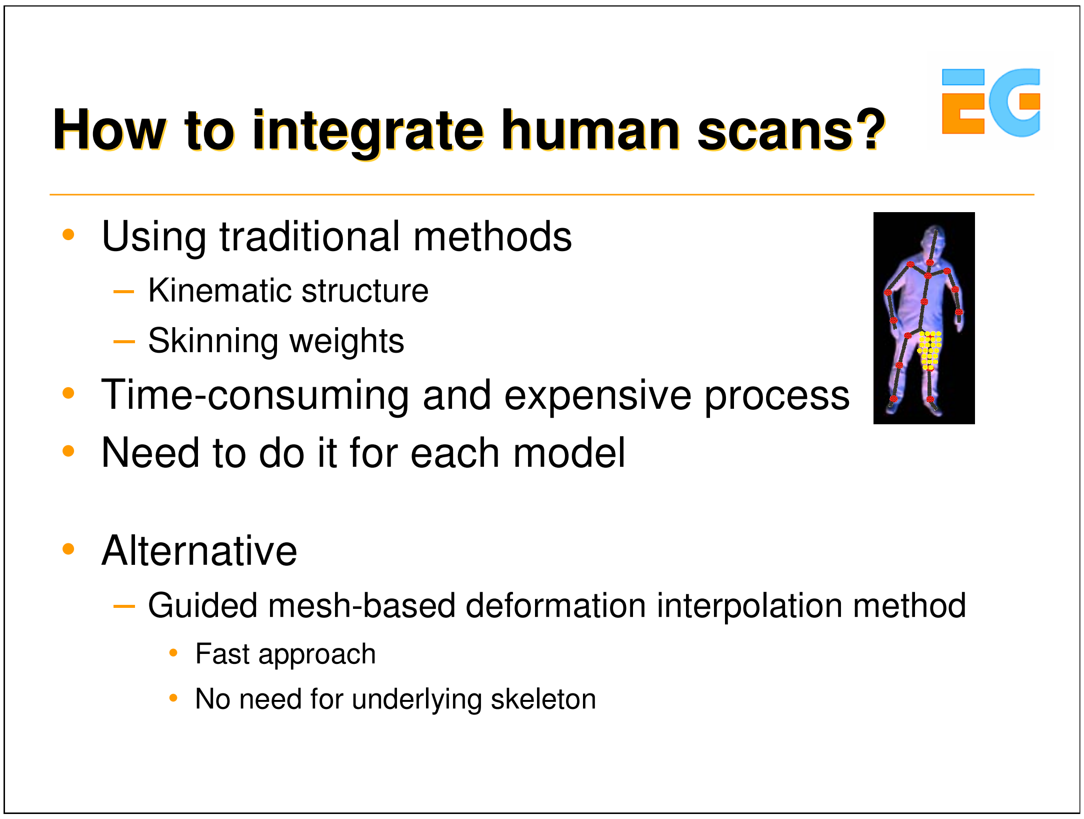


Improving 3D Video Applications

- Problem:
  - Models are coarse (3K-10KΔ)
- Goal:
  - Use better models (true shape)
- Solution
  - Laser-scanned models
  - Technology is more available
  - True geometry – more details

- How to integrate a human scan?

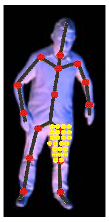
Since current models used in 3D Video applications are relatively coarse, in order to improve realism a better representation for the subject is needed. Recently, laser scanning technology is becoming cheaper and easier available. The main advantage is that a laser scanner can easily capture the true shape of the subject, including most of the important surface detail at fine resolution. By incorporating such better input models in current 3D Video systems one would expect a considerable amount of improvement in terms of quality and realism.



How to integrate human scans?

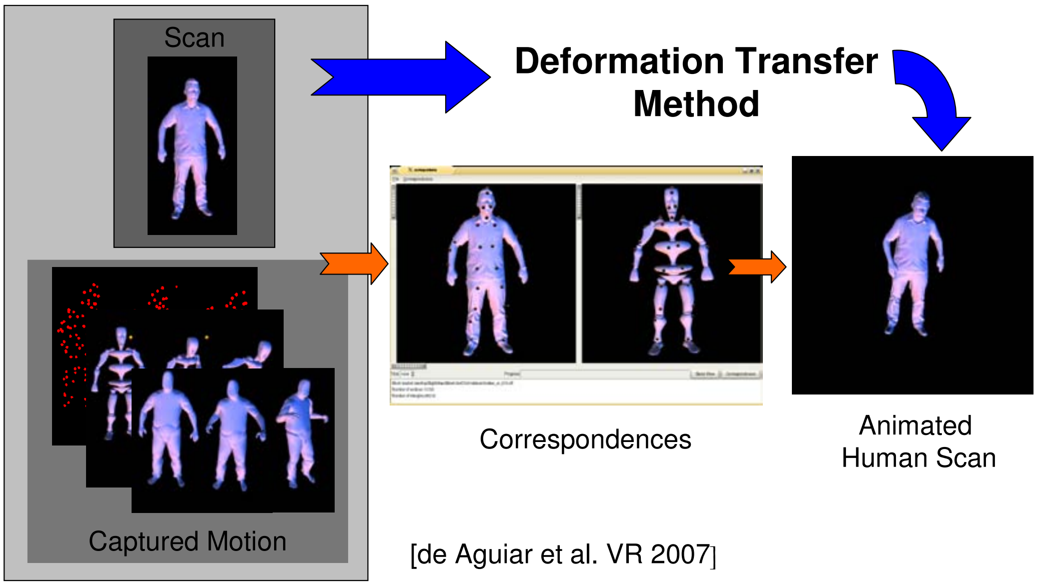
- Using traditional methods
  - Kinematic structure
  - Skinning weights
- Time-consuming and expensive process
- Need to do it for each model

- Alternative
  - Guided mesh-based deformation interpolation method
    - Fast approach
    - No need for underlying skeleton



Traditional 3D Video systems are mostly built around models with an underlying kinematic structure. If we want to replace the model with the coarse surface geometry by a laser-scanned model, we need to fit a kinematic structure to it and thereafter determine the skinning weights for the vertices in order to correctly deform it during tracking or animation. This process is time-consuming and expensive being at best a semi-automatic procedure. Another limitation is that the same procedure should be applied to each new scanned model. An alternative to this expensive process is a mesh-based deformation scheme, which is efficient and does not require the specification of any underlying skeleton structure for each particular subject.

Overview

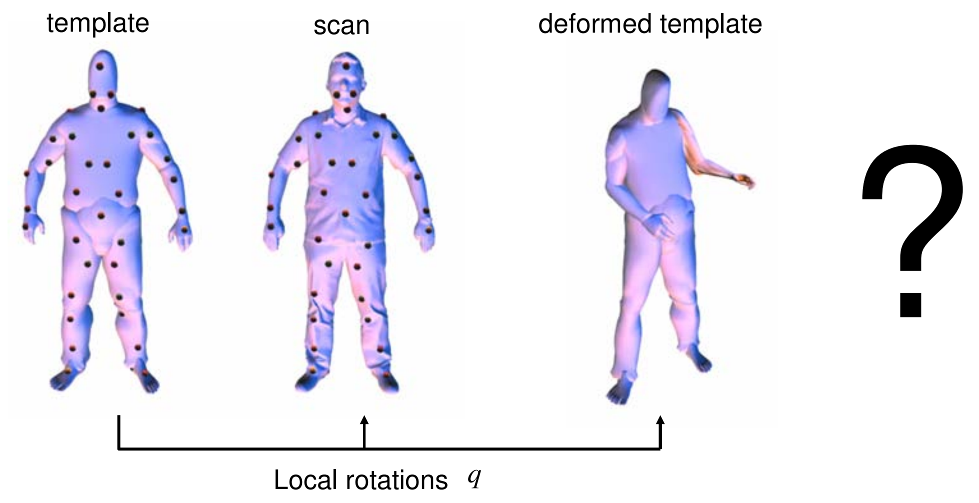


A simple and easy way to integrate a laser-scanned human in the traditional pipeline of a model-based 3D Video system can be achieved through a mesh-based Laplacian deformation scheme [1]. Having as inputs a scanned model of the subject and a description of her motion (for instance captured with the original marker-free approach described in part I of the model-based section), the main goal is to transfer the input motion to the scanned model. By formulating the motion transfer problem as a deformation transfer problem and by setting a small amount of corresponding marked vertices between both models (template and scan), an automatic guided deformation interpolation technique can be used to animate the scanned human model efficiently.

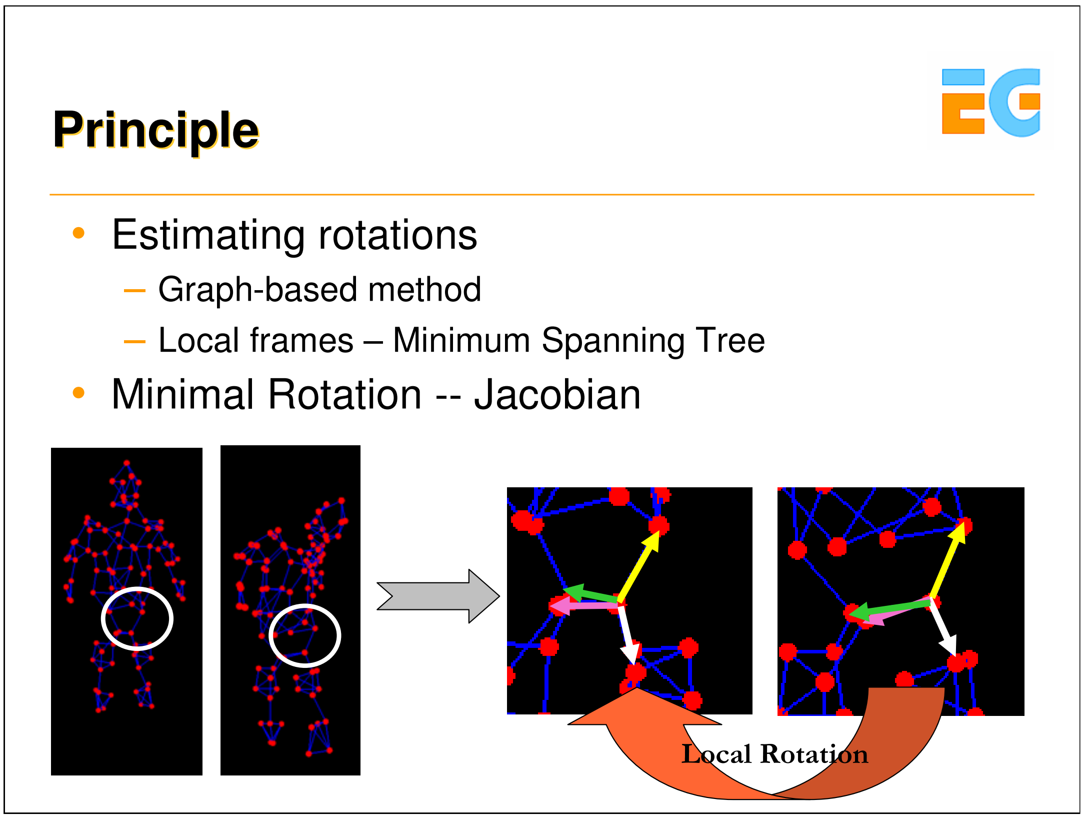
Reference:

[1] E. de Aguiar, C. Theobalt, C. Stoll and H.-P. Seidel, *Rapid Animation of Laser-scanned Humans.* In Proc. of IEEE Virtual Reality 2007, pp. 223-226, Charlotte, USA.

Principle

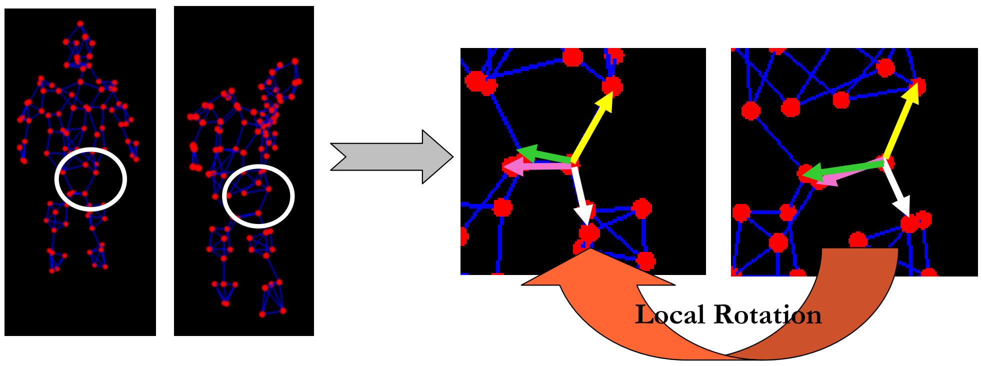


After first automatically aligning both models (template and scan) in a given reference pose, the user marks a set of vertices on the template and assigns to each of them a corresponding vertex in the scanned model. Through placement of markers the characteristics of the motion and the surface skinning are defined, but also retargeting constraints can be set. For each time instant of the input motion sequence, represented by a deformed template, it is our goal to transfer this relative deformation to the human scan, bringing it to the same pose as the deformed template. A good solution is to use a Laplacian mesh deformation scheme that jointly employs rotational and positional constraints on the markers to compute the new sequence of poses for the human scan. This method is divided in three steps and it is performed for each time step of the input motion sequence. In the first step, local rotations for all markers belonging to the scan are estimated from the rotations of the corresponding markers on the template between its reference pose and its pose at the current time step.



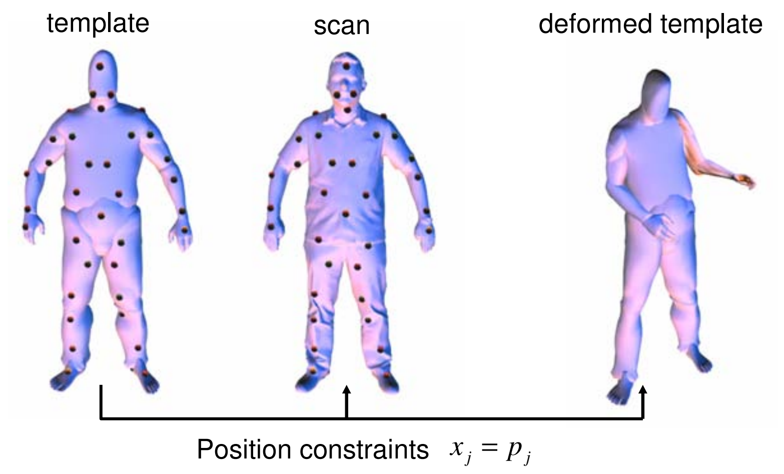
Principle

- Estimating rotations
  - Graph-based method
  - Local frames – Minimum Spanning Tree
- Minimal Rotation -- Jacobian



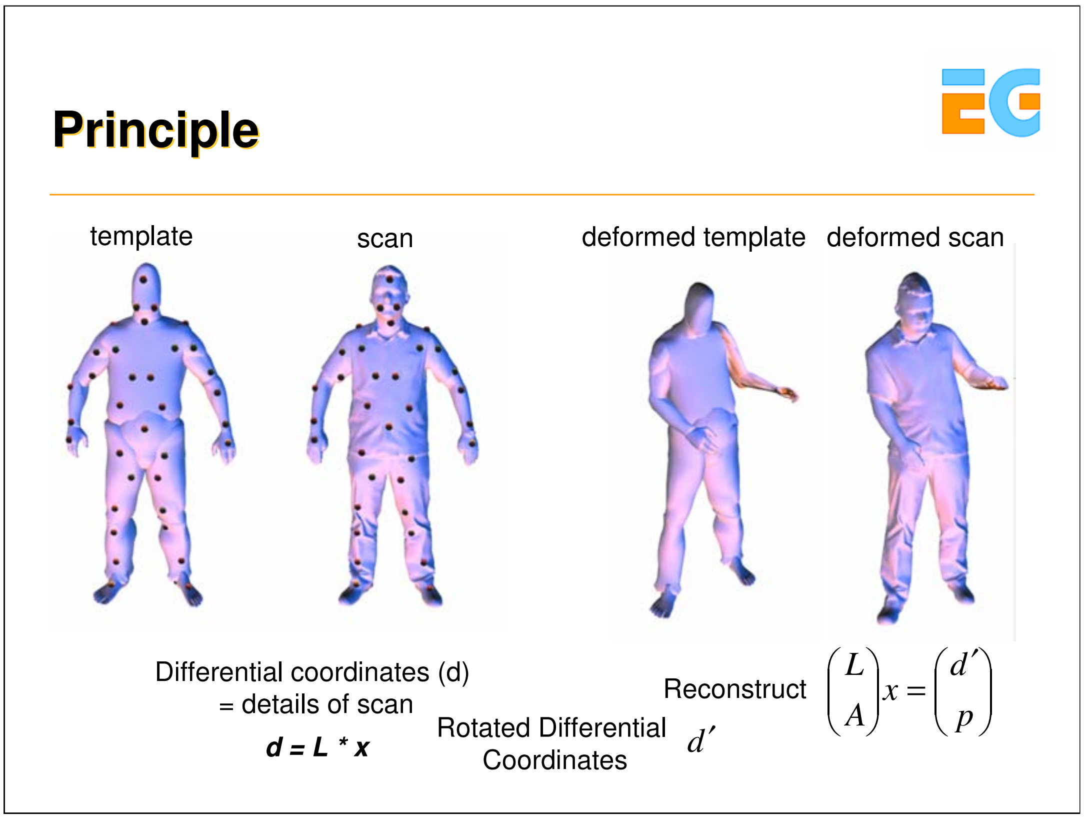
A local rotation for each marker belonging to the scan can be calculated from the rotation of the corresponding markers on the template between its reference pose and its pose at a different time by means of a graph-based method. Template markers can be considered as nodes in a graph and edges between them can be determined by constructing the minimal spanning tree. For each marker, the minimal rotation that makes its outgoing edges at the reference time matches its outgoing edges at time t can be found. Thereafter, local rotations are converted to quaternions and assigned to the corresponding partners in the human scan.

Principle

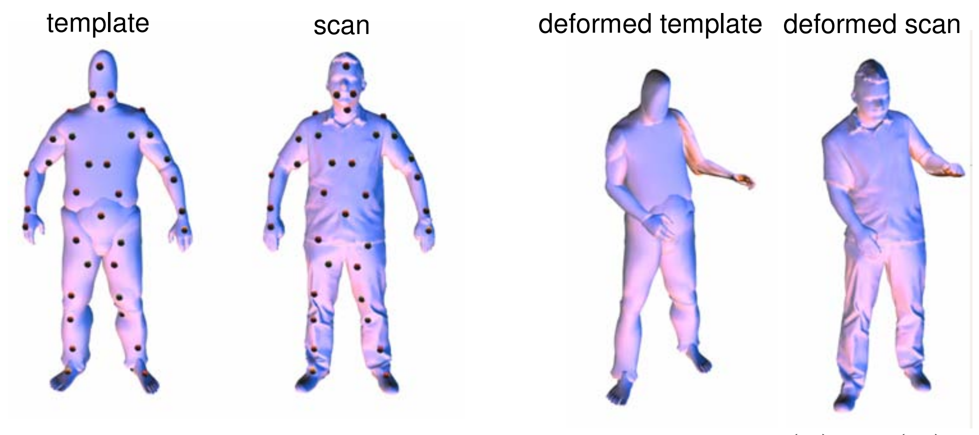


Rotation interpolation via Harmonic fields: ***L * q = 0***

In the second step, rotations are interpolated over the scanned mesh. Regarding each component of a quaternion as a scalar field defined over the entire mesh, a smooth interpolation is guaranteed by regarding these scalar fields as harmonic fields. The interpolation is performed efficiently by solving the Laplace equation (Lq = 0) over the whole mesh with constraints at the marked vertices. In the equation, L is the discrete Laplace operator based on the cotangent-weights. Position constraints for the Laplacian deformation scheme are calculated from the displacements of the corresponding markers on the template between its reference pose and its pose at the current time step.



Principle



Differential coordinates (d)
= details of scan

d* = *L* * *x

Rotated Differential Coordinates d'

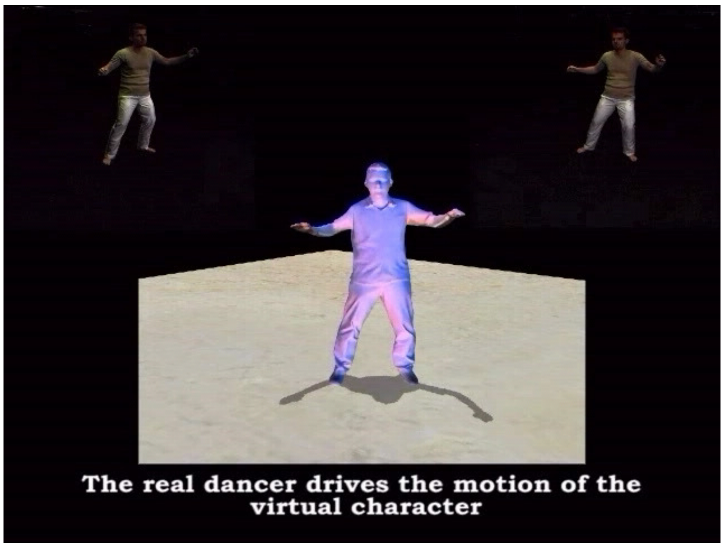
Reconstruct $\begin{pmatrix} L \\ A \end{pmatrix} x = \begin{pmatrix} d' \\ p \end{pmatrix}$

The differential coordinates d are computed once at the beginning of the sequence by solving the given linear system. In the last step, the vertex positions of the scanned model are reconstructed such that it best approximates the rotated differential coordinates, as well as the positional constraints. This can be formulated as a least-squares problem and transformed into a linear system. During the whole process the Laplacian matrix L does not change, which allows a sparse matrix decomposition and execution of only back substitution for each frame.

Marker-less Motion Capture Results

- Skinning + implicit deformation

5 FPS



Using the silhouette-based marker-free motion capture system described in part I and [2] that employs a coarse template body model, only eight video cameras are needed to capture the subject's motion. Since the template is already used for tracking, the proposed algorithm is straightforwardly applied to map the captured motion to scanned models. The video shows that this method can accurately transfer poses captured on video to scanned models of even different subjects.

Reference:

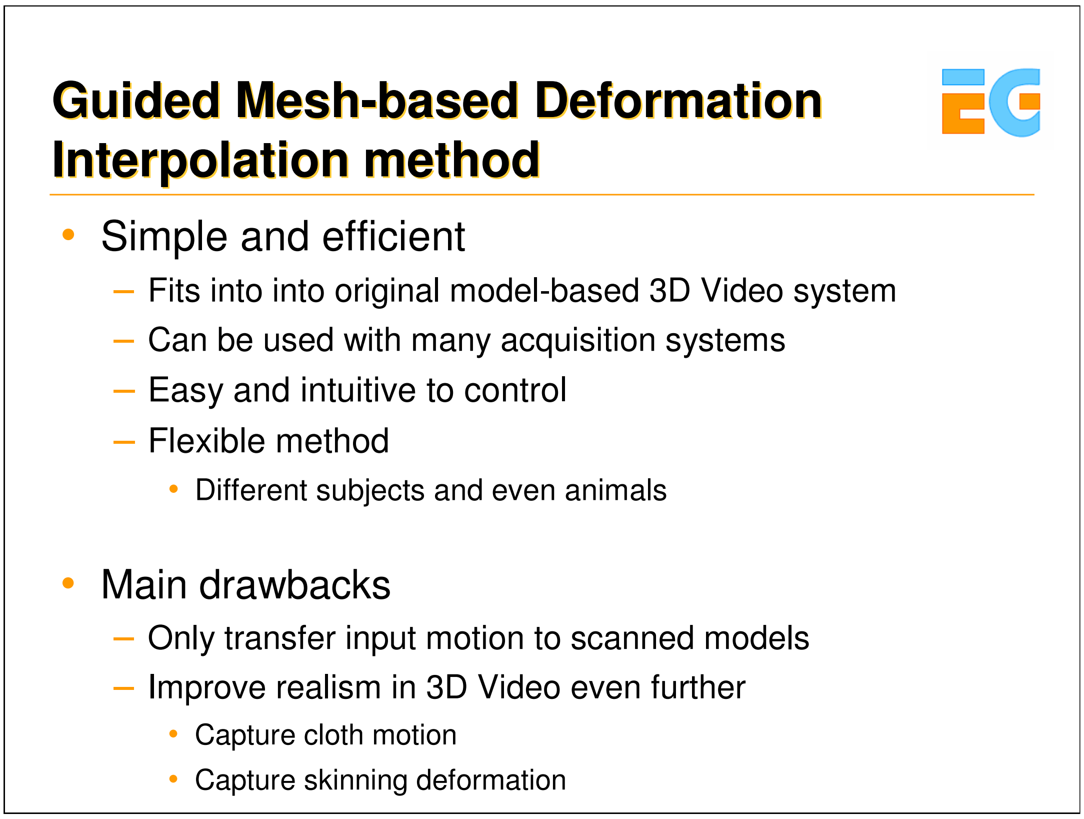
[2] J. Carranza, C. Theobalt. M. Magnor, H.P. Seidel, *Free-Viewpoint Video of Human Actors*. In Proc. of ACM SIGGRAPH 2003, p.569-577, San Diego, CA

Also useful for Character Animation

Marker-based MoCap data can also be used to animate avatars

[MoCap data from the Eyes Japan Co. Ltd. database]

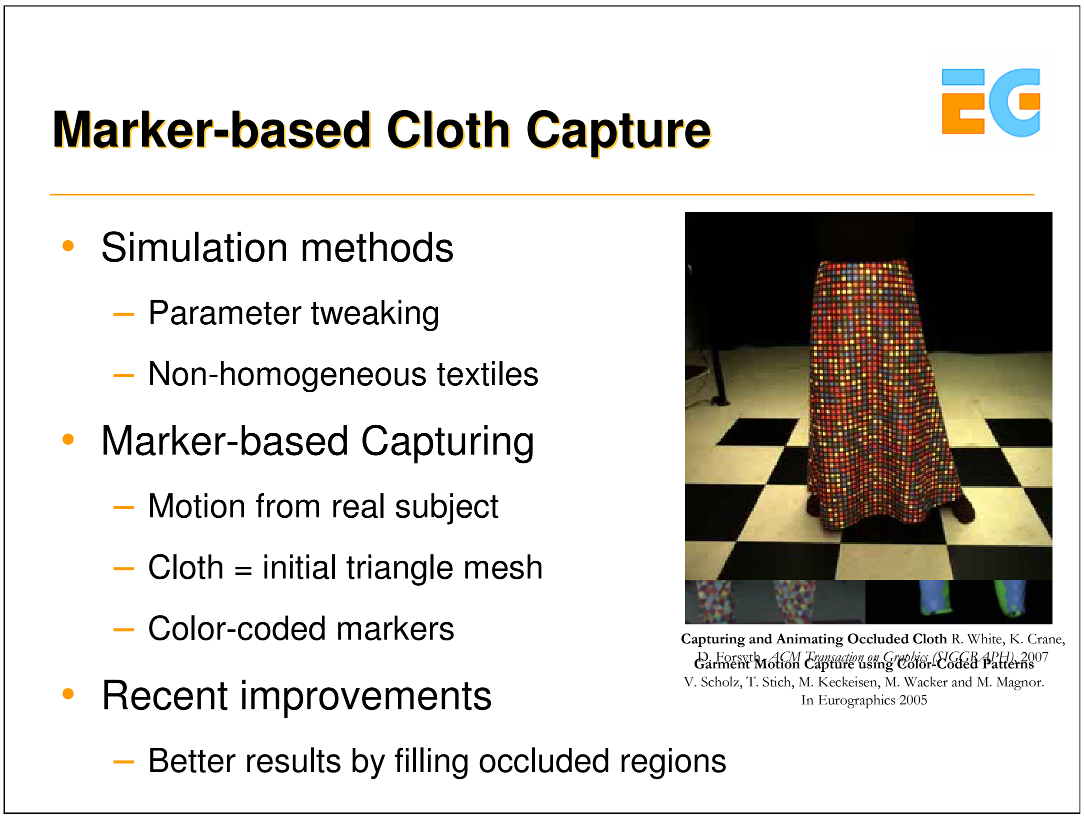
There are also many applications for the proposed method in fields related to 3D video, such as computer animation. Nowadays, optical motion capture data is presumably amongst the most widely-used motion descriptions in animation production. By using the presented method, MoCap data can be easily used to animate high-quality surface models while bypassing the drawbacks of the traditional animation pipeline. After transforming the MoCap data into a moving template model (straightforwardly done by transforming the actual bone skeleton into a triangle mesh using any standard animation software), the guided deformation interpolation method can be applied to produce the animation. Note that even raw marker-trajectories can be used as input to this method. The video shows that the scanned model realistically performs the motion while exhibiting lifelike non-rigid surface deformations. Motion retargeting is feasible by appropriately placing constraints.



Guided Mesh-based Deformation Interpolation method

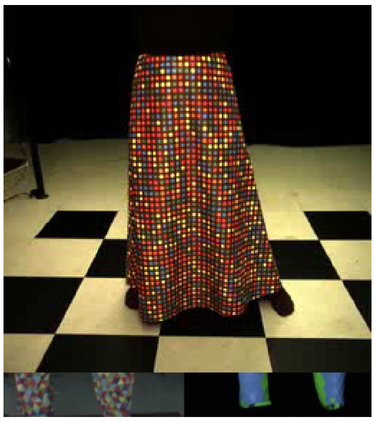
- Simple and efficient
  - Fits into into original model-based 3D Video system
  - Can be used with many acquisition systems
  - Easy and intuitive to control
  - Flexible method
    - Different subjects and even animals
- Main drawbacks
  - Only transfer input motion to scanned models
  - Improve realism in 3D Video even further
    - Capture cloth motion
    - Capture skinning deformation

The guided mesh-based deformation interpolation approach is a flexible, simple and fast scheme that allows the integration of high-quality scanned models into the pipeline of conventional 3D Video systems and consequently a considerable amount of improvement in terms of quality and realism. The method requires a minimum of manual interaction, it is flexible, easy and intuitive to use, and simultaneously solves the animation, the surface deformation, and the motion retargeting problems. Although this method allows the integration of the true shape into the pipeline, it is not able to correctly reproduce the temporal behavior of the garments and the deformation of the skin while the subject is performing, in case the input motion description does not have this information. In order to achieve this, new methods will be presented.



Marker-based Cloth Capture

- Simulation methods
  - Parameter tweaking
  - Non-homogeneous textiles
- Marker-based Capturing
  - Motion from real subject
  - Cloth = initial triangle mesh
  - Color-coded markers
- Recent improvements
  - Better results by filling occluded regions



Capturing and Animating Occluded Cloth R. White, K. Crane, D. Forsyth, *ACM Transaction on Graphics (SIGGRAPH)*, 2007

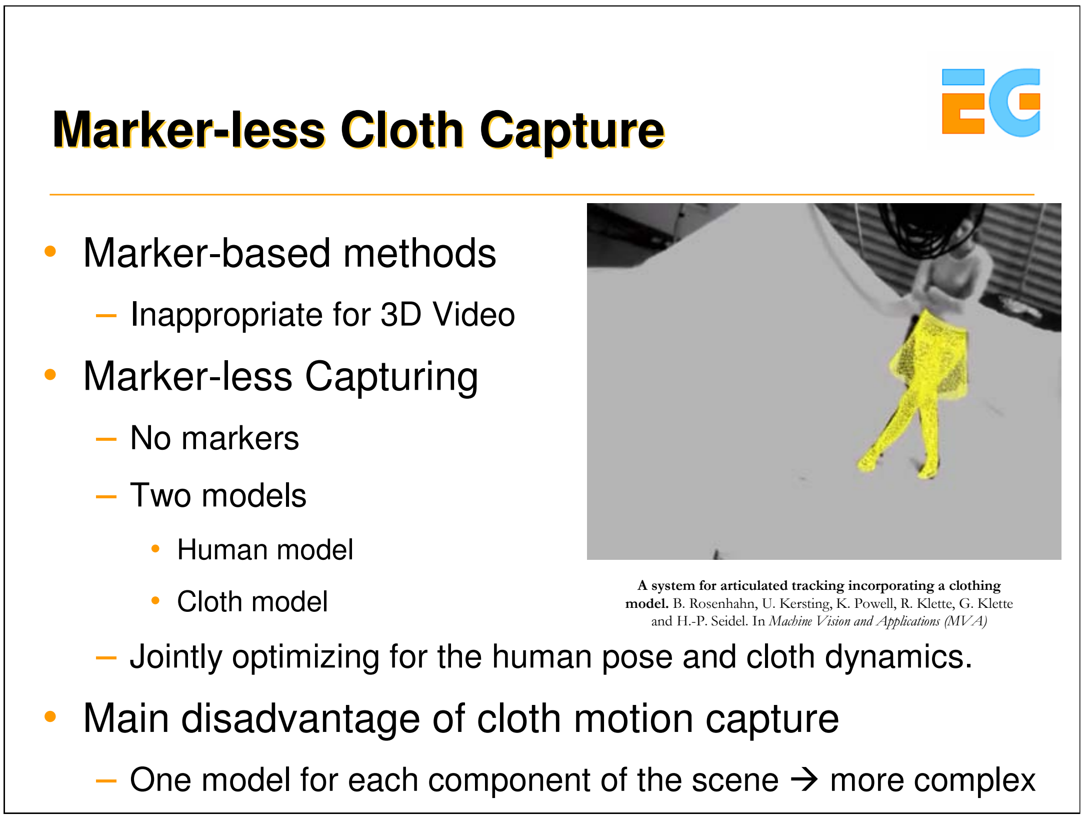
Garment Motion Capture using Color-Coded Patterns V. Scholz, T. Stich, M. Keckeisen, M. Wacker and M. Magnor. In Eurographics 2005

As pointed out earlier, capturing the motion of garment is a big challenge in 3D video reconstruction. In computer graphics, the behavior of the garments while the subject is moving is usually simulated. However, tweaking the parameters to achieve a particular look is fairly difficult and numerical instabilities may frequently happen. In order to reproduce garment motion more faithfully, marker-based capturing techniques [3] were developed that use color-coded markers and a video camera acquisition setup to capture true cloth deformation, thereby allowing for a higher animation quality. While this allows for very detailed reconstruction, traditional multi-view acquisition problems, like occlusion, make the reconstruction still a hard problem. Recently, a new approach explicitly addressed the latter problem and fills in occluded regions of the tracked garment [4].

References:

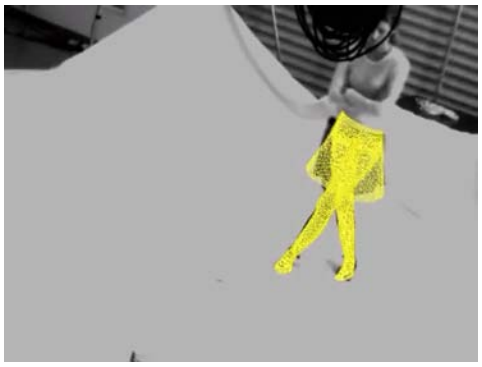
[3] V. Scholz, T. Stich, M. Keckeisen, M. Wacker and M. Magnor, Garment Motion Capture Using Color-Coded Patterns, *Computer Graphics Forum (special issue* Eurographics 2005*)*, vol. 24, no. 3, pp. 439-448, August 2005.

[4] R. White, K. Crane, D. Forsyth, Capturing and Animating Occluded Cloth, *ACM Transaction on Graphics (SIGGRAPH)*, 2007.



Marker-less Cloth Capture

- Marker-based methods
  - Inappropriate for 3D Video
- Marker-less Capturing
  - No markers
  - Two models
    - Human model
    - Cloth model
  - Jointly optimizing for the human pose and cloth dynamics.
- Main disadvantage of cloth motion capture
  - One model for each component of the scene → more complex



A system for articulated tracking incorporating a clothing model. B. Rosenhahn, U. Kersting, K. Powell, R. Klette, G. Klette and H.-P. Seidel. In *Machine Vision and Applications (MVA)*

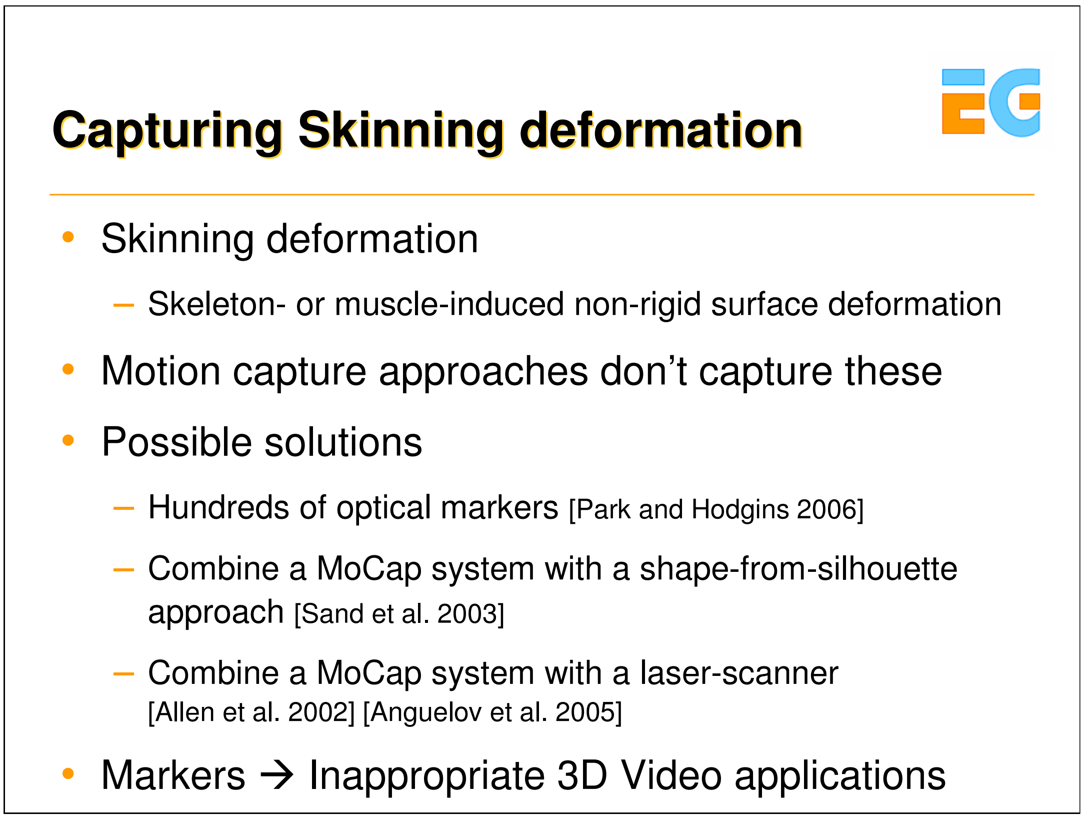
While marker-based cloth motion capture methods are able to generate good results for animation purposes, the use of the color-coded markers make them inappropriate for 3D Video applications. Instead, marker-less methods have been developed [5] where no intrusion in the scene is necessary. The behavior of the cloth while the subject is moving can be captured by using a template human model (with underlying kinematics structure) and a template cloth model. By jointly optimizing both models at each frame of the video footage, these methods are able to capture the deformation of the cloth, e.g. wrinkles. After that, simulation methods [6] or mesh deformation methods [1] can also be used for transferring the motion to different clothing styles.

Despite some very good capturing results on specific types of scenes, cloth motion capture approaches bear the disadvantage that a separate type of model and often a separate type of tracking method has to be used in combination with the already difficult body motion tracking itself. This greatly reduces the flexibility of the method.

References:

[5] B. Rosenhahn, U. Kersting, K. Powell, R. Klette, G. Klette and H.-P. Seidel. A system for articulated tracking incorporating a clothing model *Machine Vision and Applications (MVA)* Vol. 18, No. 1, pp. 25-40

[6] N. Hasler, B. Rosenhahn, H.-P. Seidel: Reverse Engingeering Garments, *Mirage 2007*, pages 200-211, Rocquencourt, France, 2007.



Capturing Skinning deformation

- Skinning deformation
  - Skeleton- or muscle-induced non-rigid surface deformation
- Motion capture approaches don't capture these
- Possible solutions
  - Hundreds of optical markers [Park and Hodgins 2006]
  - Combine a MoCap system with a shape-from-silhouette approach [Sand et al. 2003]
  - Combine a MoCap system with a laser-scanner [Allen et al. 2002] [Anguelov et al. 2005]
- Markers → Inappropriate 3D Video applications

Another category of related approaches tries to infer geometry deformation at a slightly reduced level of complexity, however for the whole body. Skinning is a technique used in computer animation to correctly reproduce non-rigid surface deformation around joints or in the vicinity of bulging muscles.

Traditional motion capture systems are not able to measure such time-varying body shape deformations and therefore their acquisition principle has to be augmented. Some methods use hundreds of optical markings for deformation capture [7], or combine a motion capture system with a range scanner [8,9] or a shape-from-silhouette approach [10] to measure a model of skinning deformation in dependence on skeletal pose parameters. Although achieving good results, most of these marker-based methods require active interference with the scene which makes them inappropriate for 3D Video applications. Also, skinning is a very reduced form to describe surface deformations and it is again inappropriate to model the surface deformation of people wearing wide apparel.

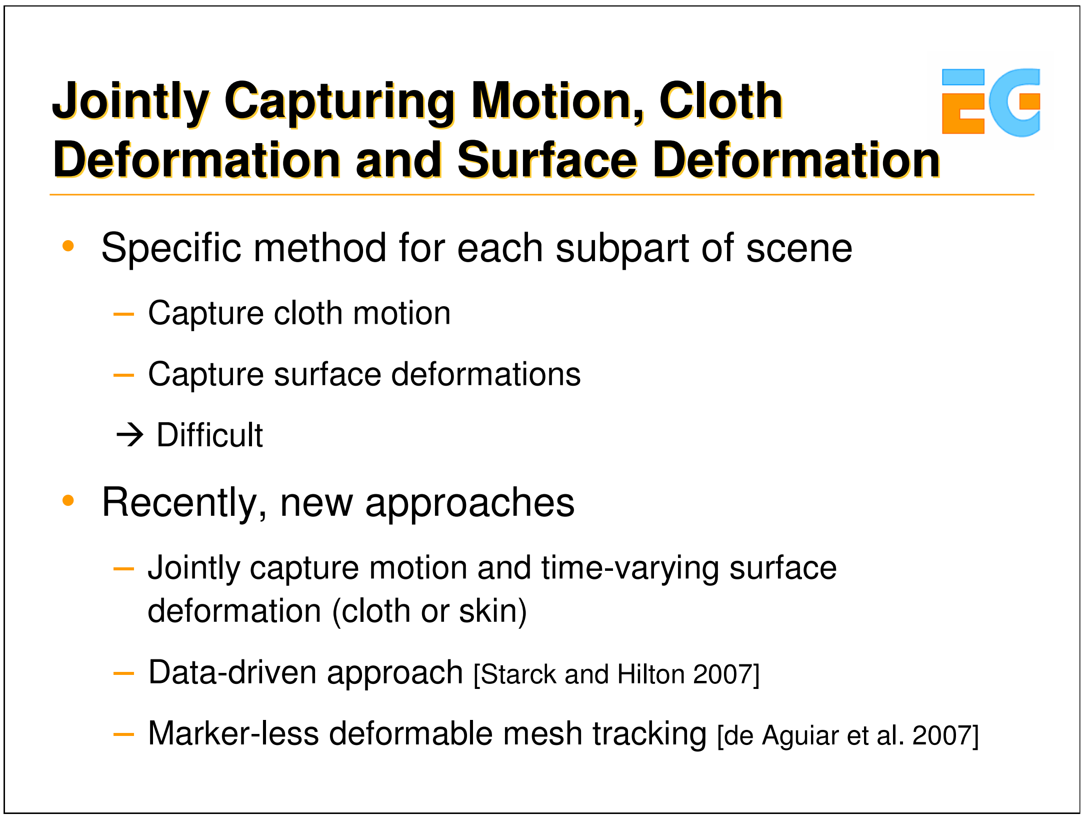
References:

[7] S. I. Park and J. K. Hodgins: Capturing and Animating Skin Deformation in Human Motion, *ACM Transactions on Graphics*, 25(3): 881-889 (2006)

[8] B. Allen, B. Curless, and Z. Popovic. Articulated body deformation from range scan data. *ACM Transactions on Graphics (ACM SIGGRAPH 2002)*, *21*, 3, 612-619.

[9] D. Anguelov, P.Srinivasan, D.Koller, S.Thrun, J. Rodgers, J.Davis. SCAPE: Shape Completion and Animation of People. Proceedings of the *SIGGRAPH Conference*, 2005.

[10] P. Sand, L. McMillan, and J. Popovic. Continuous Capture of Skin Deformation. *ACM Transactions on Graphics (TOG)* 22, 3, 578-586.



Jointly Capturing Motion, Cloth Deformation and Surface Deformation

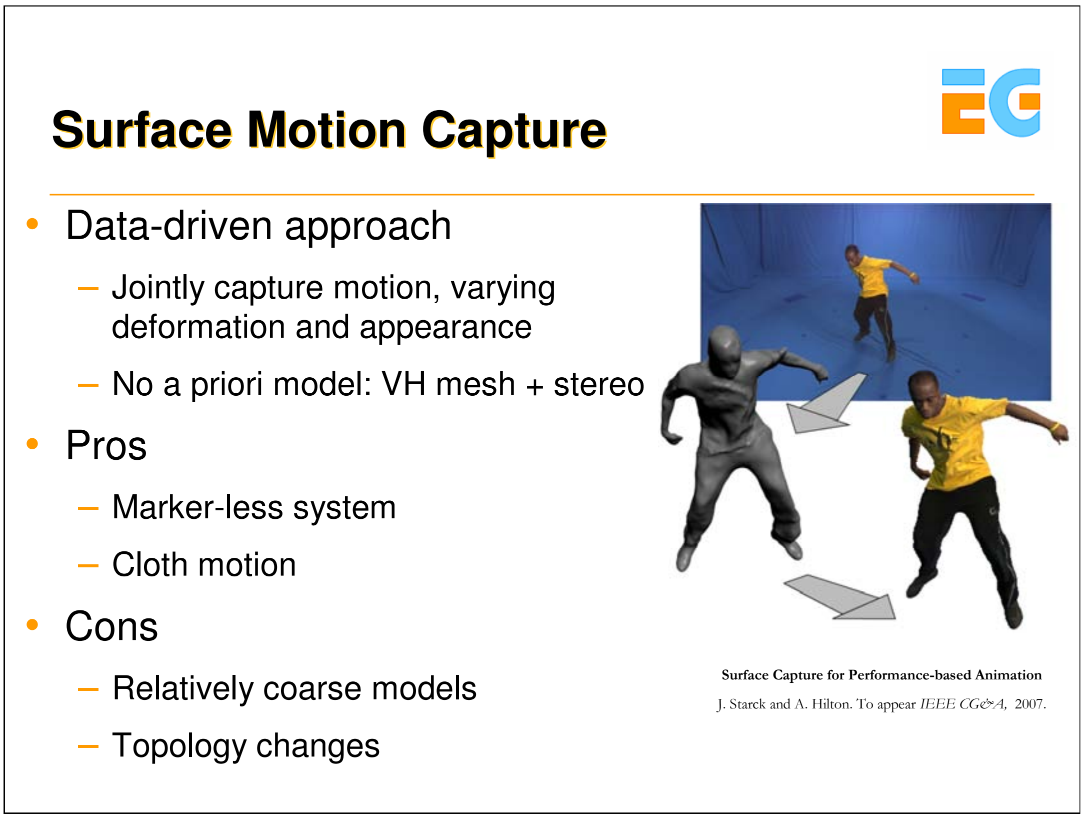
- Specific method for each subpart of scene
  - Capture cloth motion
  - Capture surface deformations

  → Difficult
- Recently, new approaches
  - Jointly capture motion and time-varying surface deformation (cloth or skin)
  - Data-driven approach [Starck and Hilton 2007]
  - Marker-less deformable mesh tracking [de Aguiar et al. 2007]

Capturing the time-varying shape of a scene featuring all components for which we previously showed individual capturing methods is a difficult task. It would be one option to jointly use all of the given methods in a single scene. However, this may be difficult in practice and not very easy to implement.

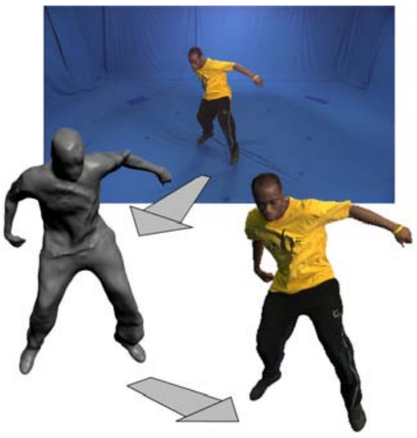
Recently new approaches have been developed that aim at jointly capturing motion and time-varying surface deformations even for subjects wearing wide apparel with complex shape. Most of these methods use a fairly general scene representation, for instance only a 3D triangle mesh instead of dedicated models for each component. By this means, the reconstruction principle remains the same, regardless of the complexity and assembly of the scene.

In the following, we will briefly look at two of these approaches.



Surface Motion Capture

- Data-driven approach
  - Jointly capture motion, varying deformation and appearance
  - No a priori model: VH mesh + stereo
- Pros
  - Marker-less system
  - Cloth motion
- Cons
  - Relatively coarse models
  - Topology changes



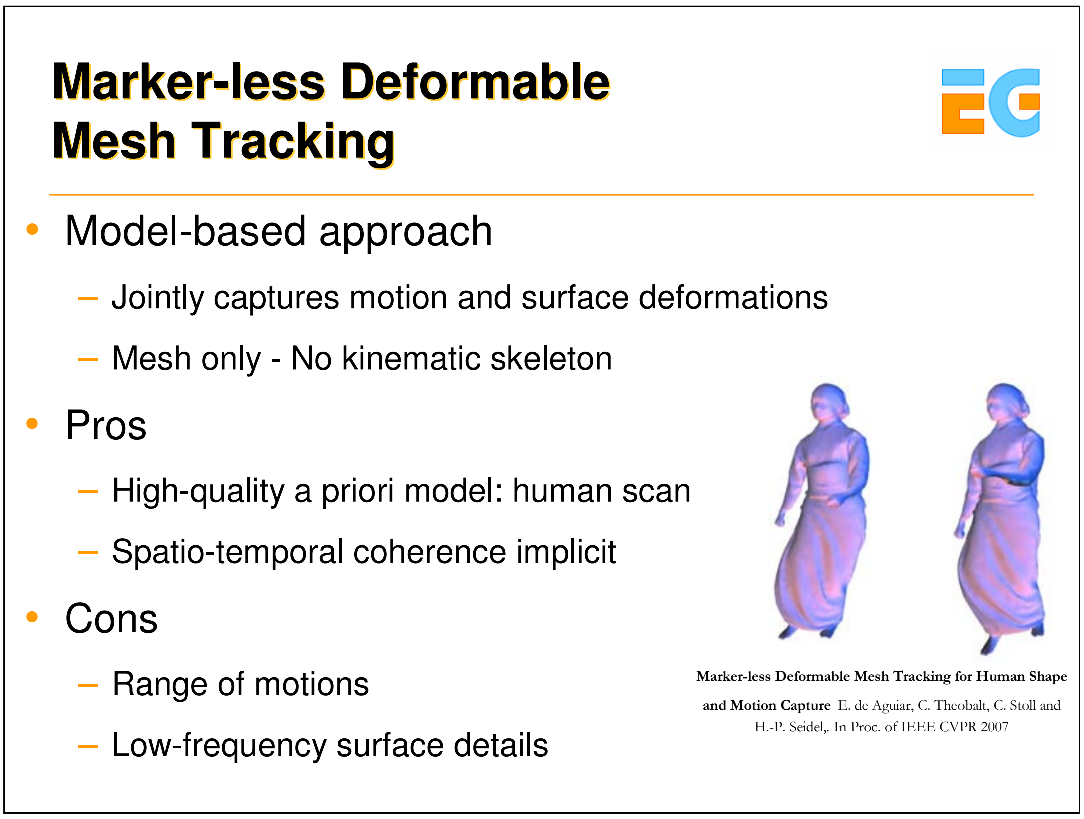
Surface Capture for Performance-based Animation

J. Starck and A. Hilton. To appear *IEEE CG&A*, 2007.

The Surface Motion Capture system [11] is a data-driven approach aiming at jointly capturing motion, time-varying deformation and appearance of a moving subject. Using this system no intrusion into the scene is necessary, which makes it appropriate for 3D Video applications. After reconstructing a triangle mesh from the visual hull (VH) and stereo information for each time step separately, additional time-consuming procedures (spherical parameterization and remapping) are used to construct a spatio-temporally coherent model. Although the system is able to nicely capture the motion and temporal cloth deformations for a relatively coarse reconstructed model, its main limitation comes from the topology changes in the model over time. This is an important issue to be considered when developing more advance applications, like 3D Video relighting.

Reference:

[11] Surface Capture for Performance-Based Animation, J. Starck and A. Hilton. to appear *IEEE Computer Graphics and Applications (CG&A)*, 2007.



Marker-less Deformable Mesh Tracking

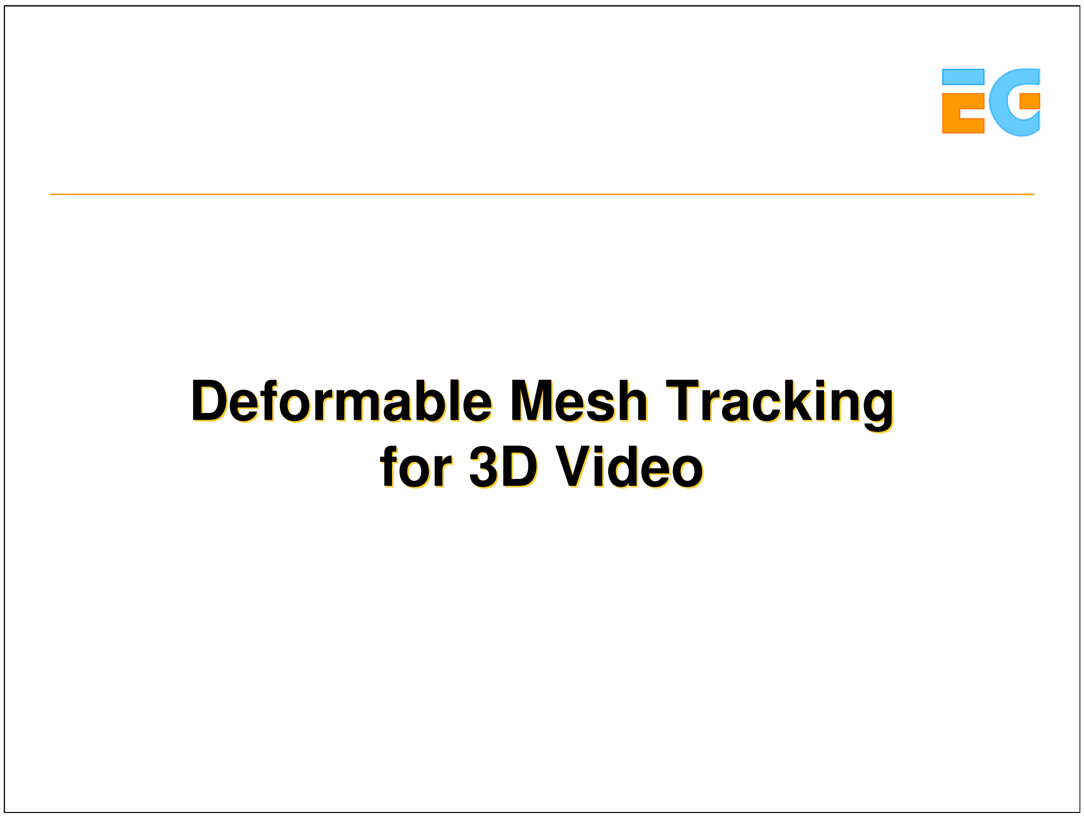
- Model-based approach
  - Jointly captures motion and surface deformations
  - Mesh only - No kinematic skeleton
- Pros
  - High-quality a priori model: human scan
  - Spatio-temporal coherence implicit
- Cons
  - Range of motions
  - Low-frequency surface details

Marker-less Deformable Mesh Tracking for Human Shape and Motion Capture E. de Aguiar, C. Theobalt, C. Stoll and H.-P. Seidel,. In Proc. of IEEE CVPR 2007

The marker-less deformable mesh tracking system [12] is a novel algorithm to jointly capture the motion and the dynamic shape of humans from multiple video streams without using optical markers. Instead of relying on kinematic skeletons, as traditional motion capture methods, it uses a deformable high-quality mesh of a human as scene representation. As opposed to many related methods, it can track people wearing wide apparel, it can straightforwardly be applied to any type of subject, e.g. animals, and it straightforwardly preserves the connectivity of the mesh over time. The main limitations are still the range of motions that can be correctly captured, since only eight cameras are used for recording the multi-view video sequences, and that the system cannot capture the true shape variation of low-frequency surface details, such as wrinkles in clothing. While the cloth and skin globally deform with the model, they seem to be 'baked in' to the surface. However, a combination of this method with a per-time-step stereo algorithm would overcome this limitation.

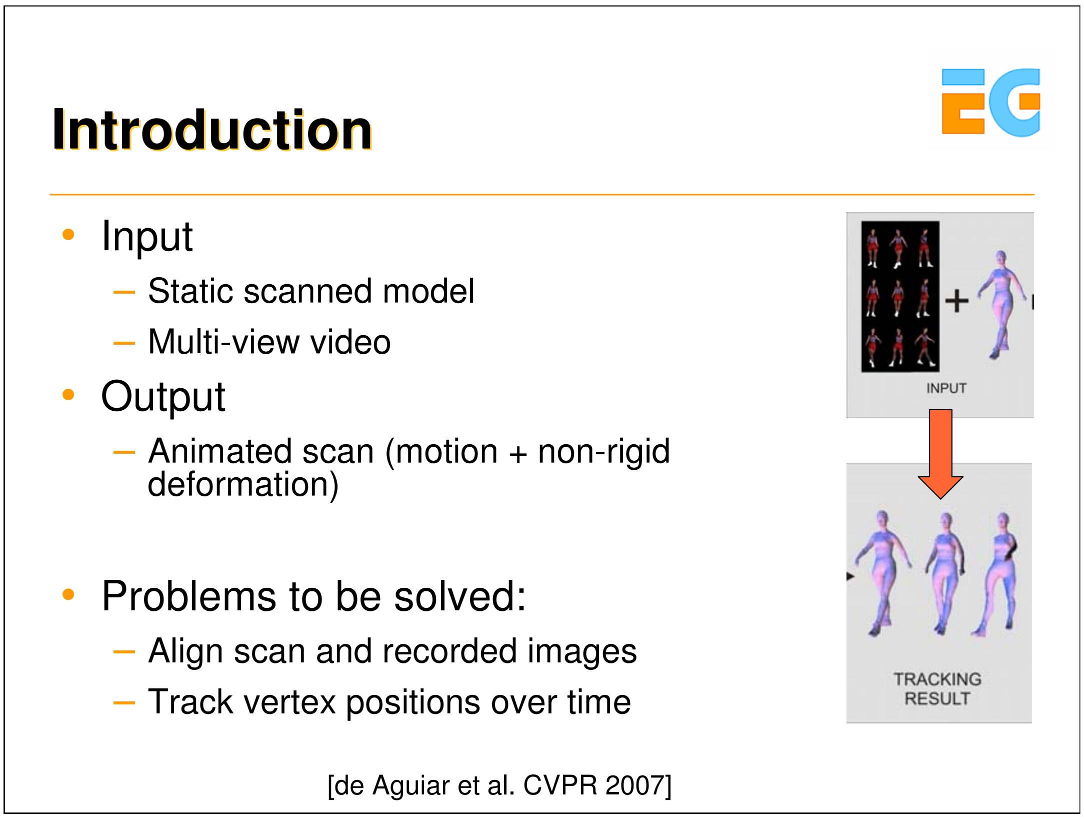
Reference:

[12] E. de Aguiar, C. Theobalt, C. Stoll and H.-P. Seidel, *Marker-less Deformable Mesh Tracking for Human Shape and Motion Capture.* In Proc. of IEEE CVPR 2007, Minneapolis, USA.



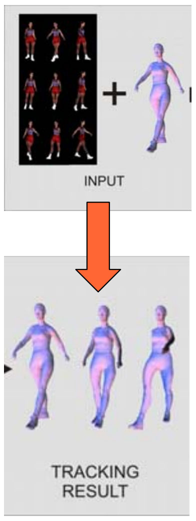
Deformable Mesh Tracking for 3D Video

Since it is more suitable for model-based 3D Video applications, the second algorithm will be described in more details.



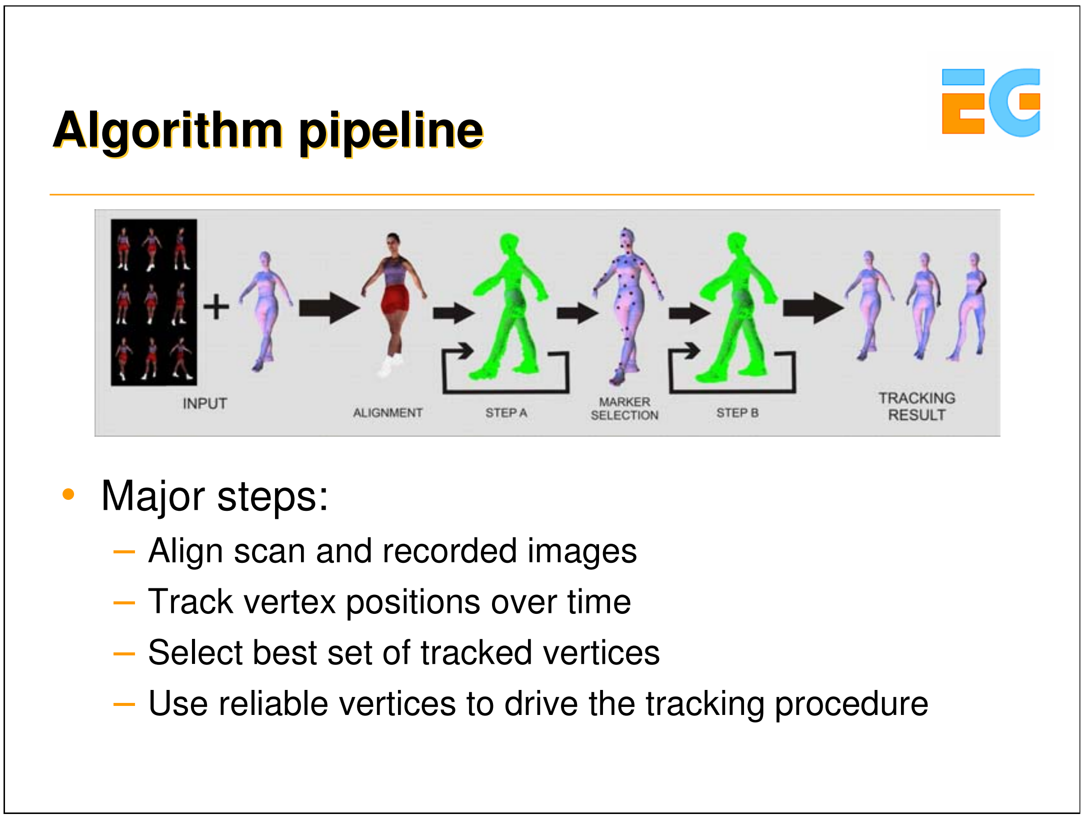
Introduction

- Input
  - Static scanned model
  - Multi-view video
- Output
  - Animated scan (motion + non-rigid deformation)
- Problems to be solved:
  - Align scan and recorded images
  - Track vertex positions over time

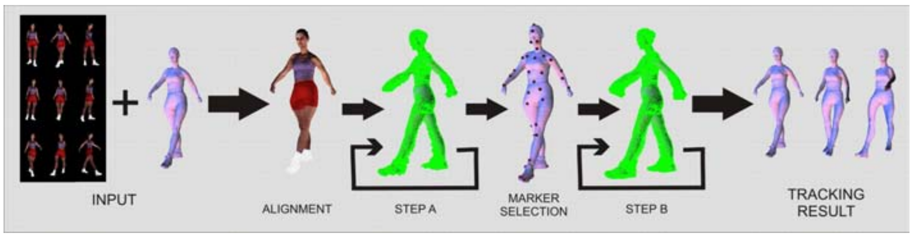


[de Aguiar et al. CVPR 2007]

The input to the system comprises of a static laser-scanned triangle mesh of the moving subject and a multi-view video sequence that shows her moving arbitrarily. As a result it outputs an animated mesh sequence where the model correctly follow the motion of the actor in all video frames. In order to accomplish that, first the laser scan should be aligned to the pose of the subject in the first time step of video and, thereafter, the vertices should be robustly tracked over time.



Algorithm pipeline



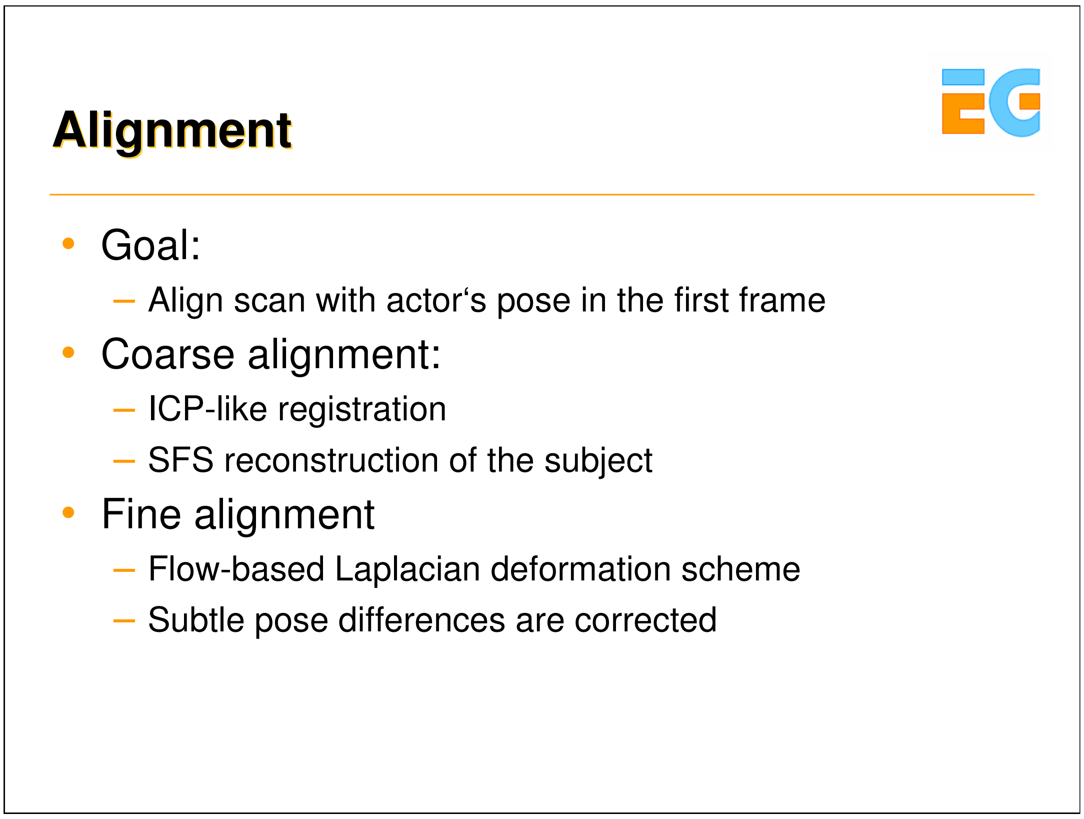
- Major steps:
  - Align scan and recorded images
  - Track vertex positions over time
  - Select best set of tracked vertices
  - Use reliable vertices to drive the tracking procedure

The method is composed of four steps. First, the scanned mesh is registered to the pose of the person in the first time step of video. Then, an iterative 3D flow-based deformation scheme is used to extract the motion information of each vertex over time from the images. Thereafter, N marker vertices that are tracked reliably over time are automatically identified. At the end, a more robust method that implicitly enforces structural integrity of the underlying mesh is used to track all vertices correctly.

Data acquisition

- Human scan:
  - Vitus Smart^{TM} full body laser scanner
  - Fast reconstruction (less than 10s)
  - Resolution of 1-2 mm
- Multi-View Video sequences
  - *Post-processing: Silhoeutte images*

For each test subject, the scanned model and several multi-view video sequences are acquired. The triangle mesh is captured with a Vitus Smart full body laser scanner. After scanning, the subject immediately moves to the nearby area where she is recorded with eight synchronized video cameras that run at 25 fps and provide 1004x1004 pixels frame resolution. The calibrated cameras are placed in an approximately circular arrangement around the center of the scene. After acquiring the multi-view sequences, silhouette images are calculated via color-based background subtraction.

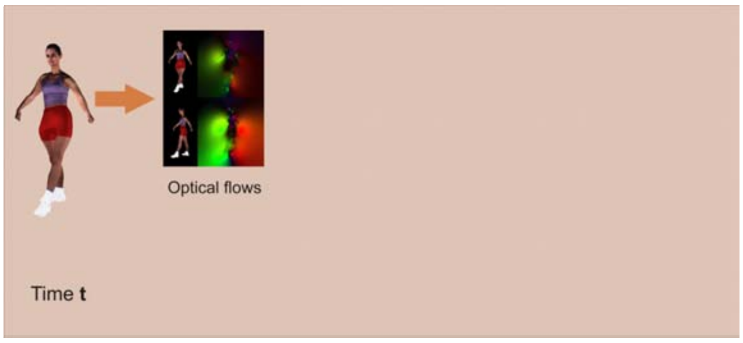


Alignment

- Goal:
  - Align scan with actor‘s pose in the first frame
- Coarse alignment:
  - ICP-like registration
  - SFS reconstruction of the subject
- Fine alignment
  - Flow-based Laplacian deformation scheme
  - Subtle pose differences are corrected

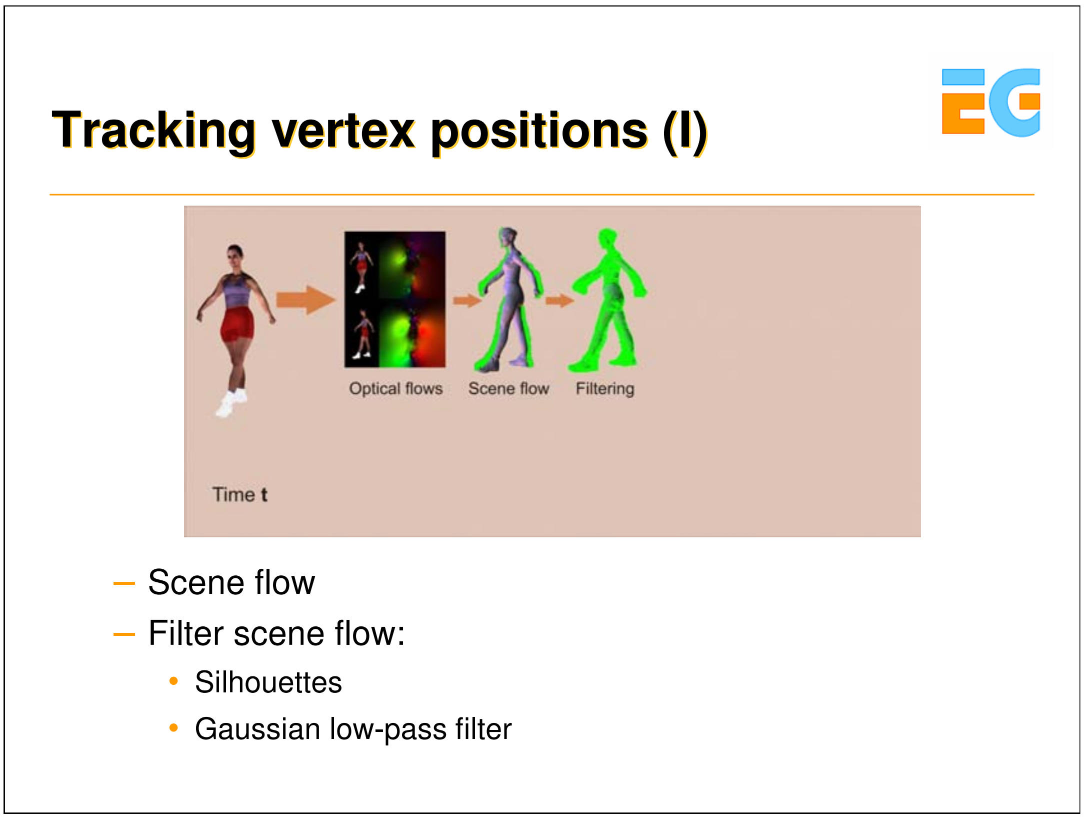
In an initial alignment we register the scanned mesh with the pose of the person in the first time step of video. To this end, she initially strikes the same pose that she was scanned in. By means of an ICP-like registration the mesh is first coarsely aligned to a shape-from-silhouette reconstruction of the person. Thereafter, the flow-based Laplacian deformation scheme is applied to correct for subtle pose differences.

Tracking vertex positions (I)

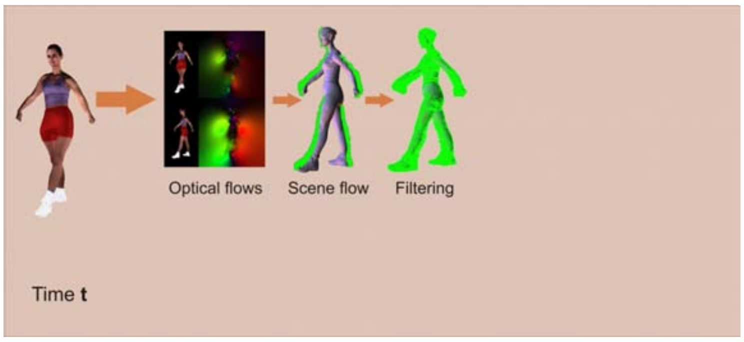


- Texture model
- Temporary images: project model back into camera views
- Optical flow between original and temporary images

After initial alignment, each individual vertex of the mesh is iteratively deformed based on the 3D optical flow fields that have been reconstructed from the multi-view images. Using subsequent time steps t and t + 1, the purely flow-driven mesh tracking approach consists of the following steps: first, the model is projectively textured using the images recorded with the K cameras at time step t. Then, K temporary images are generated by projecting the textured model back into all K camera views. After that, K 2D optical flow fields are calculated between temporary and original images.

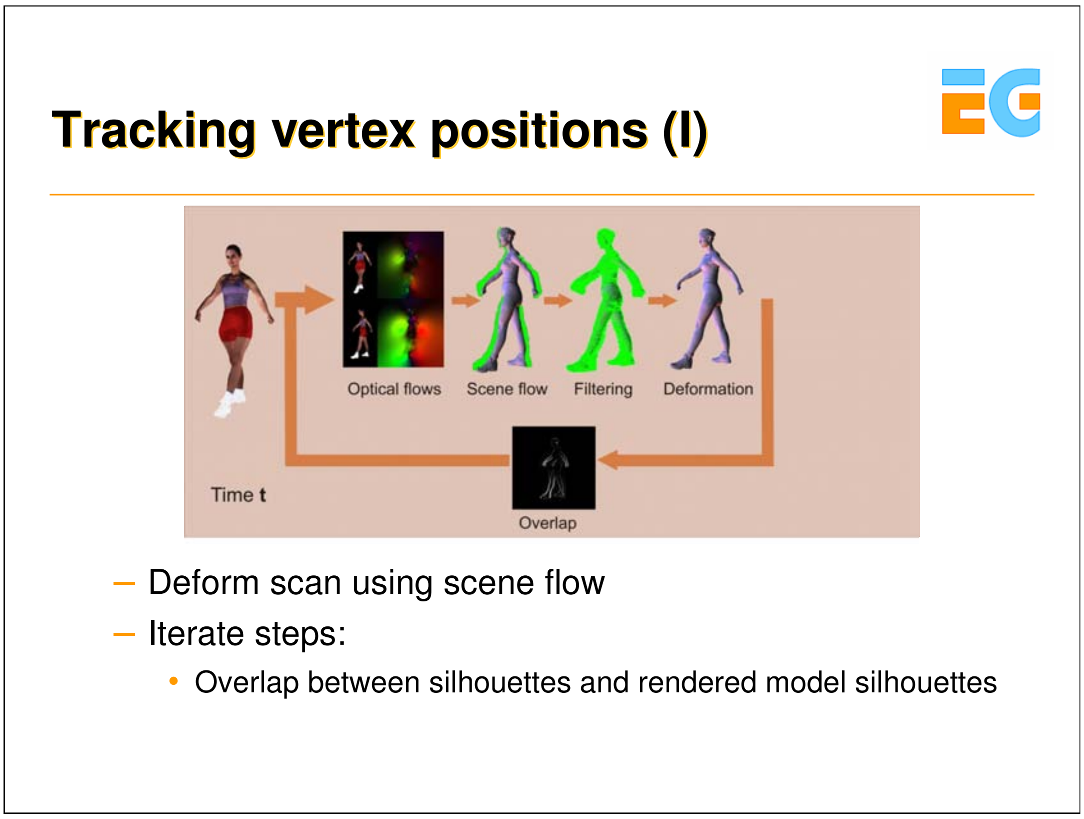


Tracking vertex positions (I)

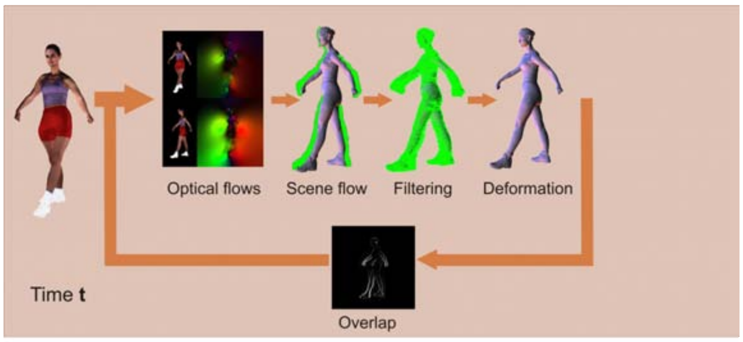


- Scene flow
- Filter scene flow:
  - Silhouettes
  - Gaussian low-pass filter

Given the model, calibrated cameras and the optical flow fields for all camera views, the scene flow can be computed by solving a linear system for each vertex that is visible from at least two camera views. The generated 3D flow field is parameterized over the mesh's surface and it describes the displacement by which the vertex should move from its current position. Thereafter, the 3D motion field is filtered in order to remove noise and outliers according to a silhouette-consistency criterion, and a Gaussian low-pass kernel is applied over the entire flow field.



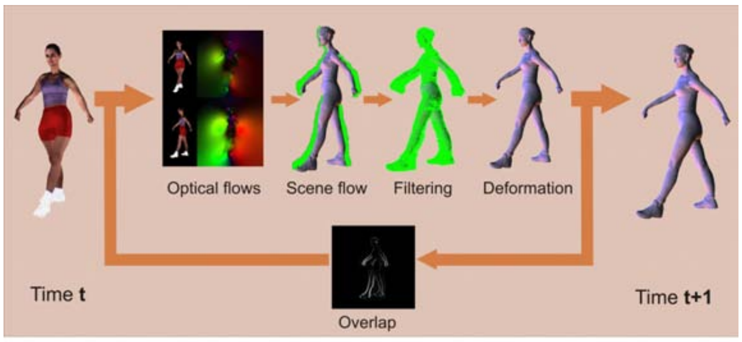
Tracking vertex positions (I)



- Deform scan using scene flow
- Iterate steps:
  - Overlap between silhouettes and rendered model silhouettes

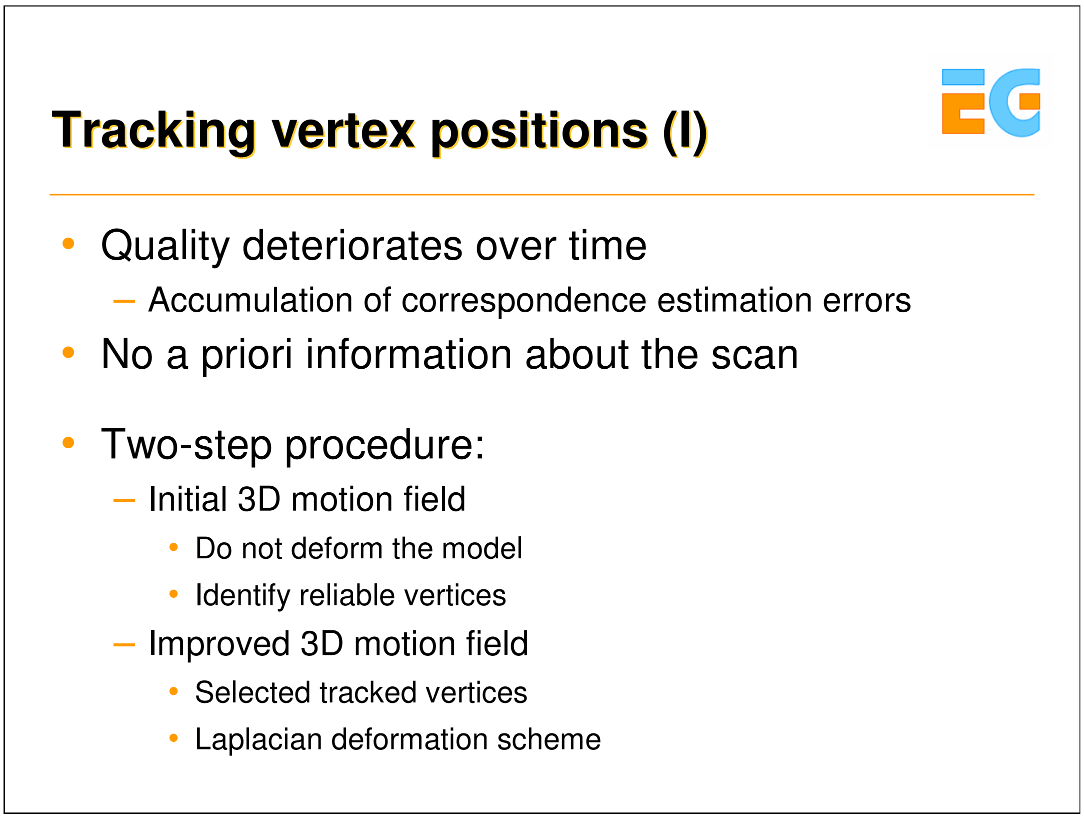
Using the filtered scene flow the model is updated by moving its vertices according to the computed displacements and the iterative process starts again until the overlap error between the rendered model silhouettes and the video-image silhouettes at time t + 1 in all camera views is below a threshold.

Tracking vertex positions (I)



- Scanned model is tracked over time!
- However:
  - Quality of animated scan deteriorates over time

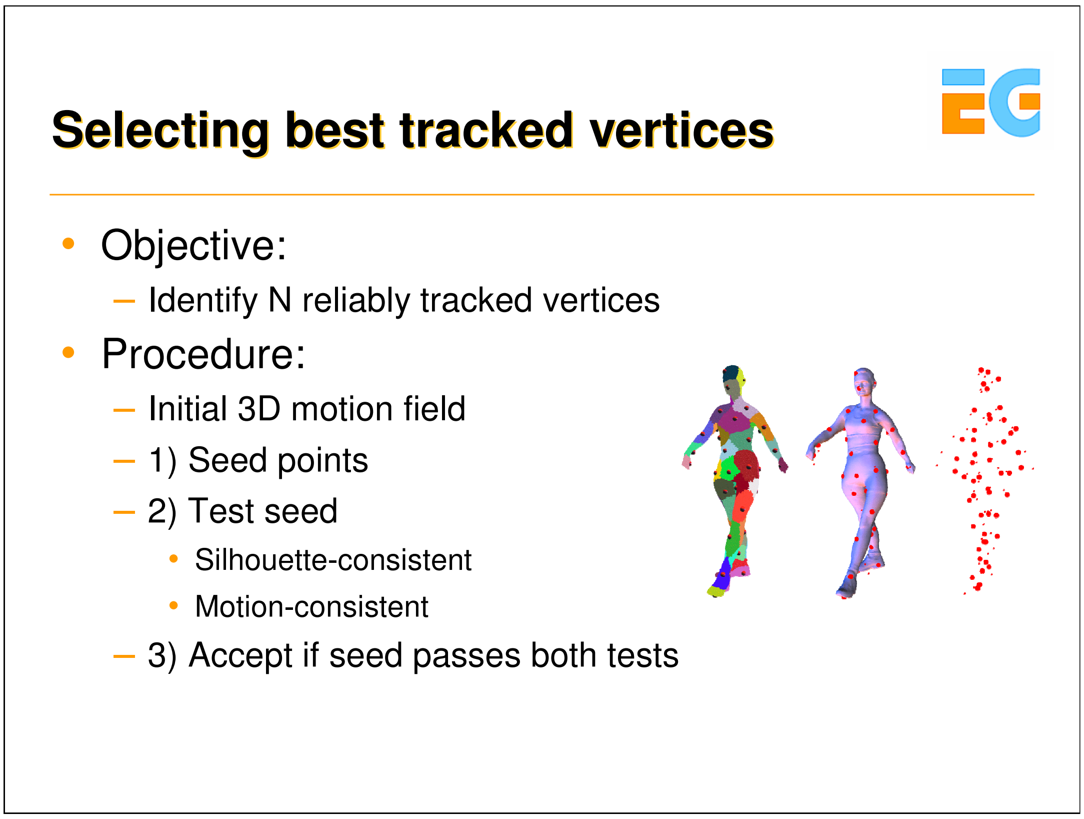
At the end, after applying the previously described steps to all pairs of subsequent time steps the model is tracked over the whole sequence. However, since this scheme calculates 3D displacements without taking into account a priori information about the shape of the scanned model, deformation errors accumulate over time.



Tracking vertex positions (I)

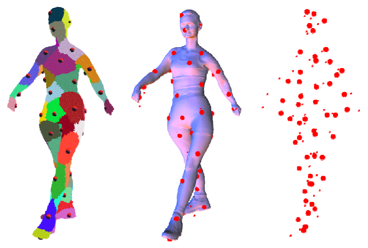
- Quality deteriorates over time
  - Accumulation of correspondence estimation errors
- No a priori information about the scan
- Two-step procedure:
  - Initial 3D motion field
    - Do not deform the model
    - Identify reliable vertices
  - Improved 3D motion field
    - Selected tracked vertices
    - Laplacian deformation scheme

The results of this simple tracking scheme quickly deteriorate due to accumulation of correspondence estimation errors. Since 3D displacements are calculated without taking into account a priori information about the shape of the model, the overall mesh quality is limited. Nonetheless, using this scheme it is possible to automatically identify N marker vertices that can be tracked reliably. Thereafter, an improved tracking scheme, more robust against flow errors, can be used which implicitly enforces structural integrity of the underlying model. This improved method uses the moving marked vertices as deformation constraints to drive a Laplacian deformation framework that makes all vertices correctly follow the motion of the actor.



Selecting best tracked vertices

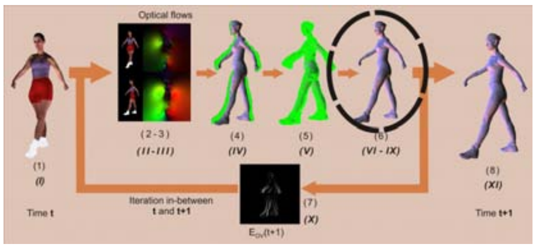
- Objective:
  - Identify N reliably tracked vertices
- Procedure:
  - Initial 3D motion field
  - 1) Seed points
  - 2) Test seed
    - Silhouette-consistent
    - Motion-consistent
  - 3) Accept if seed passes both tests



Based on the initial deformation results, this scheme selects N marked vertices of the model that were accurately tracked over time. To this end, first L candidate vertices are chosen that are regularly distributed over the model's surface. A candidate vertex is considered a marked vertex if it has a low error according to two spatio-temporal selection criteria based on silhouette-consistency and motion-consistency measures.

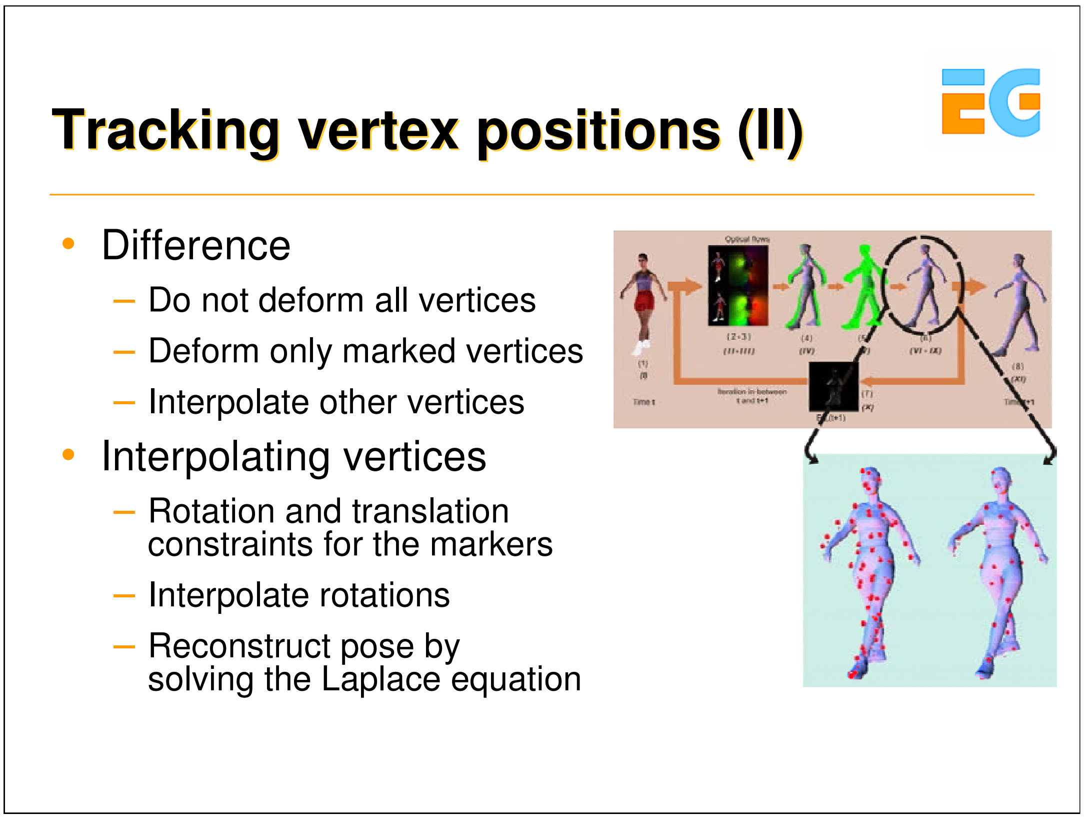
Tracking vertex positions (II)

- Laplacian scheme
  - Differential coordinates (d = Lv)
  - Structural details of scan
- Laplacian deformation scheme
  - Constraints from marked vertices
  - Interpolate remaining vertices
- Pipeline:
  - Similar to previous one
  - Deformation step



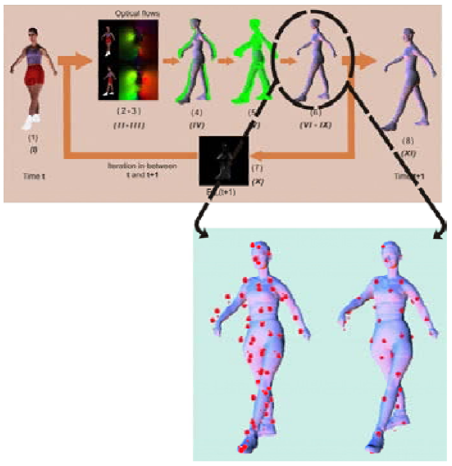
In the final step, the algorithm uses a Laplacian scheme to encode the knowledge about the structural details of the scanned model in terms of the mesh's differential coordinates d. They are computed by solving a linear system of the form d = Lv, where L is the discrete Laplace operator based on the cotangent-weights and v is the vector of vertex coordinates.

The main idea is to extract rotation and translation constraints from the motion of the N marked vertices to drive the Laplacian mesh deformation approach. By this means we can extract novel motion fields for each vertex that make the model correctly move and deform like the recorded subject. The individual steps of the Laplacian tracking scheme are very similar to the steps of the previous approach, differing however, in how the deformations are applied.

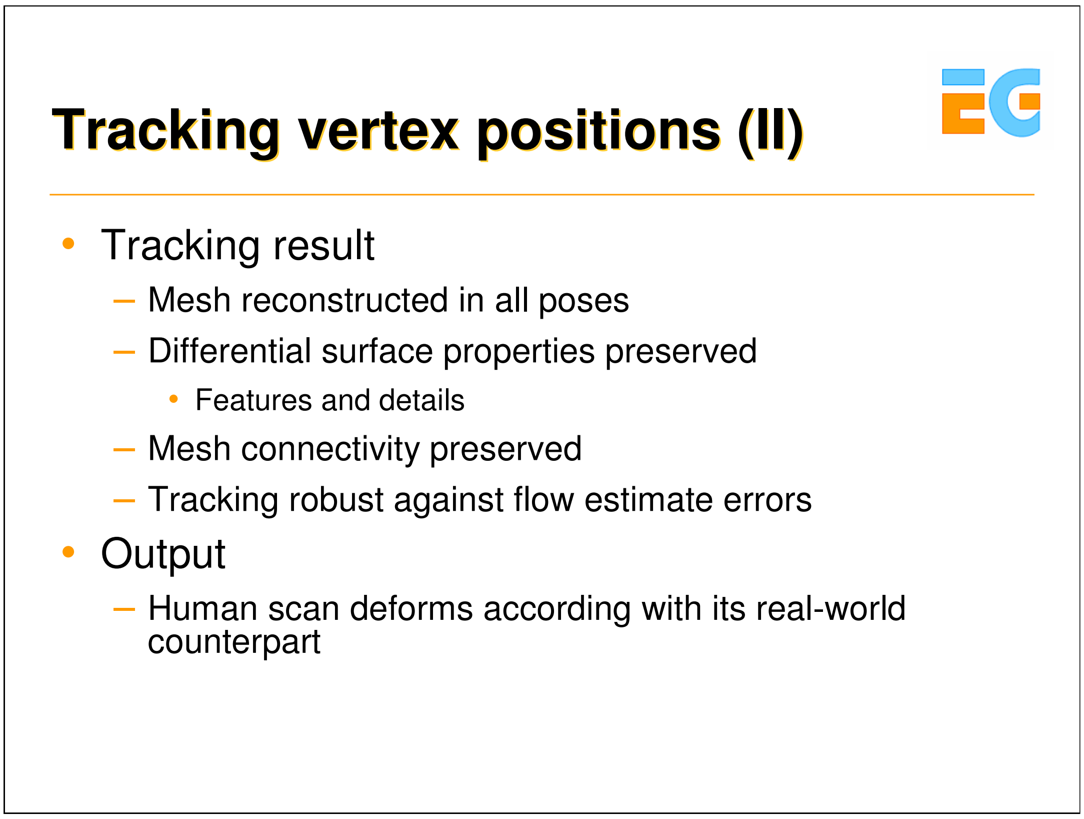


Tracking vertex positions (II)

- Difference
  - Do not deform all vertices
  - Deform only marked vertices
  - Interpolate other vertices
- Interpolating vertices
  - Rotation and translation constraints for the markers
  - Interpolate rotations
  - Reconstruct pose by solving the Laplace equation



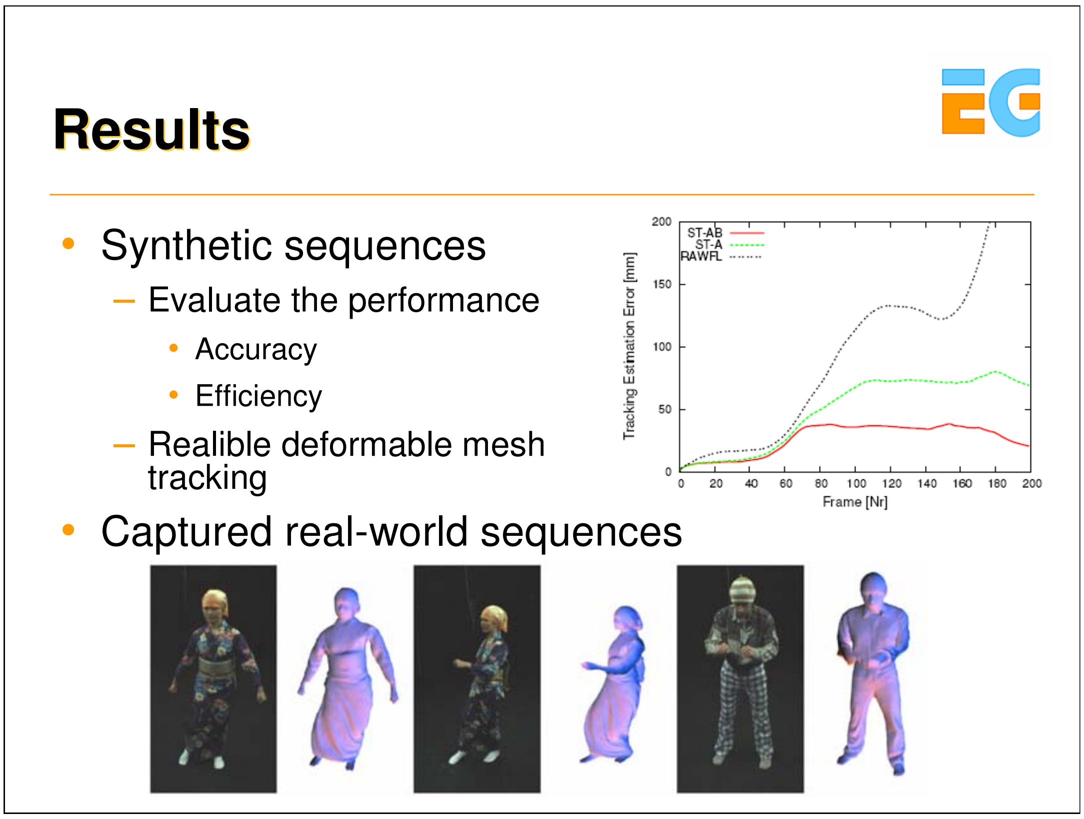
The main difference between the first and second tracking schemes is the way the calculated deformations are applied. In the first scheme, deformations are directly applied to all vertices. In the second scheme, the deformations for the marked vertices are used to guide our Laplacian deformation interpolation method: from the motion of each marked vertex a set of rotation and translation constraints is computed. Then, rotations for all markers are interpolated over the model by regarding the quaternion components as harmonic fields. At the end the model in its new target pose is reconstructed by solving the Laplace equation, subject to the constraints derived from the motion of the markers.



Tracking vertex positions (II)

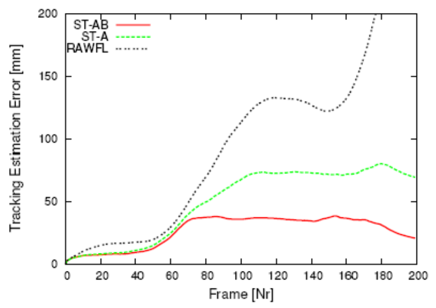
- Tracking result
  - Mesh reconstructed in all poses
  - Differential surface properties preserved
    - Features and details
  - Mesh connectivity preserved
  - Tracking robust against flow estimate errors
- Output
  - Human scan deforms according with its real-world counterpart

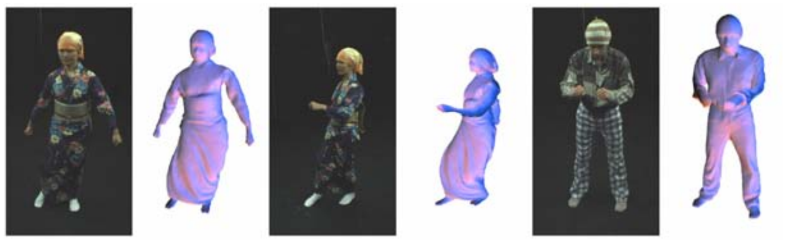
By applying these steps to all subsequent time steps the mesh is tracked over the whole video sequence. The Laplacian scheme reconstructs the mesh in its new pose in a way that preserves the differential surface properties of the original scan. Due to this implicit shape regularization, this improved tracking approach is robust against inaccurate flow estimates and deforms the mesh in accordance to its real-world counterpart in the video streams.



Results

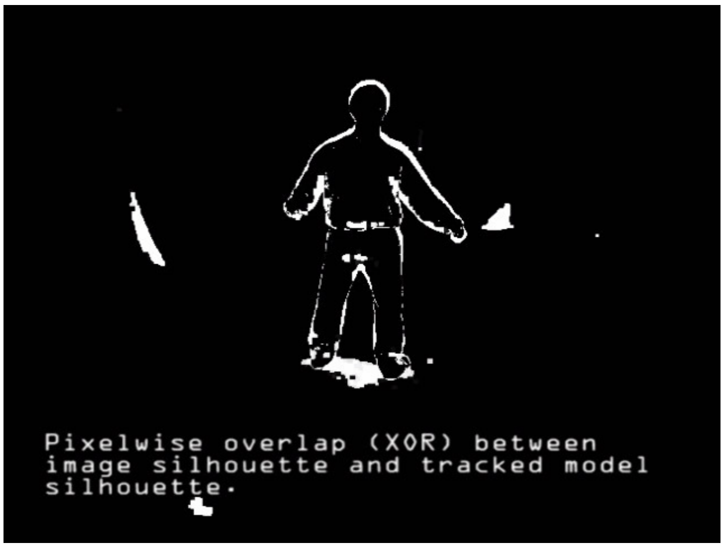
- Synthetic sequences
  - Evaluate the performance
    - Accuracy
    - Efficiency
  - Realible deformable mesh tracking
- Captured real-world sequences



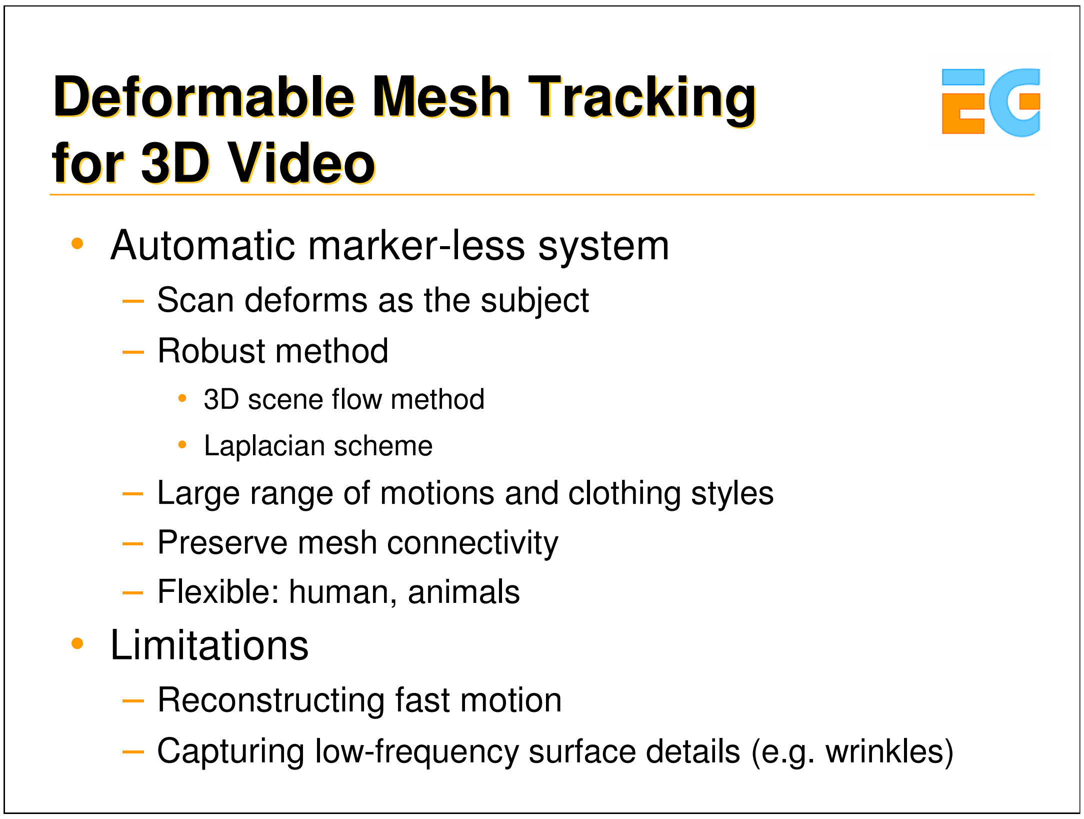


The algorithm was tested on several synthetic and captured real-world data sets. The synthetic sequences allow us to compare the results against the ground truth and evaluate the performance of the algorithm in terms of efficiency and accuracy. Different deformation alternatives were also compared, namely deformation along the unfiltered flow (*RAWFL*), deformation according to the first tracking method (*ST-A*), and deformation with our complete pipeline (*ST-AB*). Using *RAWFL*, the measurement error grows almost exponentially. Tracking with the first scheme leads to significantly better results, but the absolute inaccuracy is still comparably high. In contrast, the complete pipeline leads to satisfactory results. These experiments confirm that the complete tracking pipeline in combination with a high-quality dense flow method can reliably track human motion from raw unmodified video streams, as seen for the captured real-world sequence results.

Results


Pixelwise overlap (XOR) between
image silhouette and tracked model
silhouette.

For captured real-world results, the captured video sequences are between 300 and 600 frames long and show a variety of different clothing styles, including normal everyday apparel and a traditional Japanese kimono. Many different motions have been captured ranging from simple walking to gymnastic moves. The algorithm reliably recovers the pose and surface deformation for the subjects wearing comparably wide apparel, e.g. it can capture the motion and the cloth deformation for a woman wearing a kimono. The results show that this purely passive mesh-based tracking approach can automatically capture both pose and surface deformation of human actors. The combination of an a priori model, a fast Laplacian mesh deformation scheme, and a 3D flow-based correspondence estimation method enables us to capture complex shape deformations from only a few cameras.

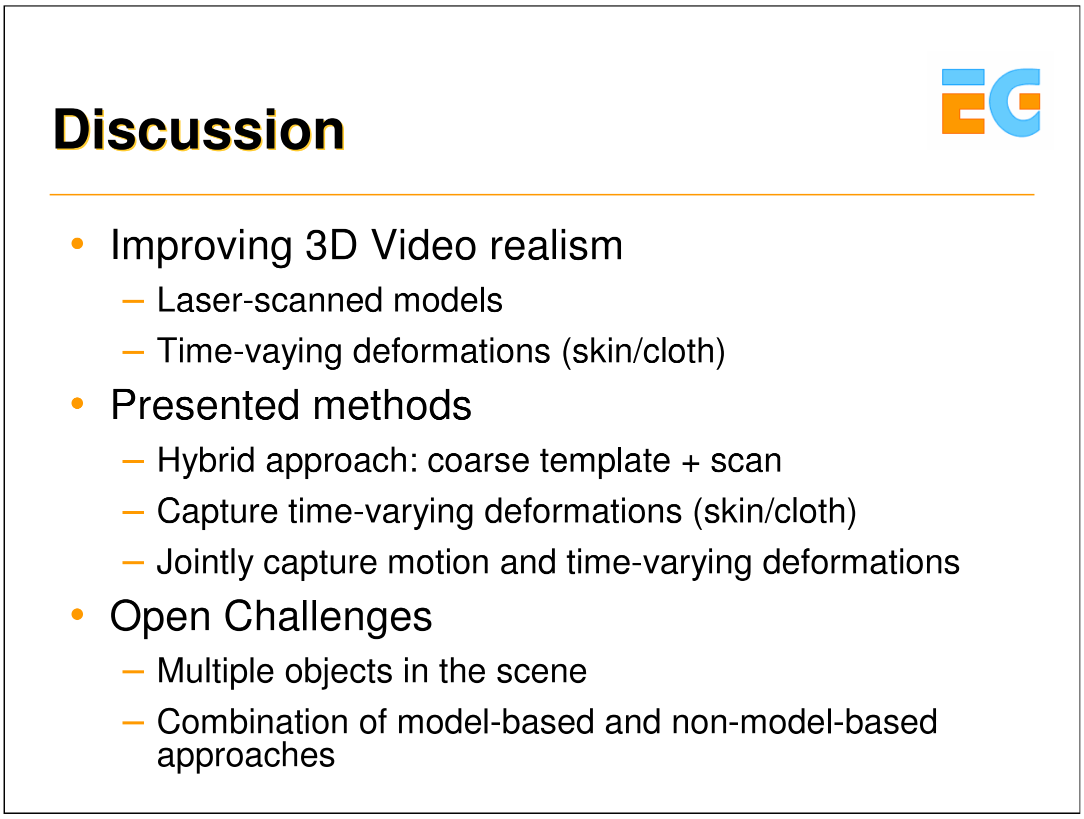


Deformable Mesh Tracking for 3D Video

- Automatic marker-less system
  - Scan deforms as the subject
  - Robust method
    - 3D scene flow method
    - Laplacian scheme
  - Large range of motions and clothing styles
  - Preserve mesh connectivity
  - Flexible: human, animals
- Limitations
  - Reconstructing fast motion
  - Capturing low-frequency surface details (e.g. wrinkles)

The deformable mesh tracking algorithm is a new solution for automatic marker-less tracking of deformable human models from a handful of video streams. The combination of a 3D scene flow-based correspondence estimation approach with a Laplacian mesh deformation scheme enables this method to make a laser scan of a subject move and deform in the same way as its real-world counterpart in video. The algorithm is easy to implement, preserve the mesh's connectivity and can handle a large range of human motions and clothing styles.

Nonetheless, this algorithm is subject to a few limitations. Problems may arise if the subject in the scene moves very quickly. In these situations, optical flow tracking may fail. However, this can be solved by using a high-speed camera that is available today for capturing fast scenes. Also, the algorithm cannot capture the true shape variation of low-frequency surface details, such as wrinkles in clothing. While they globally deform with the model, they seem to be 'baked in' to the surface. Although in typical 3D video applications, this inaccuracy does not play a major role, the authors are planning to extend the method in the future to capture these small details by means of a multi-view stereo algorithm. Despite these limitations this method is a flexible, easy to implement and reliable purely passive method to capture the time-varying shape of subjects from video.

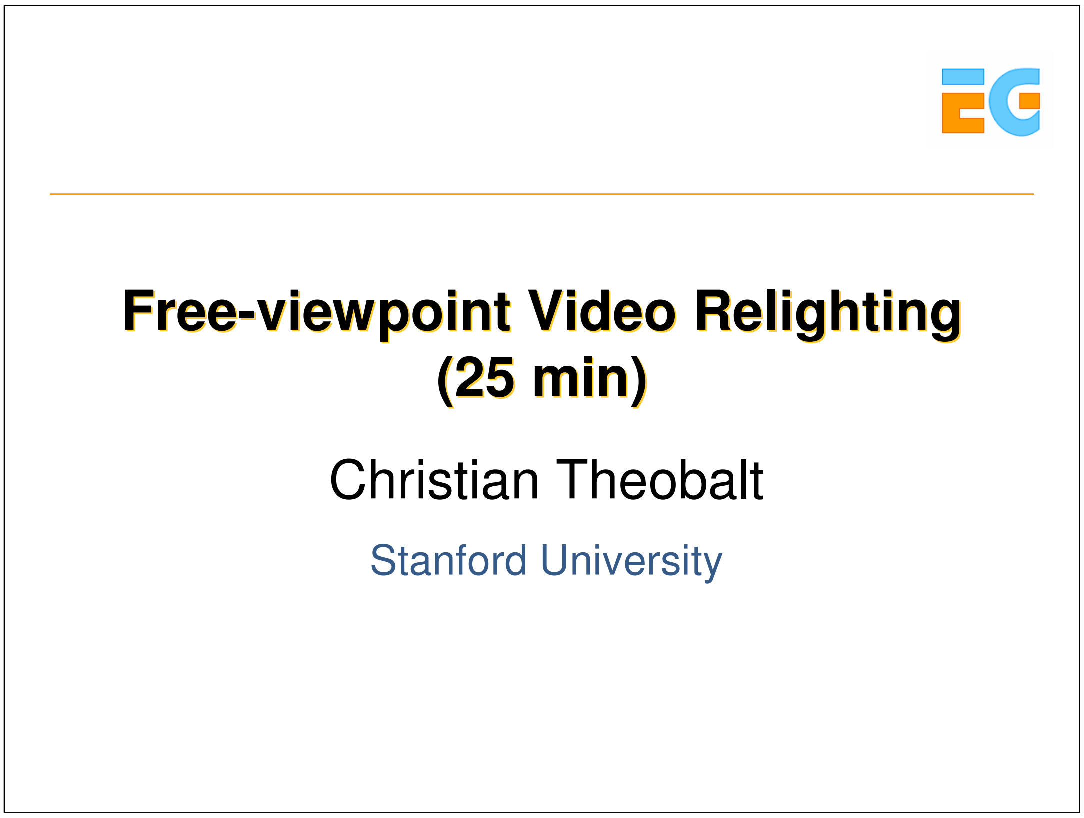


Discussion

- Improving 3D Video realism
  - Laser-scanned models
  - Time-vaying deformations (skin/cloth)
- Presented methods
  - Hybrid approach: coarse template + scan
  - Capture time-varying deformations (skin/cloth)
  - Jointly capture motion and time-varying deformations
- Open Challenges
  - Multiple objects in the scene
  - Combination of model-based and non-model-based approaches

To conclude, we presented some algorithmic alternatives that can improve the quality and realism of dynamic geometry in model-based 3D Video applications. As seen in this part of the course, by using a more detailed shape representation of the subject and by properly capturing time-varying deformations (cloth or skin), the realism of model-based 3D video can be increased. We first presented a hybrid method that can be used to incorporate a laser-scanned shape prior into the original 3D Video system. Thereafter, we showed model-based methods that are able to capture different types of non-trivially deforming complex surface geometry, such as the motion of cloth or non-rigid skinning. Finally, we presented a model-based marker-less mesh tracking approach that enables faithful reconstruction of the motion and time-varying geometry of even people wearing complex apparel.

However, many challenges remain open, two of them are named on the slide above.



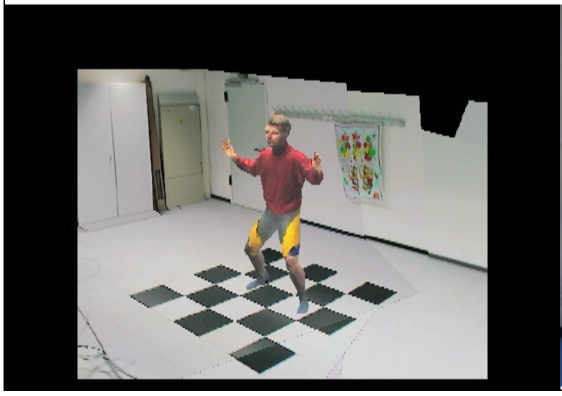
Free-viewpoint Video Relighting (25 min)

Christian Theobalt

Stanford University

Relightable Free-Viewpoint Video

- So far …
  - Arbitrary viewpoint
  - **Fixed lighting conditions**

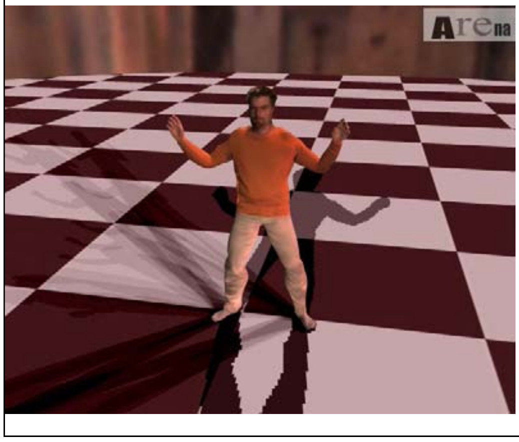


All approaches so far are able to reconstruct the dynamic and sometimes even view-dependent appearance of real-world scenes under the lighting conditions that prevailed at the time of recording.

Although this is one big step forward already, in many applications (e.g. 3D video compositing and postproduction, as well as many applications in games) one would like to be able to implant the captured 3D video footage into novel virtual scenes in which completely different lighting conditions exist.

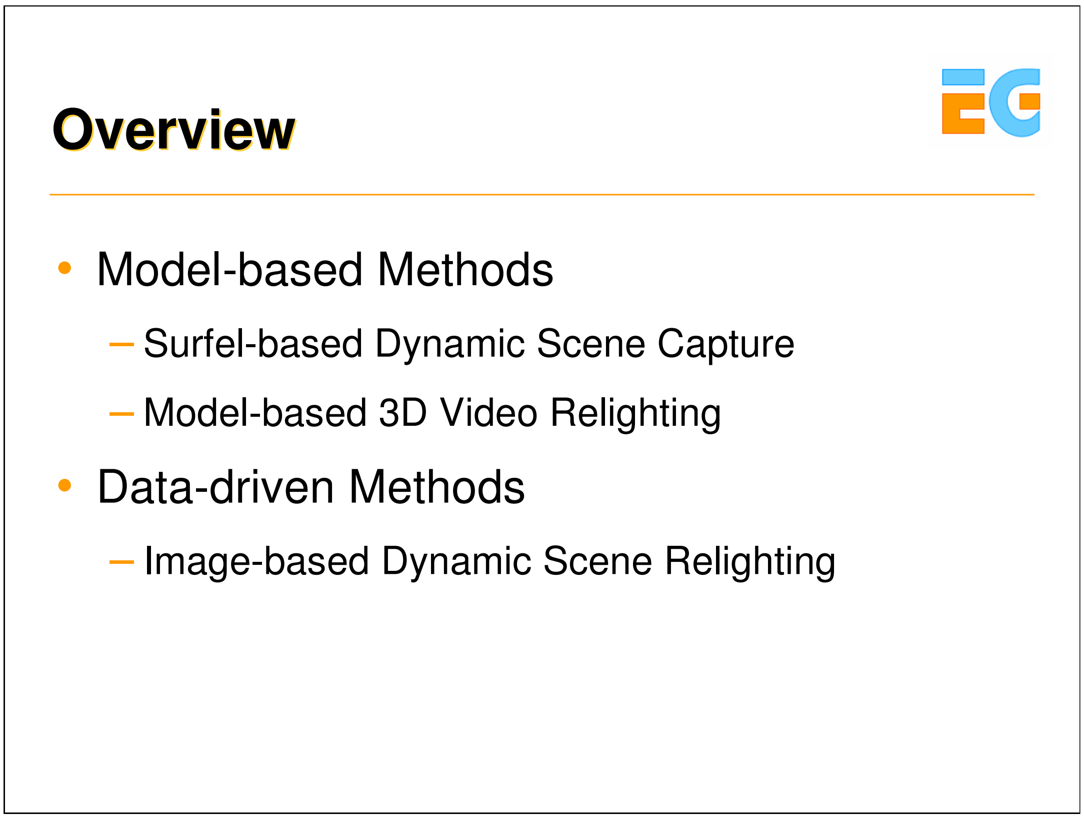
Relightable Free-Viewpoint Video

- New Lighting Conditions
  - **Dynamic Reflectance Capture**



To achieve this goal, one has to reconstruct more than only the dynamic surface appearance from the captured footage, namely information on the actual dynamic surface reflectance. This means one needs to know how the scene's appearance varies under changing incident lighting and outgoing viewing directions.

Only recently has acquisition and computation hardware become powerful enough to enable researchers to attack this even more challenging reconstruction problem. Just few approaches attacking this problem have therefore been published so far, and we will review the most important ones in this part of the course. The videos on this slide show exemplary results obtained with methods whose working principles we will detail in the following.

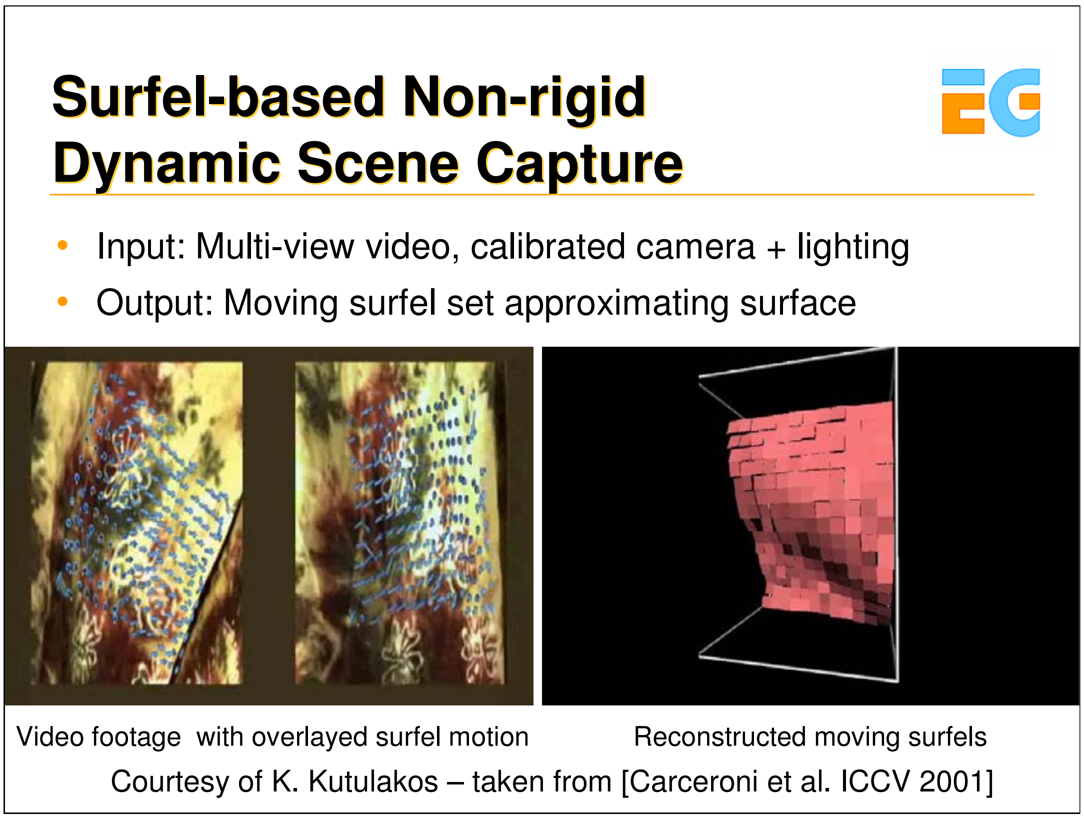


Overview

- Model-based Methods
  - Surfel-based Dynamic Scene Capture
  - Model-based 3D Video Relighting
- Data-driven Methods
  - Image-based Dynamic Scene Relighting

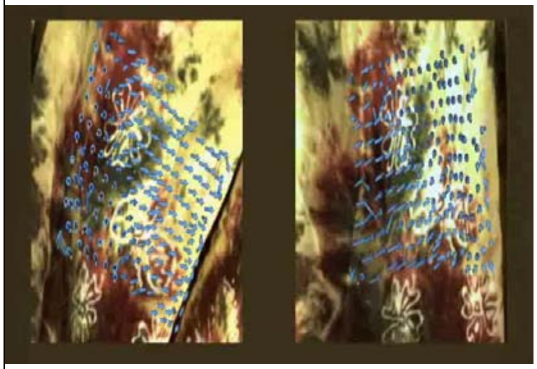
Two major categories of approaches have been proposed to attack the challenging problem of reconstructing dynamic relightable scene representations,. The first category of approaches is an extension of the original model-based 3D video pipeline. This will be the subject of the first part of this section.

An alternative way of attacking the problem is to take a data-driven or image-based approach. Instead of explicitly reconstructing shape and appearance models, these algorithms densely sample the space of camera viewpoints and lighting conditions and reconstruct novel views by appropriately combining the recorded input image streams. Data-driven methods will be the subject of the second part of this section.

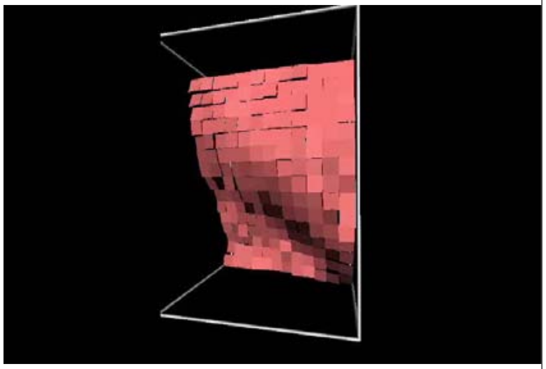


Surfel-based Non-rigid Dynamic Scene Capture

- Input: Multi-view video, calibrated camera + lighting
- Output: Moving surfel set approximating surface



Video footage with overlayed surfel motion



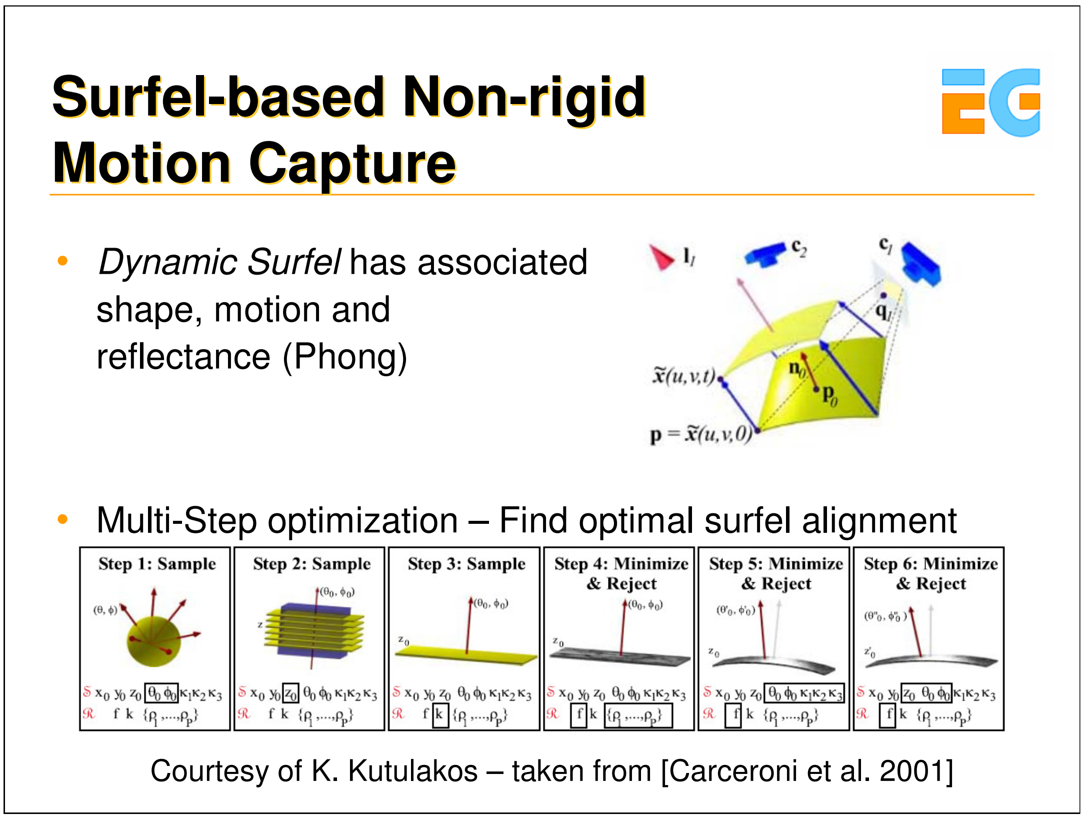
Reconstructed moving surfels

Courtesy of K. Kutulakos – taken from [Carceroni et al. ICCV 2001]

One of the first approaches that lays the path for model-based dynamic scene relighting is presented in [1]

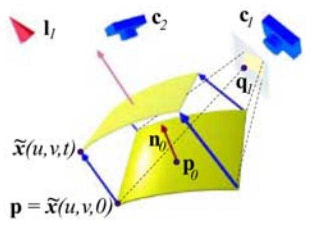
The goal of their algorithm is the reconstruction of non-rigidly deforming dynamic scene geometry from multi-view video footage that was recorded under calibrated lighting. Although the focus of this paper is not relighting itself, the proposed reconstruction algorithm also infers reflectance properties for discrete surface elements in order to recover the dynamic scene geometry as accurately as possible. Therefore, the ideas proposed in this paper can be regarded as a motivation for many of the concepts employed in the 3D video relighting approach discussed in the last part of this section.

This slide and the two following slides contain material that was kindly provided to the authors by Kyriakos Kutulakos from the University of Toronto. The videos shown on this slide can also be found here: http://homepages.dcc.ufmg.br/~carceron/surfels/

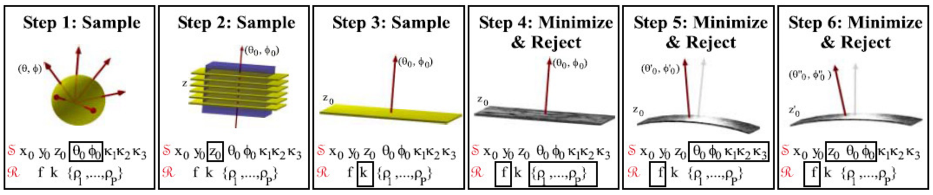


Surfel-based Non-rigid Motion Capture

- *Dynamic Surfel* has associated shape, motion and reflectance (Phong)



- Multi-Step optimization – Find optimal surfel alignment



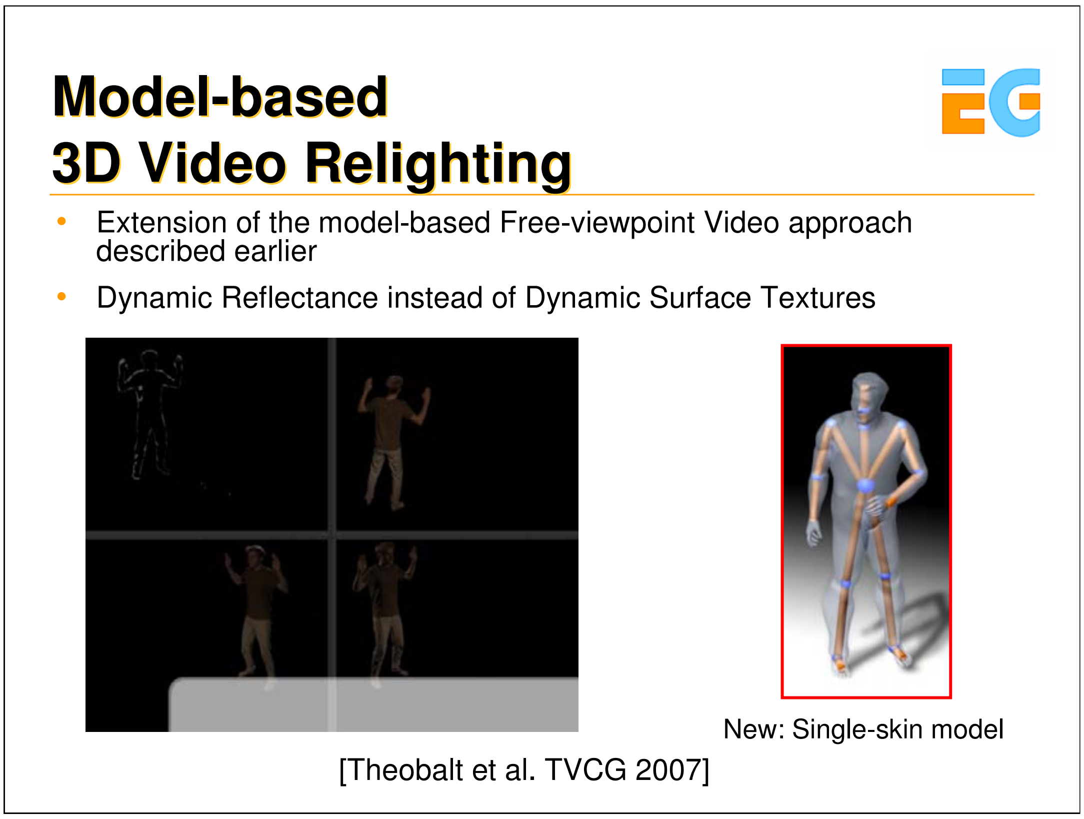
Step 1: Sample
Step 2: Sample
Step 3: Sample
Step 4: Minimize & Reject
Step 5: Minimize & Reject
Step 6: Minimize & Reject

Courtesy of K. Kutulakos – taken from [Carceroni et al. 2001]

The method tries to find the motion and orientation of so-called dynamic surfels relative to a discretized voxel grid in space. The mathematical description of each surfel features: A 3D shape component, more specifically position, normal orientation and curvature information, a reflectance component modeled by the Phong reflectance model [2], a description of the surfel's motion in terms of an instantaneous 3D velocity vector.

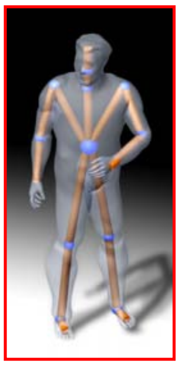
The approach now explores the space of possible surfel orientations and motions to reconstruct for each time step a complete hole-free approximation to the moving surface. This exploration, i.e. the search for optimal surfel parameters, is performed in the specific manner illustrated on the bottom of the slide.

The details of the method are described in [1]. Although scene relighting is not the primary goal in their work, many of the proposed ideas motivated the development of the model-based dynamic scene relighting approach shown in the following.



Model-based 3D Video Relighting

- Extension of the model-based Free-viewpoint Video approach described earlier
- Dynamic Reflectance instead of Dynamic Surface Textures



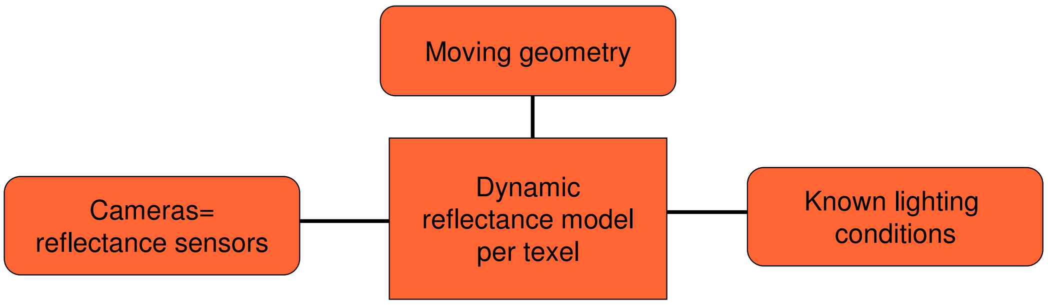
New: Single-skin model

[Theobalt et al. TVCG 2007]

Inspired by ideas from the paper by Carceroni et al. and recent progress in model-based reflectance reconstruction for static scenes we extended our original model-based 3D video pipeline in order to capture relightable scene representations [3].

Dynamic scene geometry is still reconstructed by capturing the motion of a kinematic template model. However, instead of using a segmented surface representation, we now use a single-skin model to represent the surface. This model is created from the shape-adapted segmented template model in a semi-automatic procedure.

Dynamic Reflectometry


Moving geometry
Cameras= reflectance sensors
Dynamic reflectance model per texel
Known lighting conditions

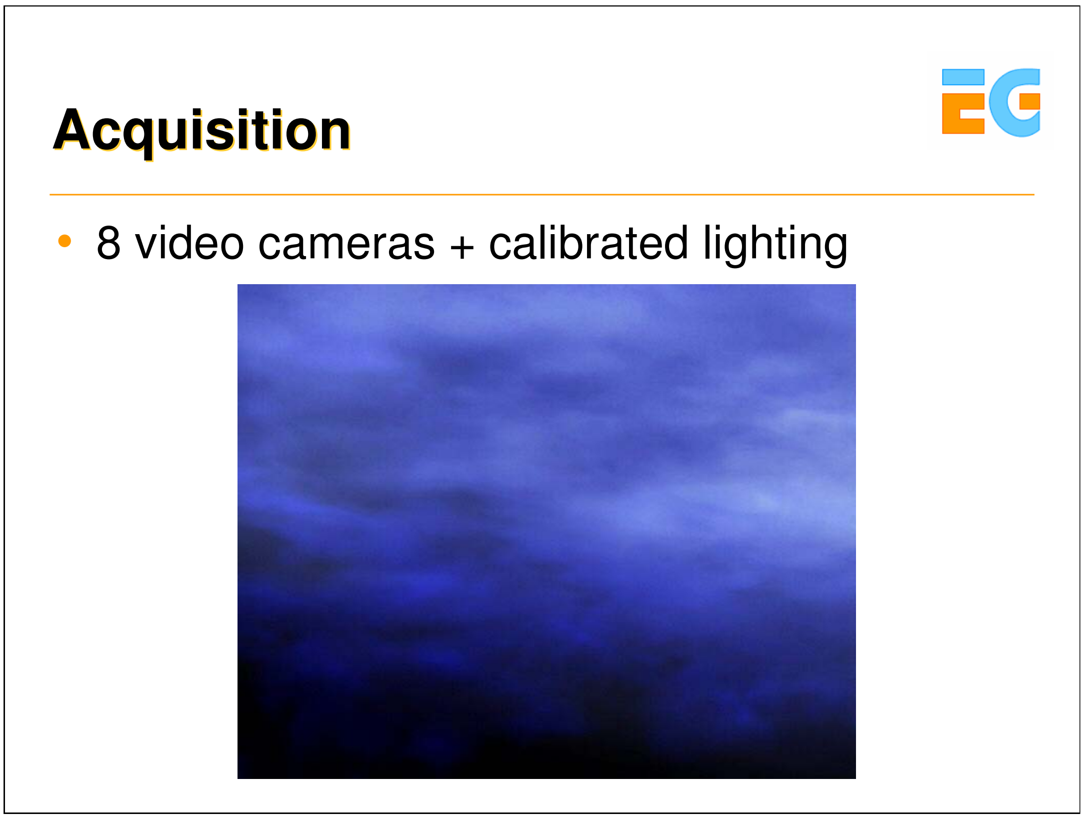
- Per-texel parametric BRDF model

  Phong, Lafortune $f_r(\hat{v},\hat{l}) = \frac{k_d}{\pi} + \frac{1}{\hat{n}\cdot\hat{l}}\sum_{i=1}^{N}(C_{x,i}l_x v_x + C_{y,i}l_y v_y + C_{z,i}l_z v_z)^{k_{e,i}}$

- Dynamic normal map

From the marker-based motion capture step we know how the human shape model moves with respect to the recording cameras. If we now record the input sequences under calibrated lighting, the cameras are not only texture sensors but turn into reflectance sensors. As the person moves with respect to the acquisition setup, each point on the surface is seen under different incoming lighting and outgoing viewing directions. Therefore, each pixel value in combination with the information on light and viewing directions represents a reflectance sample. Many of such samples are acquired in subsequent video frames as the person moves.

Using a process called dynamic reflectometry we fit to each texel on the model's surface a time-varying reflectance model which comprises of two main components.
The first component is a static parametric representation of the bidirectional reflectance distribution function (BRDF). The BRDF at a surface point is defined as the quotient of the outgoing radiance in a particular direction to the irradiance incoming from a specific direction. It is usually represented as a 6D function of incoming and outgoing directions as well as location on the surface. By computing an integral over the hemisphere of incoming light directions, the BRDF can be used to compute the outgoing radiance for any novel viewing direction. Due to their compactness and modeling power, we employ in our work parametric BRDF models that enable us to represent surface reflectance in terms of a few tunable parameters only [2,4].
The second component of our model is a time-varying normal direction for each texel. By this means, dynamic changes in surface geometry, such as folds in the apparel, can also be represented and realistically relit.



Acquisition

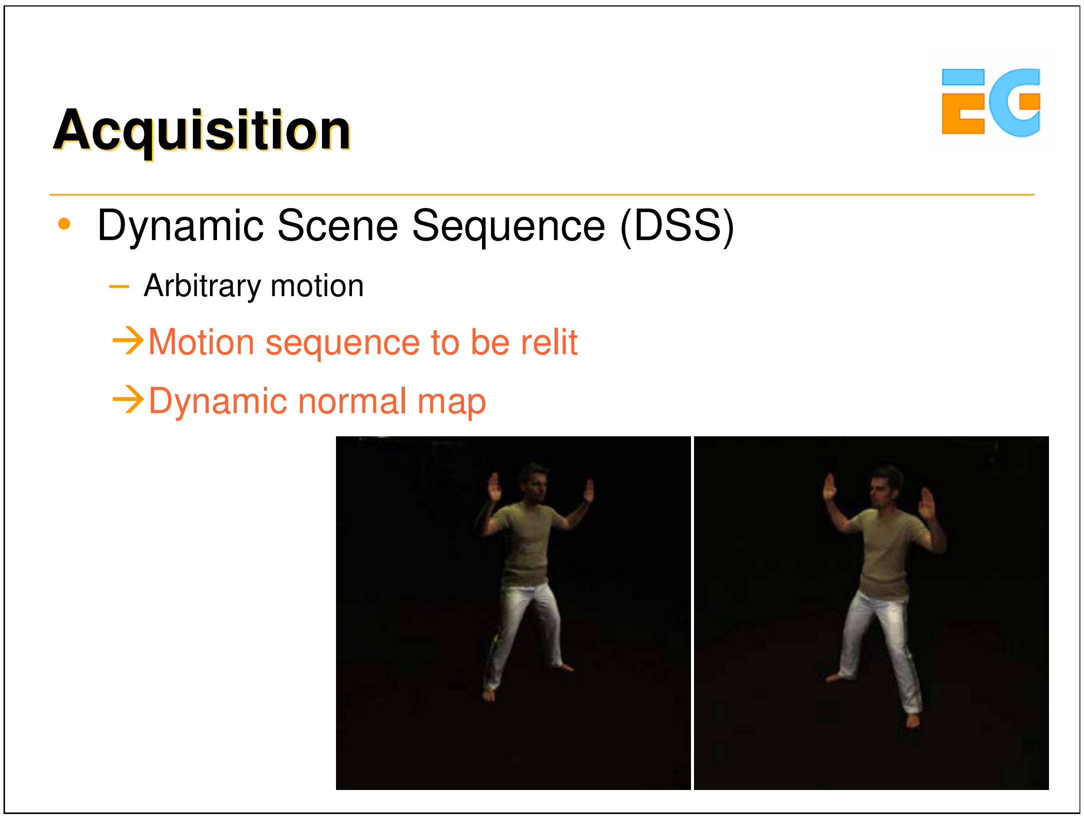
- 8 video cameras + calibrated lighting

Input footage was recorded in the 3D Video recording studio at MPI. Eight 1-Megapixel video cameras running at 25 fps are arranged in a approximately circular setup around the center of the scene. All scenes are recorded under a calibrated lighting setup. We employ two light sources that are placed at opposite corners of the setup to illuminate the scene. The positions of cameras and light sources are calibrated. In addition to geometric calibration, we also perform photometric and color calibration prior to recording.

Acquisition

- Reflectance Estimation Sequence (RES)
  - one for each type of apparel
    - → BRDF model

We record two types of multi-view video sequence to reconstruct relightable 3D videos. The first type of sequence, called reflectance estimation sequence (RES), is later used to estimate the per-texel BRDF parameters. We record one RES for each person and each type of apparel. In the RES, the person slowly rotates in front of the acquisition setup while trying to maintain a static upper body pose. By this means, we can capture many reflectance samples and prevent unwanted changes in local surface detail during acquisition. The pose of the body model in each frame is captured using our silhouette-based approach.



Acquisition

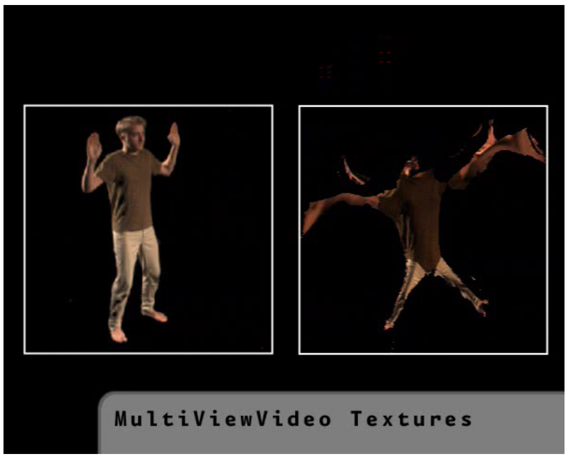
- Dynamic Scene Sequence (DSS)
  - Arbitrary motion
  - →Motion sequence to be relit
  - →Dynamic normal map

The second type of sequence is the so-called dynamic scene sequence (DSS). One DSS is captured for each performance of the actor that shall be reconstructed. Besides the actual motion, we also reconstruct the dynamic normal map from the DSS.

Multi-view Video Texture Generation

- Transform each input video frame into texture domain to obtain a

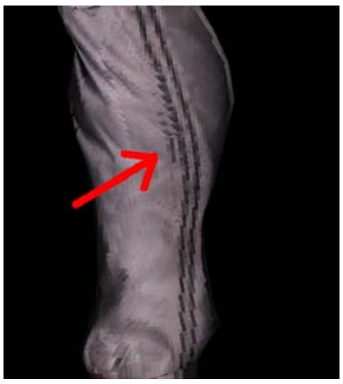
  Multi-view video (MVV) texture



To facilitate reconstruction and rendering, we are transforming each captured video frame into the texture domain and are thereby creating so-called multi-view video textures. One multi-view video texture is created for each frame and each camera.

Image-based Warp-Correction

- Problem: approximate geometry causes texture registration errors
- One solution: deform geometry
- Our solution: warp correction of input video frames



[Ahmed et al. ICIP 2007]

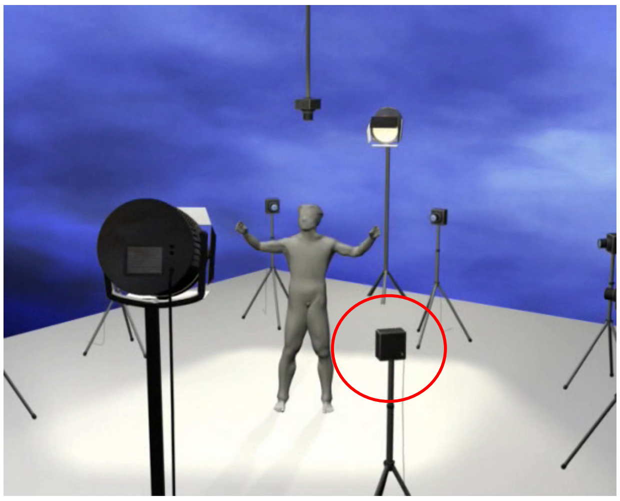
Before we reconstruct reflectance information from the input data, we have to solve a couple of registration problems. The first problem that we are facing is the fact that, due to approximate body geometry, there might be artifacts when projecting the textures back onto the model.

One solution would be to deform the geometry until the body model's shape best matches the input footage. Instead of doing the adjustment on the geometry level, we do the adjustment in image space. The idea is to warp the input images such that they optimally correspond to the multi-view image material captured. The warping of input images is performed as part of the multi-view texture generation process. It is described on the following slides and in [5].

Image-based Warp Correction

MVV texture generation for camera 0

**Trivial case:
Camera 0 sees surface point best**



Warp correction happens during multi-view video texture assembly. In the following, the warp correction steps are illustrated using a single surface point of the model as an example.

Let’s assume we would like to assemble a multi-view video texture for camera 0. In case it is camera 0 that sees a surface point best, which in this case means most head-on, we look up the appropriate color in the image captured by camera 0. This is the trivial case and no warp correction step is required.

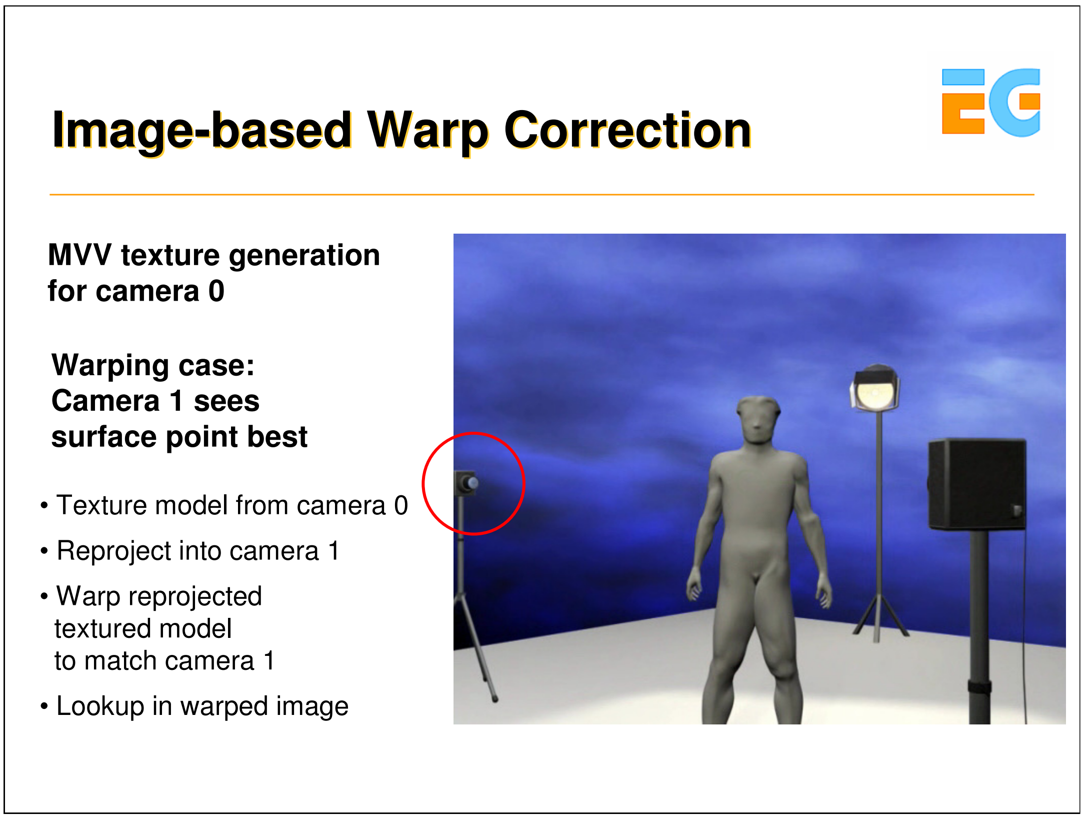


Image-based Warp Correction

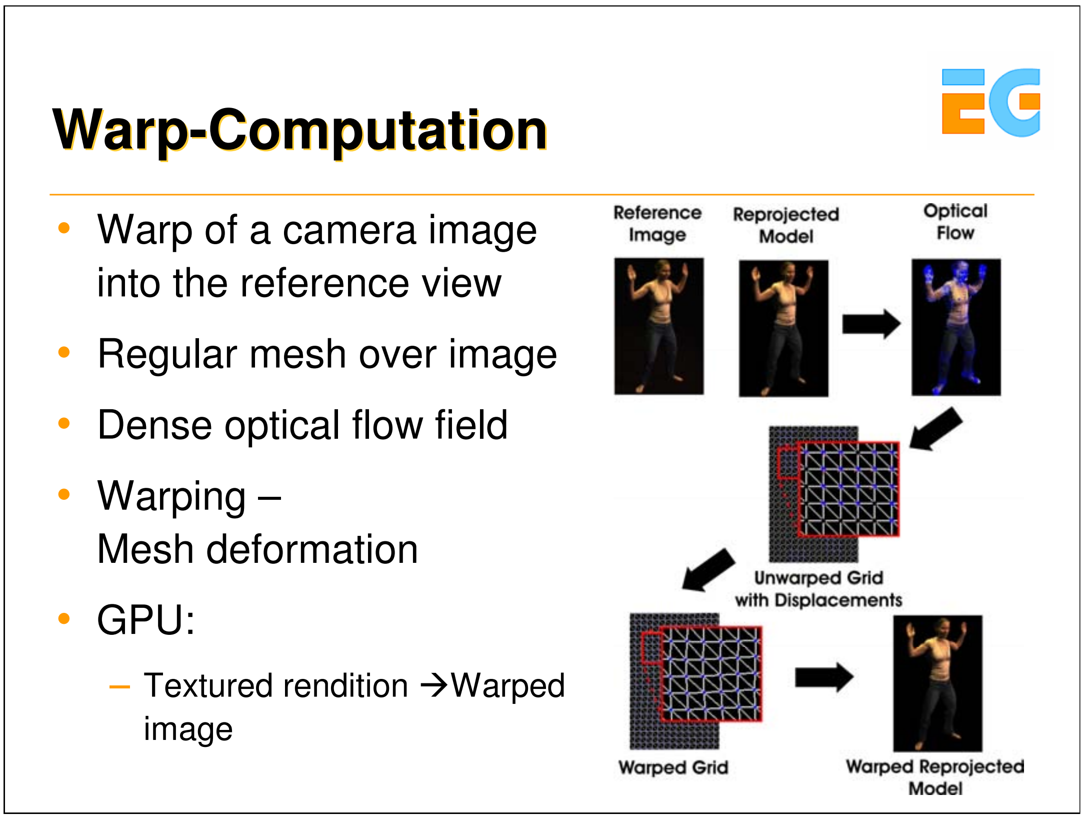
MVV texture generation for camera 0

Warping case: Camera 1 sees surface point best

- Texture model from camera 0
- Reproject into camera 1
- Warp reprojected textured model to match camera 1
- Lookup in warped image

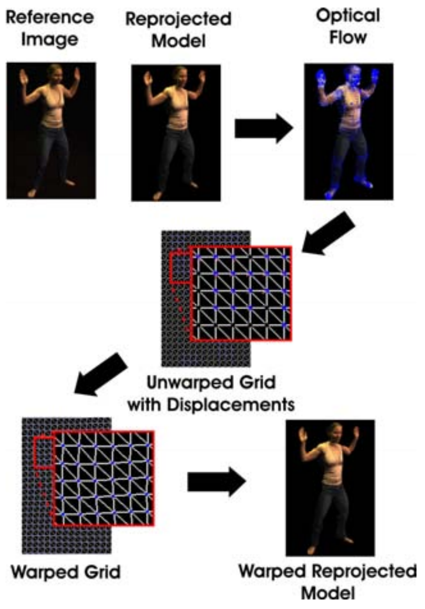
The warping case occurs if it is not camera 0 that sees a point best, but (without loss of generality) for instance camera 1 (red circle in image above).

In this case, the warping procedure is applied. To this end, the model is textured with the image from camera 0 and the so-textured model is projected back into the image of camera 1. The textured back-projected image is warped such that it optimally overlaps with the image actually captured from camera 1. The texel color is now looked up from the warped re-projected image. By this means, we make sure that the texture information always comes from camera 0 although we warp it into multiple camera views.



Warp-Computation

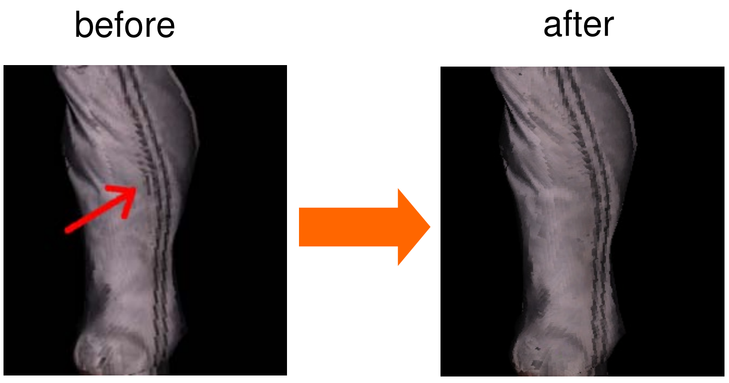
- Warp of a camera image into the reference view
- Regular mesh over image
- Dense optical flow field
- Warping – Mesh deformation
- GPU:
  - Textured rendition →Warped image



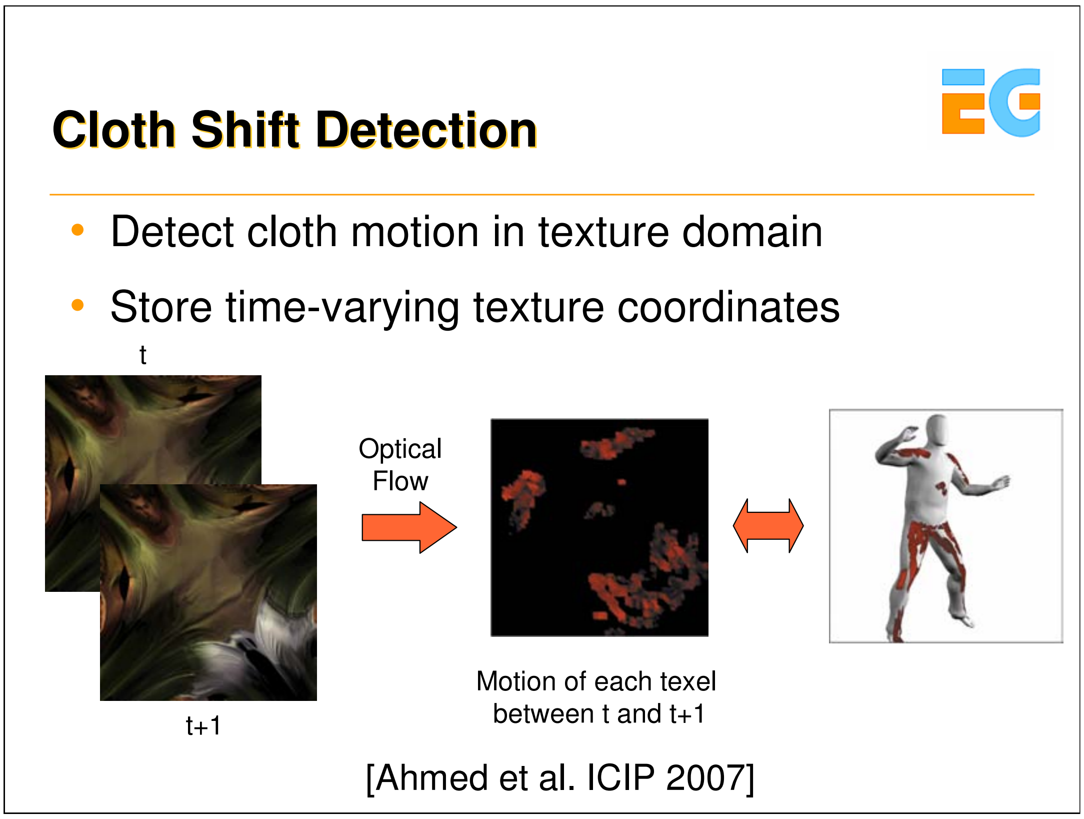
The image warping procedure itself, i.e. the procedure used to align the re-projected and captured images from camera 1, is based on optical flow computation and a subsequent image warping step. The working principle of the warping algorithm is illustrated in the flow diagram on this slide. To produce the final warped images, we make use of the texturing and filtering capabilities of the GPU and render image-aligned textured triangle meshes. Image-warping itself is implemented as a 2D smooth deformation of textured meshes.

Image-based Warp-Correction

- Improvements

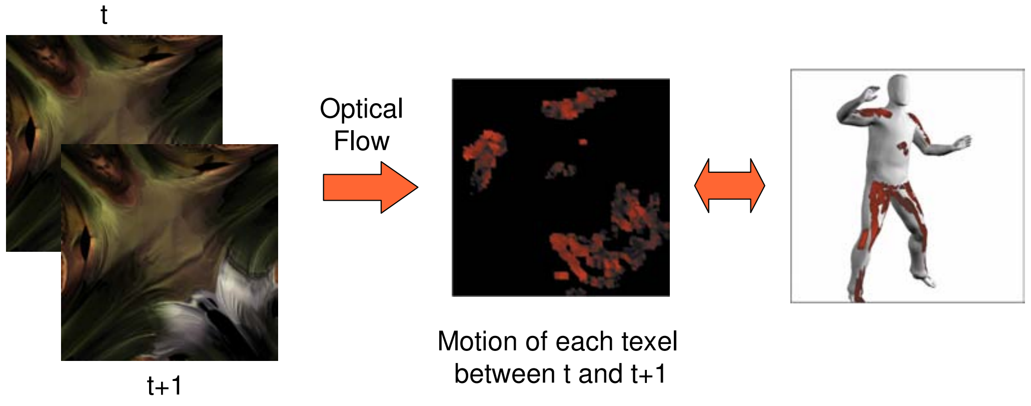


As one can see in the image above the “ghosting artifact” seen on the pants of the actor, which are due to shape misalignments, are corrected if our warping method is applied.



Cloth Shift Detection

- Detect cloth motion in texture domain
- Store time-varying texture coordinates



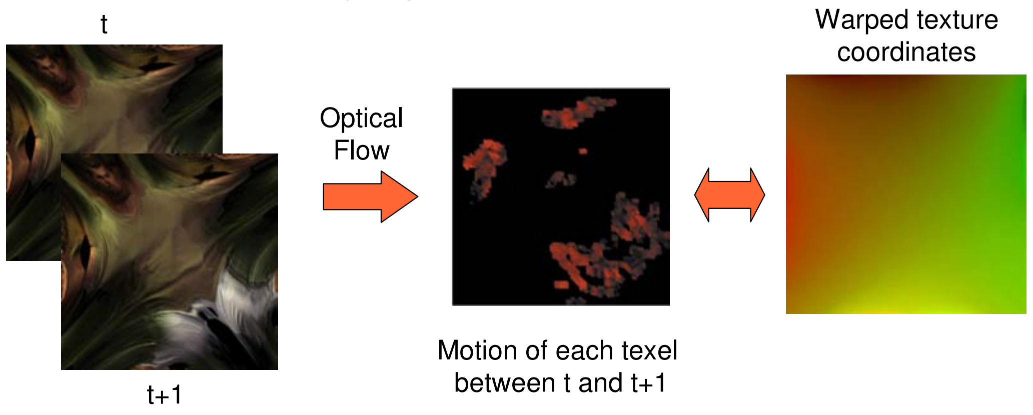
[Ahmed et al. ICIP 2007]

The second spatio-temporal registration problem we are confronted with is due to the fact that when the person is moving his apparel shifts across the body surface. This contradicts our assumption that we can assign a constant set of BRDF parameters to each location on the model's surface. In order to extract this cloth shifting information and in order to make it accessible to both our reflectance estimation framework and our renderer, we employ the cloth shift detection procedure illustrated in the slide above.

We identify the motion of apparel by means of a dense optical flow field between the complete surface texture images of the person in subsequent time steps. The complete surface texture is computed by weightedly blending all input images. An example of such a flow field is visualized in the image above. The 2D sections marked in red are areas in which motion has been detected. These areas correspond to specific surface areas on the model which are illustrated in the image to the right.

Cloth Shift Detection

- Detect cloth motion in texture domain
- Store time-varying texture coordinates

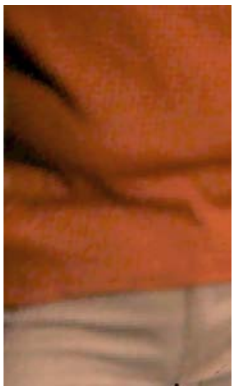

t
Optical Flow
Warped texture coordinates
Motion of each texel between t and t+1
t+1

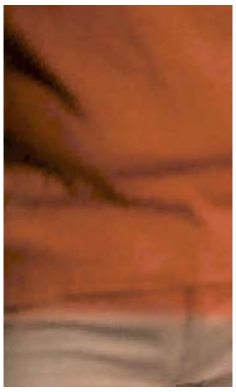
[Ahmed et al. ICIP 2007]

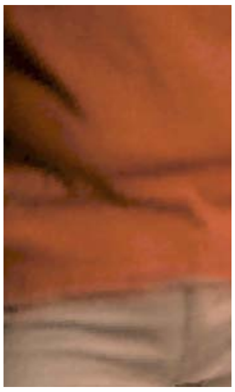
Flow fields are computed between all successive frame pairs. In the end, we make the information on cloth motion available to our renderer by warping the texture coordinates according to the recovered flow fields. The stream of warped texture coordinates is our final representation for cloth motion that enables us to do proper sample lookups during reconstruction and to render moving apparel despite a static assignment of material properties to the model surface. In both cases, i.e. reconstruction and rendering, texture information is looked up using the appropriately warped texture coordinates.

Cloth Shift Detection

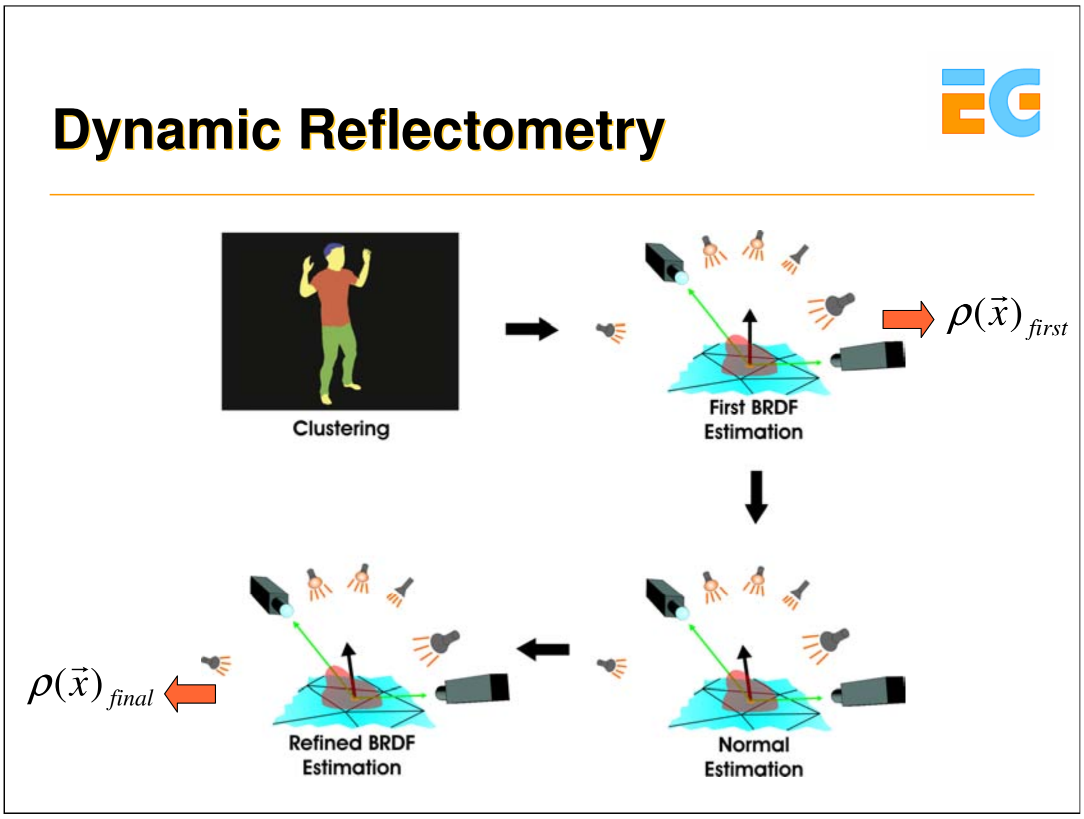
- Improvement


Input

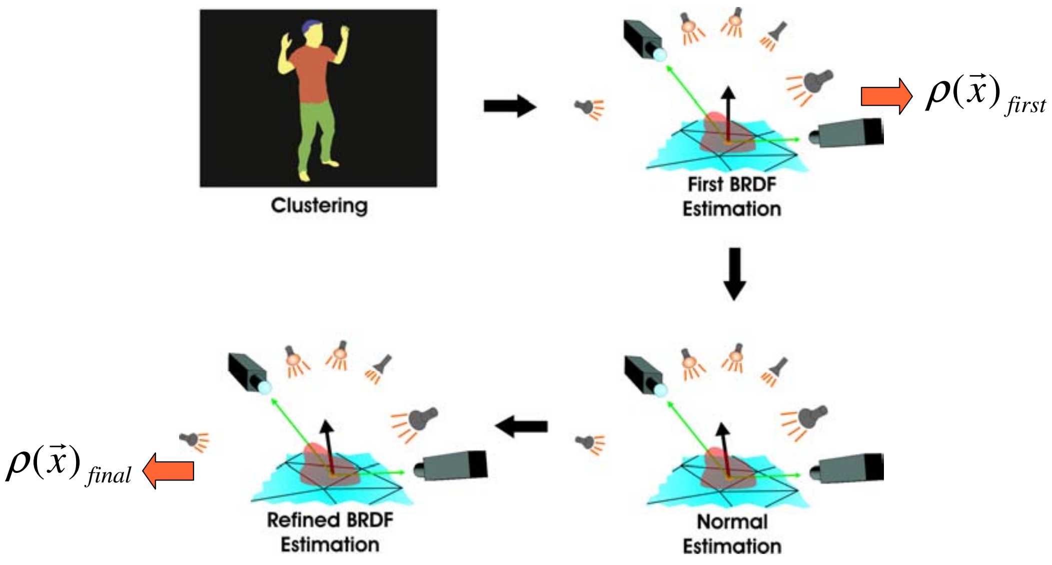

Rendering without shift detection


Rendering with shift detection

This set of images shows a close-up view of the waist area of a reconstructed relightable 3D video. The images illustrate the usefulness of our cloth motion detection method. The leftmost image is a ground truth image in which the boundary of the t-shirt shifted upwards in comparison to the first frame of the sequence. The image in the middle shows a reconstruction without cloth shift correction where the boundary of the shirt is improperly reproduced. The image to the right shows the correct assignment of the t-shirt color to the boundary location when cloth shift detection is applied during reconstruction and rendering.



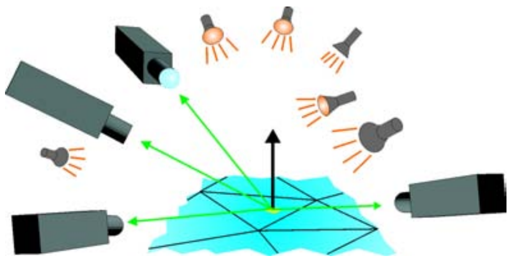
Dynamic Reflectometry



After all pre-processing steps are completed and the spatio-temporal reconstruction problems are solved, the actual reflectance estimation commences. In a process called dynamic reflectometry a separate set of BRDF parameters is estimated for each texel on the model's surface. The input to the dynamic reflectometry procedure is the reflectance estimation sequence.

The first step in the dynamic reflectometry process is the clustering of texels into groups of similar surface material according to the average diffuse color. After clustering, a first set of BRDF parameters is estimated via an energy minimization procedure. Given the first BRDF estimates, the default normal field of the template model is refined to better reproduce the true surface geometry in the RES. In a final estimation step, a new set of BRDF parameters is estimated using the now refined geometry description.

Dynamic Reflectometry - BRDF Estimation



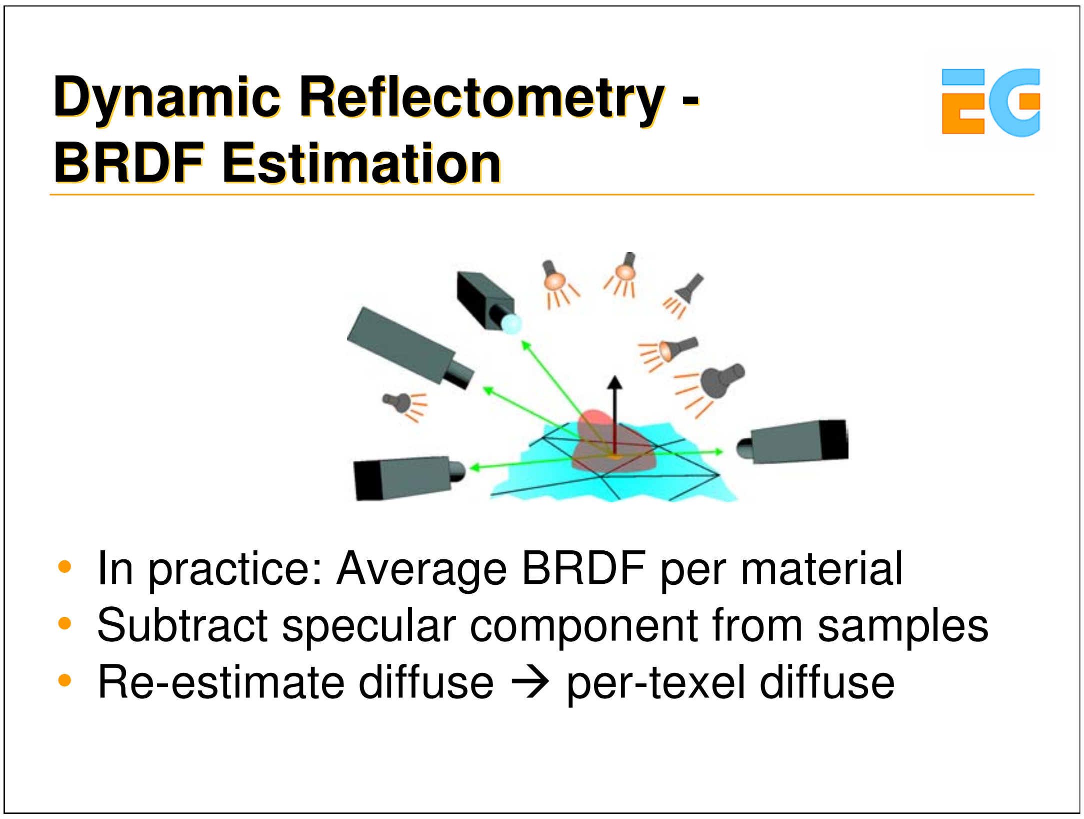
Minimize $$E_{BRDF}(\vec{x}, \boxed{\rho(\vec{x})}) = \sum_{t}^{N} \sum_{c}^{8} (\boxed{S_c(t)} - \boxed{P_c(t, \rho(\vec{x}))})^2$$

BRDF params

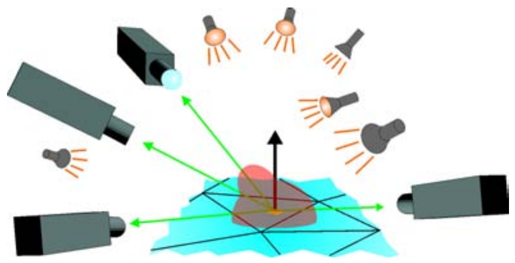
Reflectance sample

Predicted appearance

BRDF estimation is formulated as an energy minimization problem as it was proposed in a similar way in [6]. For each texel, the functional shown above is minimized in the BRDF parameters. The functional measures the quadratic error between the measured samples for that texel and the prediction according to the current BRDF parameters across all time steps of video and all camera views. Visibility from light sources and cameras is taken into account.

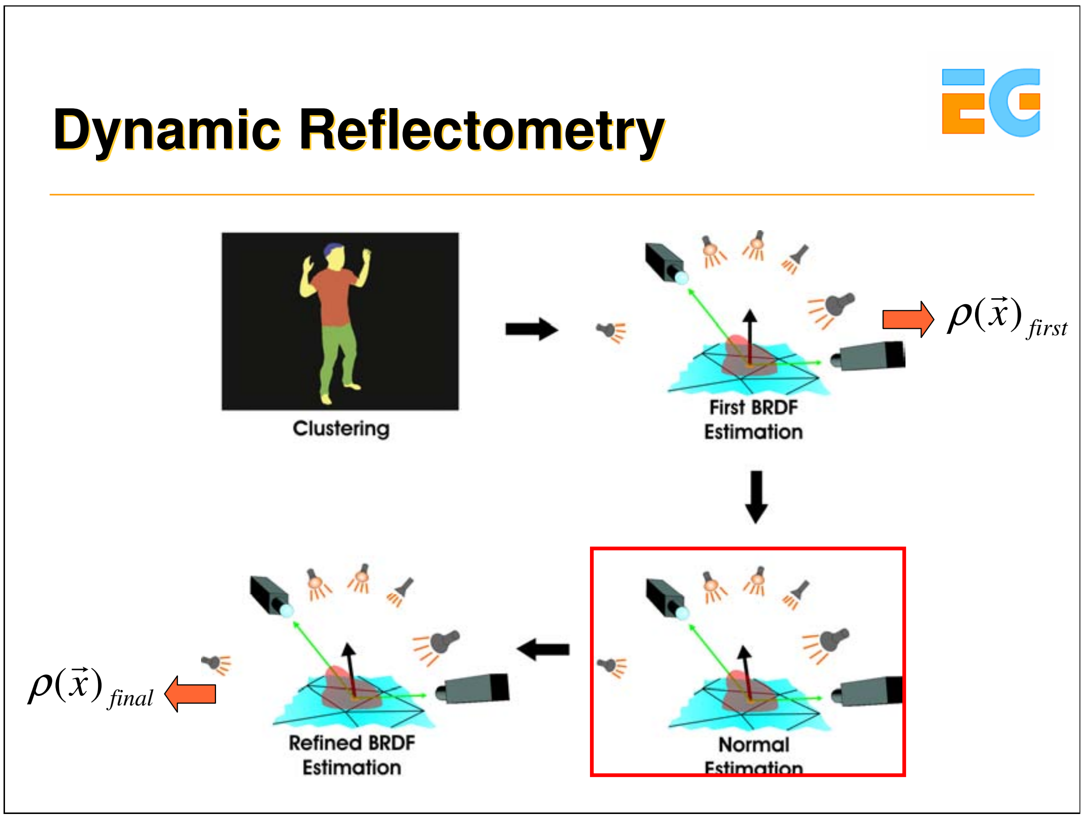


Dynamic Reflectometry - BRDF Estimation

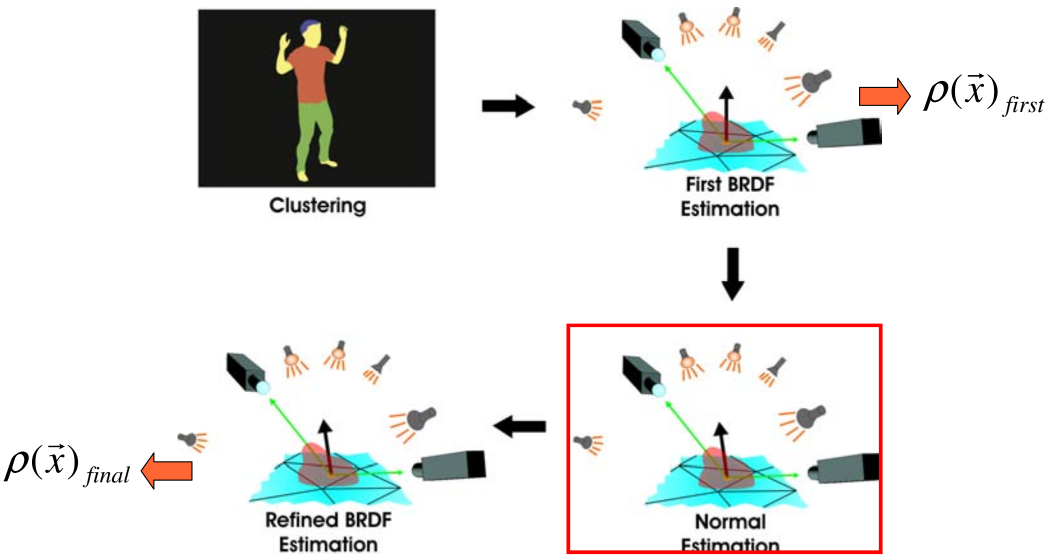


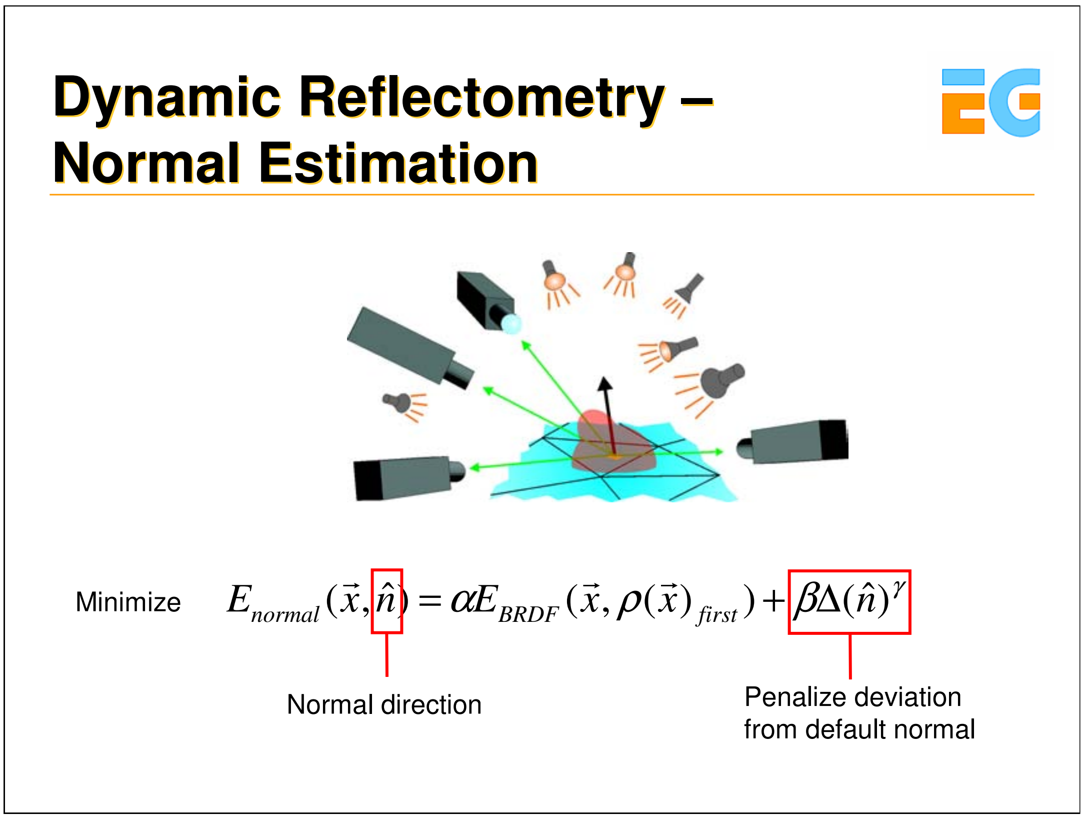
- In practice: Average BRDF per material
- Subtract specular component from samples
- Re-estimate diffuse → per-texel diffuse

Our BRDF models [2,4] treat specular and diffuse reflectance as separate terms. Specular reflectance is a high frequency signal and one has to make sure that a sufficient number of samples is available in order to properly reconstruct it. To obtain a sufficient sampling density of the space of reflectance samples we therefore estimate an average specular BRDF for each material cluster, but an individual diffuse component for each texel. In practice, BRDF estimation itself therefore comprises of two sub-steps. The first sub-step is the estimation of an average BRDF for each material. Using this average BRDF, purely diffuse reflectance samples are created by subtracting the estimated average specular components. Finally, a separate diffuse component for each texel is estimated using the purely diffuse samples.

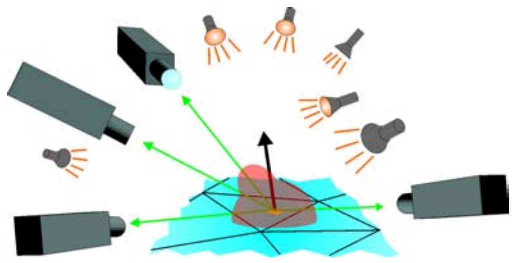


Dynamic Reflectometry





Dynamic Reflectometry – Normal Estimation



Minimize $$E_{normal}(\vec{x},\boxed{\hat{n}}) = \alpha E_{BRDF}(\vec{x},\rho(\vec{x})_{first}) + \boxed{\beta\Delta(\hat{n})^{\gamma}}$$

Normal direction

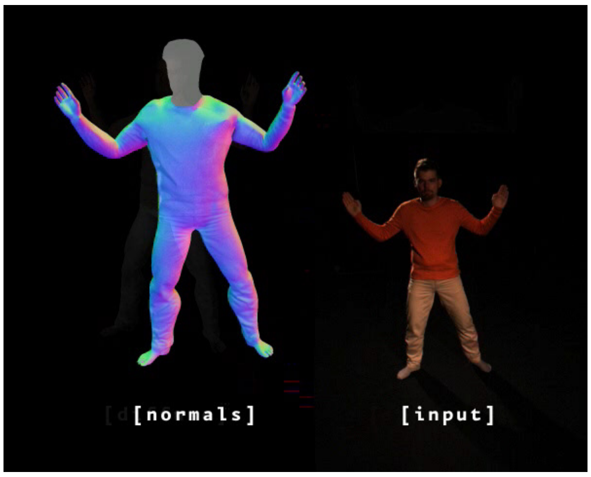
Penalize deviation from default normal

The normal estimation step capitalizes on the first set of estimated reflectance parameters to create a refined estimate of 3D shape. To this end, a new energy functional is employed which is shown in the slide above. The new functional is a weighted sum of original BRDF error functional and a second term which is used for regularization. The regularization term penalizes strong deviations of the estimated normal direction (the delta term) from the template normal. As before, the energy functional is minimized separately for each texel.

Dynamic Normal Maps

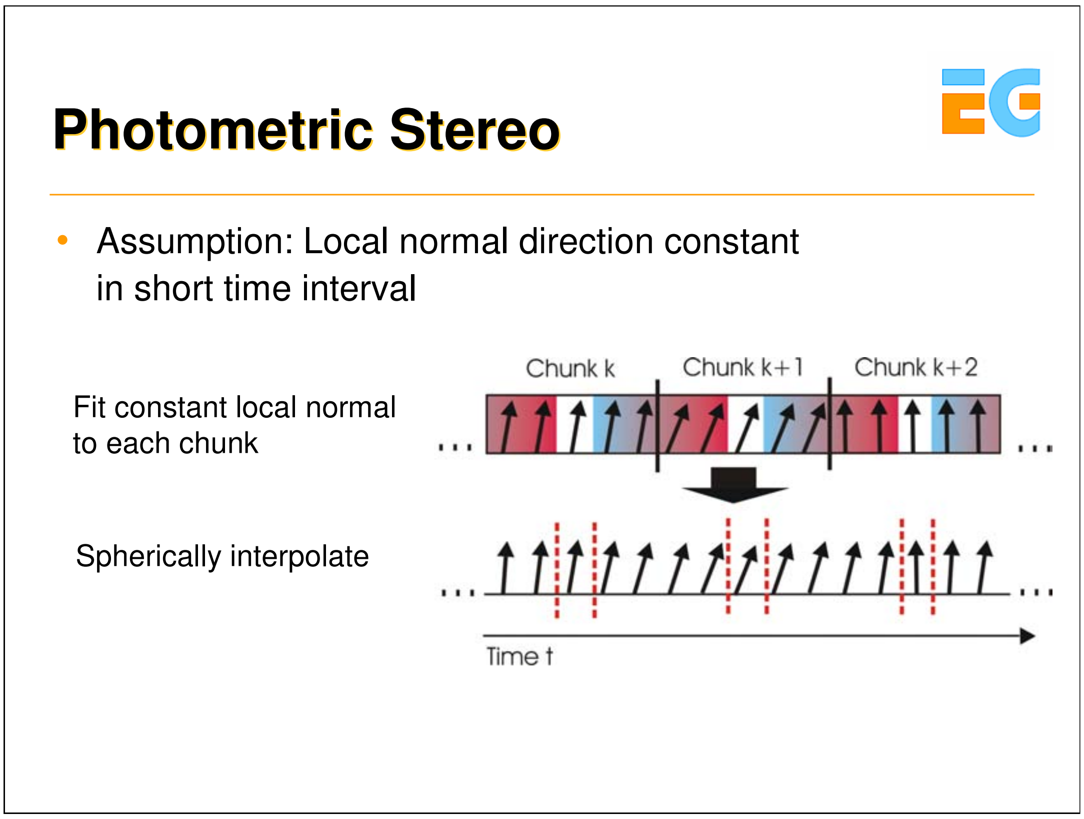
– Input: DSS

– Per-texel: Minimize energy



Once the BRDF parameters for each surface point are estimated and a refined estimate of surface geometry is available, we can reconstruct the time-varying normal field from the dynamic scene sequence in order to complete our reflectance description.

The procedure applied to recover the per-time step normal direction is again the energy minimization described on the previous slides. In contrast to the normal estimation which was part of the dynamic reflectometry process, however, we are now facing the problem of reconstructing a dynamic normal field.

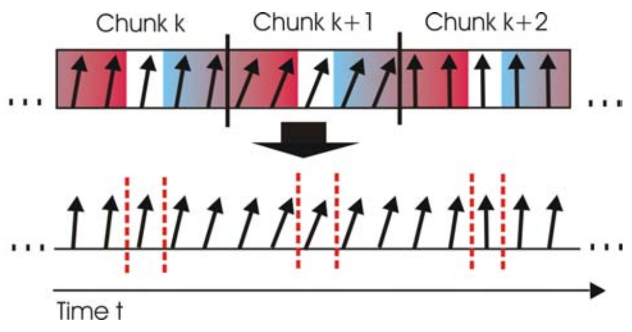


Photometric Stereo

- Assumption: Local normal direction constant in short time interval

Fit constant local normal to each chunk

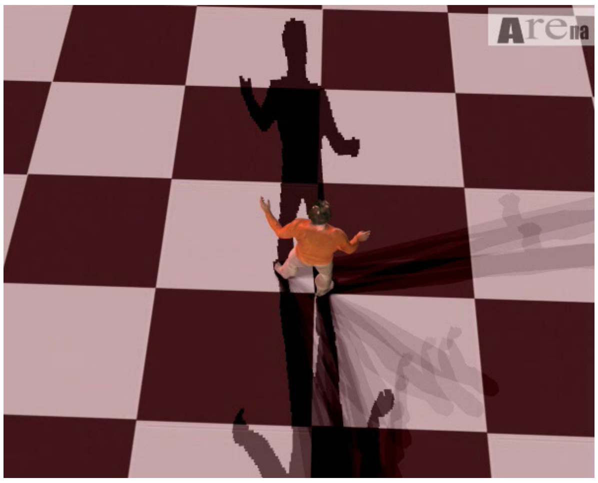
Spherically interpolate



In order to reconstruct a temporally smooth normal field, we employ the following procedure. First, we assume that the local normal direction of a texel does not change within a short window in time. By this means, we can combine reflectance samples from several subsequent time steps to estimate a single normal direction. Once the normal directions have been estimated on this coarser scale, we employ spherical linear interpolation to obtain normals that smoothly change their orientation. Please refer to [3] for details on this reconstruction procedure.

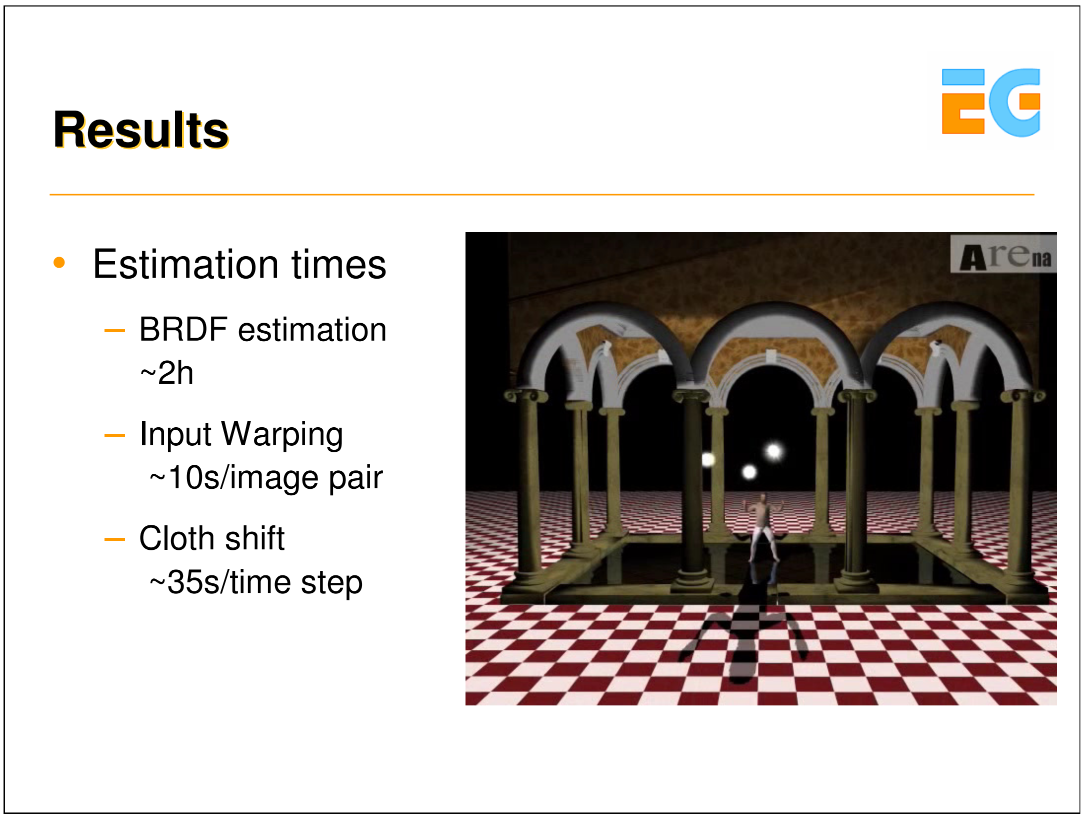
Results

- Lighting
  - Spot lights
  - Environment map
- On Pentium 4 3 GHz, GeForce 6800
  - 16 fps (4 lights)
  - 6 fps (16 lights)



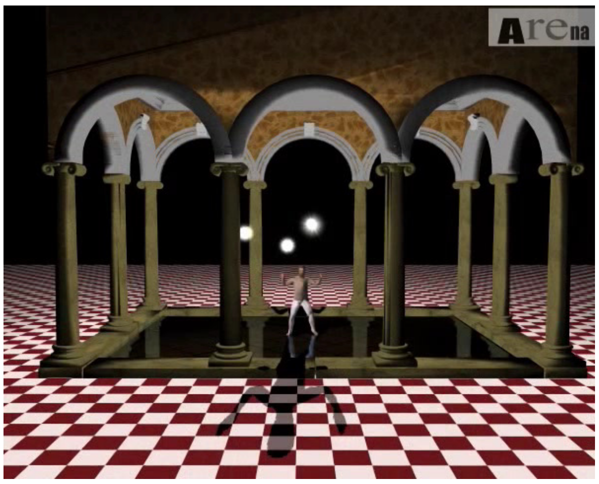
[Environment maps courtesy of Paul Debevec]

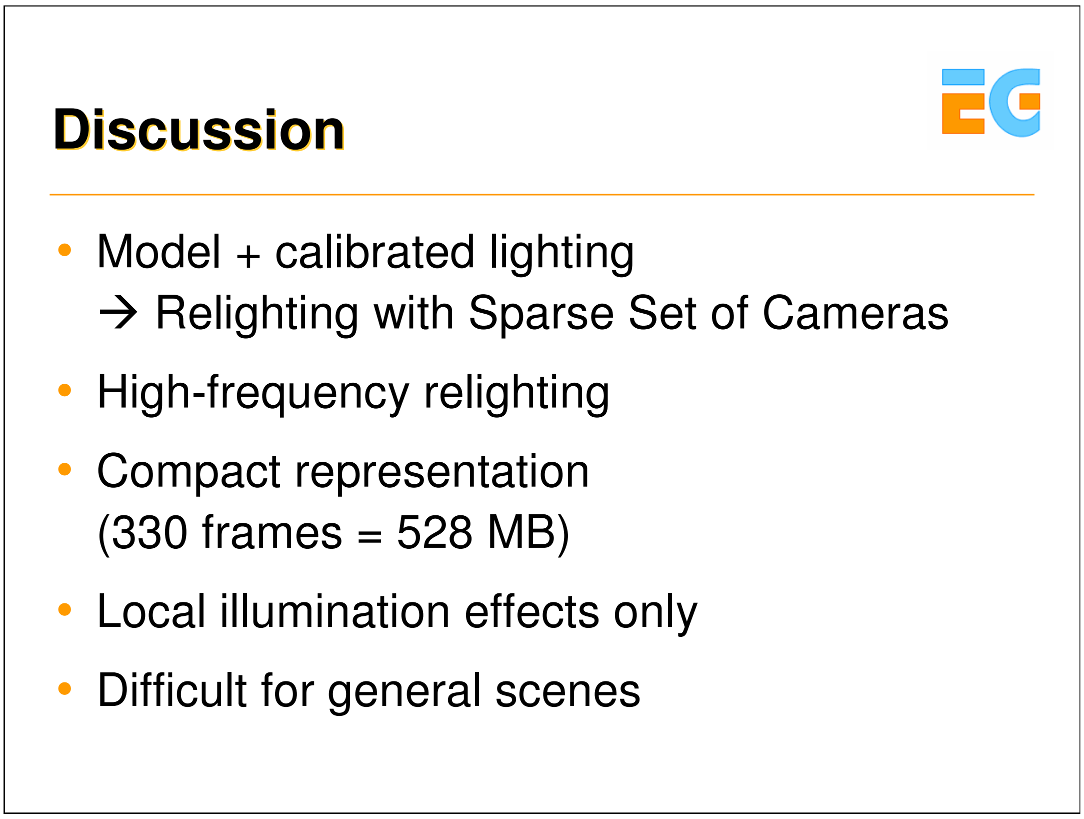
This and the following slides show several examples of relightable 3D videos rendered in real-time and lit from simulated and captured real-world illumination.



Results

- Estimation times
  - BRDF estimation ~2h
  - Input Warping ~10s/image pair
  - Cloth shift ~35s/time step



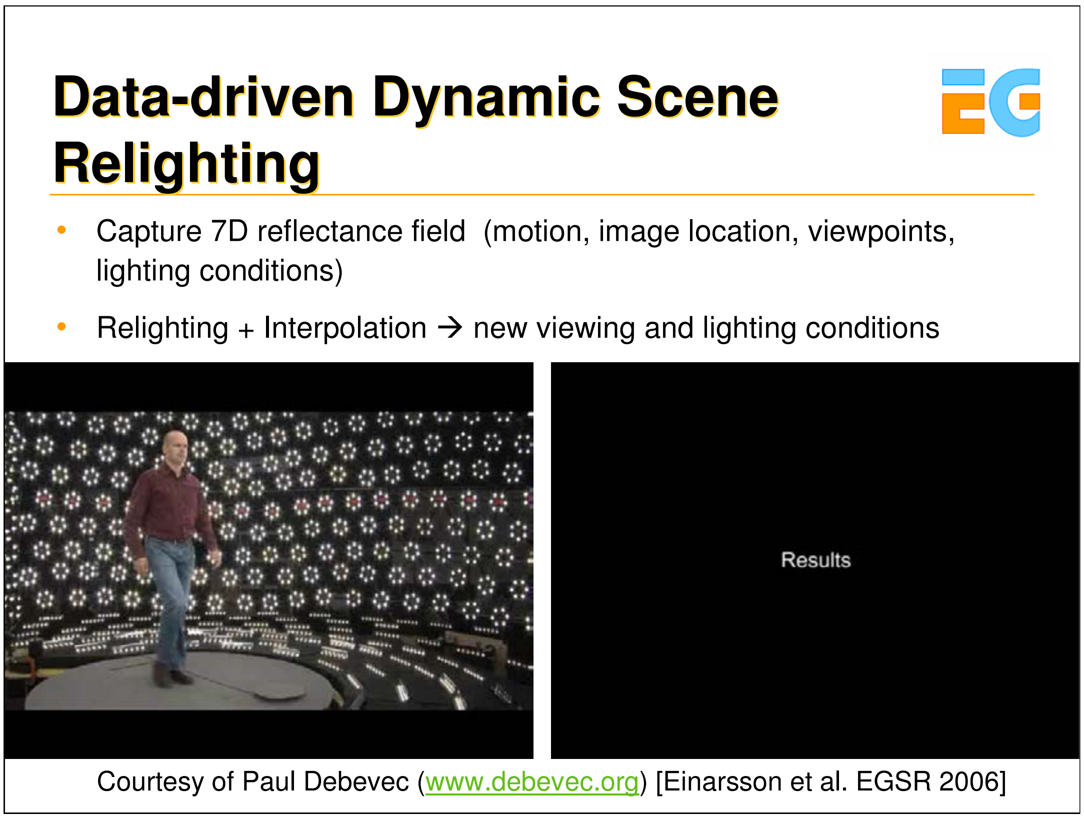


Discussion

- Model + calibrated lighting
  → Relighting with Sparse Set of Cameras
- High-frequency relighting
- Compact representation
  (330 frames = 528 MB)
- Local illumination effects only
- Difficult for general scenes

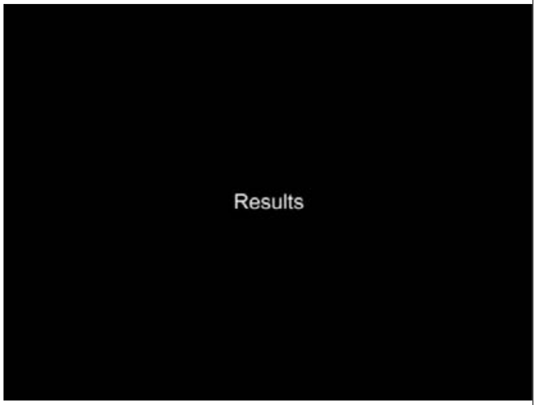
The model-based approach to dynamic scene relighting bears a couple of important advantages. First, 3D videos can be reconstructed using a moderately complex acquisition setup comprising of only a handful of cameras. Furthermore, the captured reflectance descriptions enable us to reproduce all-frequency lighting effects. An additional benefit is the fairly compact scene representation in terms of a dynamic geometry model, a static set of BRDF textures, and dynamic normal field and texture coordinate textures. The dancing sequence shown before that comprises of 330 frames of video has a total size of 528 MB. Our specific scene representation is very well suited for rendering on state-of-the-art graphics hardware.

Nonetheless, the model-based approaches also has several disadvantages. First, we can currently only reproduce local illumination effects. Secondly, for each type of subject in the scene a dedicated a priori model has to be available. Some of these limitations can be overcome by a data-driven approach to dynamic scene relighting.



Data-driven Dynamic Scene Relighting

- Capture 7D reflectance field (motion, image location, viewpoints, lighting conditions)
- Relighting + Interpolation → new viewing and lighting conditions



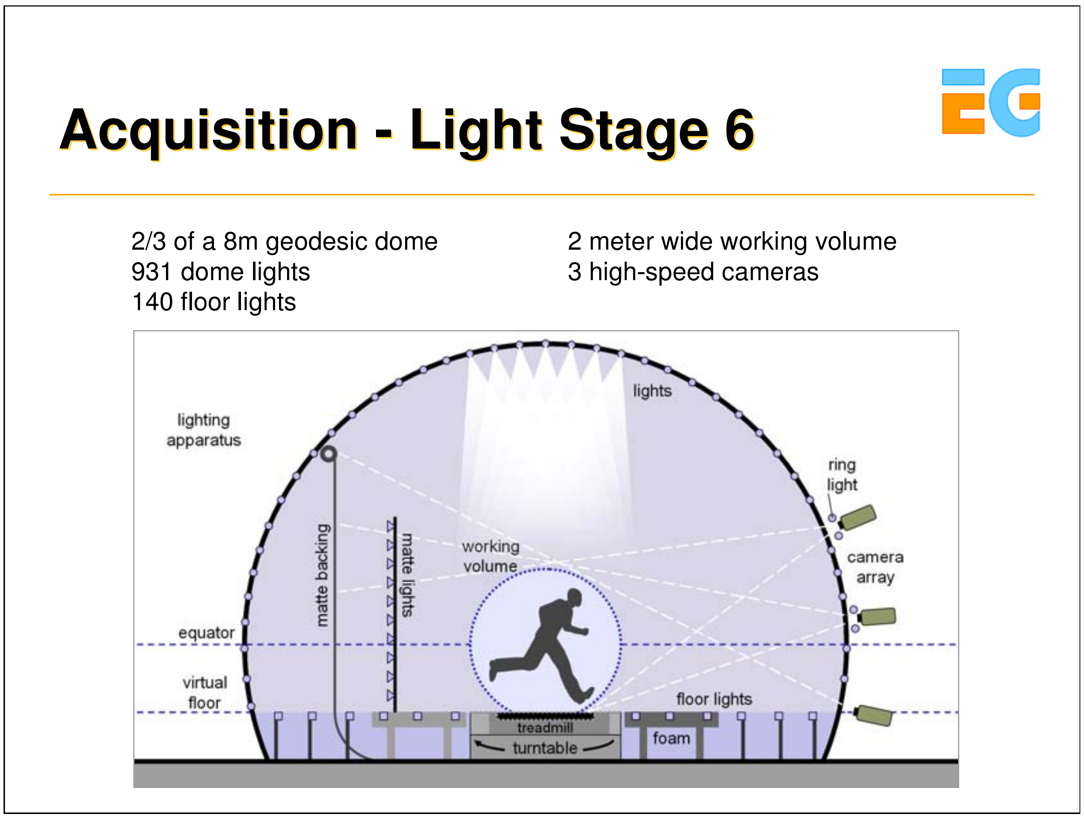
Courtesy of Paul Debevec (www.debevec.org) [Einarsson et al. EGSR 2006]

A data-driven approach to do dynamic scene relighting was developed in the lab of Paul Debevec at USC Los Angeles and is described in [7].

They built a device called Lightstage 6 that enables them to capture a 7-dimensional full dynamic reflectance field. This means they capture a complete set of images for dynamic scenes that spans 2dimensions for the images themselves, 2 dimensions (hemispherical directions) for the incident lighting, 2 dimensions for the viewpoints (hemispherical directions) and 1 dimension for time.

When rendering a novel viewpoint of a particular captured moment under novel lighting conditions, the novel lighting conditions are projected into the employed lighting basis and the images in the 7D data set are appropriately combined to generate the output view.

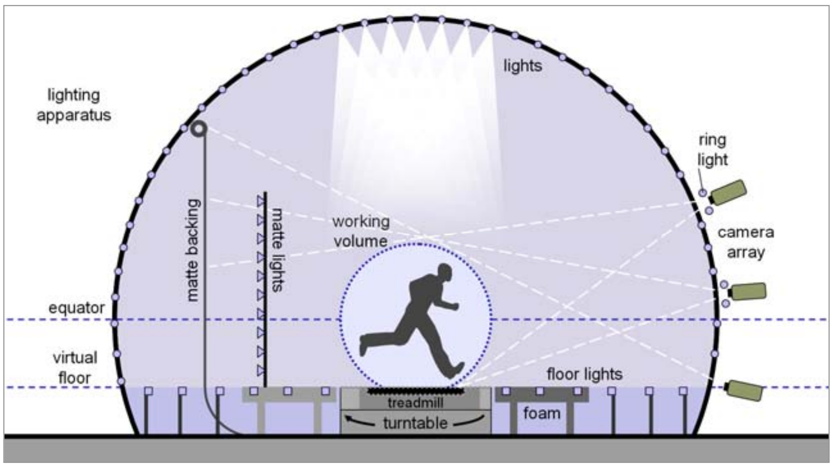
This and the following slides on this project were kindly provided by Paul Debevec.



Acquisition - Light Stage 6

2/3 of a 8m geodesic dome
931 dome lights
140 floor lights

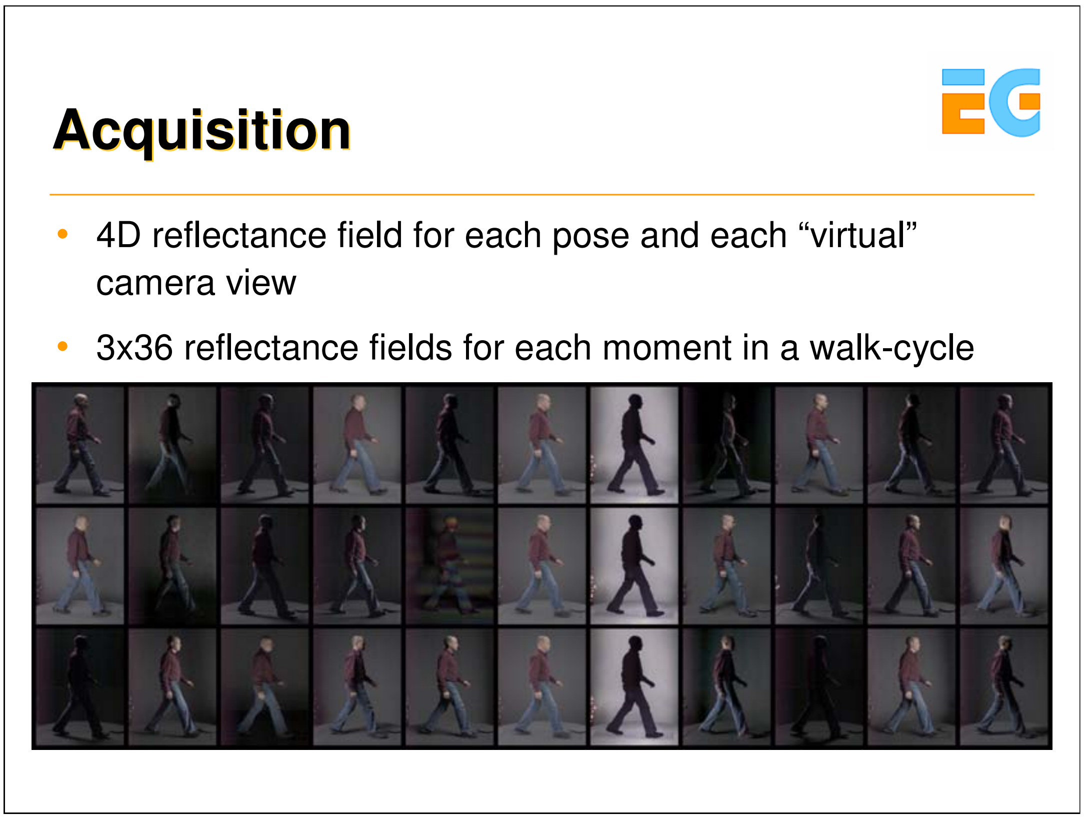
2 meter wide working volume
3 high-speed cameras



The acquisition setup, called Light stage 6, features 931 LED dome lights that can be switched on and off very rapidly. The lights are all mounted in a 2/3 full geodesic dome, 140 floor light simulate light bouncing from Lambertian ground plane.

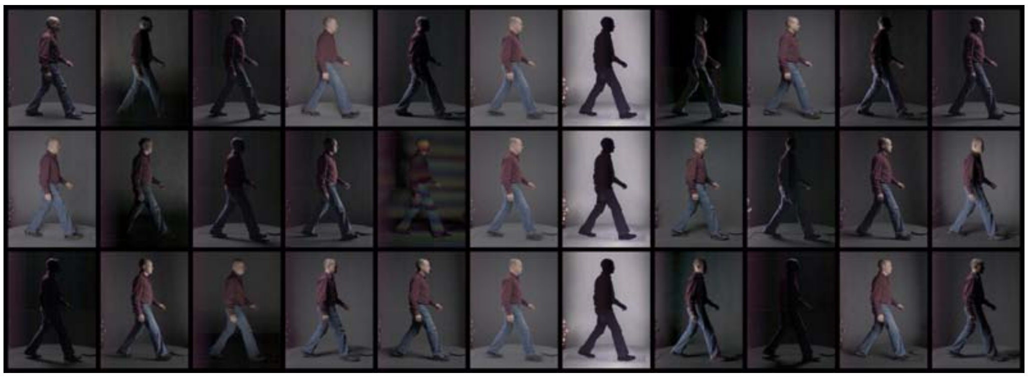
For video recording, three high-speed cameras are used that are running at roughly 1000 frames per second. The light sources are triggered in synchronization with the cameras. Due to the high frame rate of the cameras, each pose of the actor can be illuminated and recorded under a full set of 26 different basis lighting conditions. The frame rate of the cameras is enough to capture 30 complete lighting cycles per second.

As high-speed cameras are still very expensive, the authors restrict their setup and only use 3 cameras instead of a full dome of imaging sensors. They also use a treadmill which can be rotated in the setup and restrict captured scenes to cyclic human motion like walking. This way, the authors can simulate Nx3 virtual recording cameras although only 3 cameras are physically available. However, this simplification also comes at the cost of restrictions in motion generality.



Acquisition

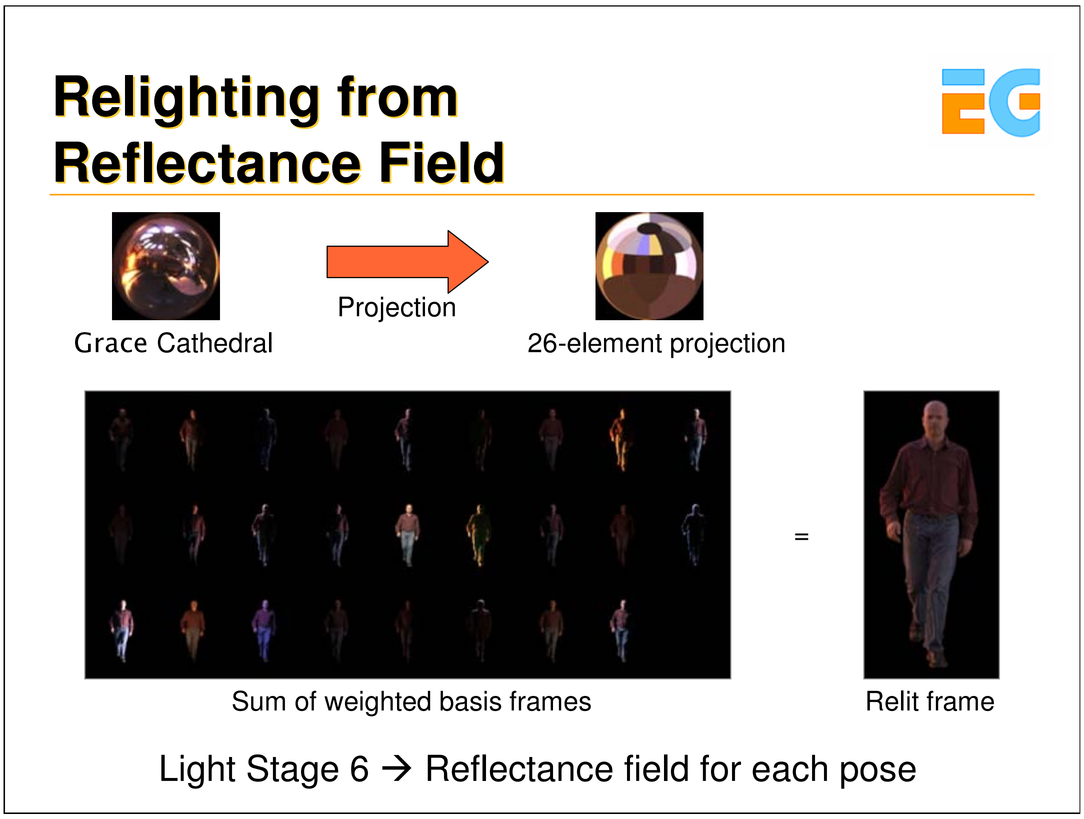
- 4D reflectance field for each pose and each “virtual” camera view
- 3x36 reflectance fields for each moment in a walk-cycle



For each pose of the walking cycle, i.e. each 1/30 of a second, a full 4D reflectance field is captured. A reflectance field is a function that transforms an incident light field hitting a surface to an outgoing light field exiting the surface [8,9]. Therefore, in its most general form, it is an 8-dimensional function (position + direction incomig; position + direction outgoing). If the light can be assumed to arrive from infinity and if the output viewpoint is fixed, as in this case, the dimensionality of the reflectance field reduces to 4.

In this particular case, the reflectance field is simply parameterized as a set of images, each one being illuminated by one of the basis conditions.

In other words, the light stage enables capturing a 3x36 array of 4D light fields for each moment of a walking cycle.



Relighting from Reflectance Field

Grace Cathedral

Projection

26-element projection

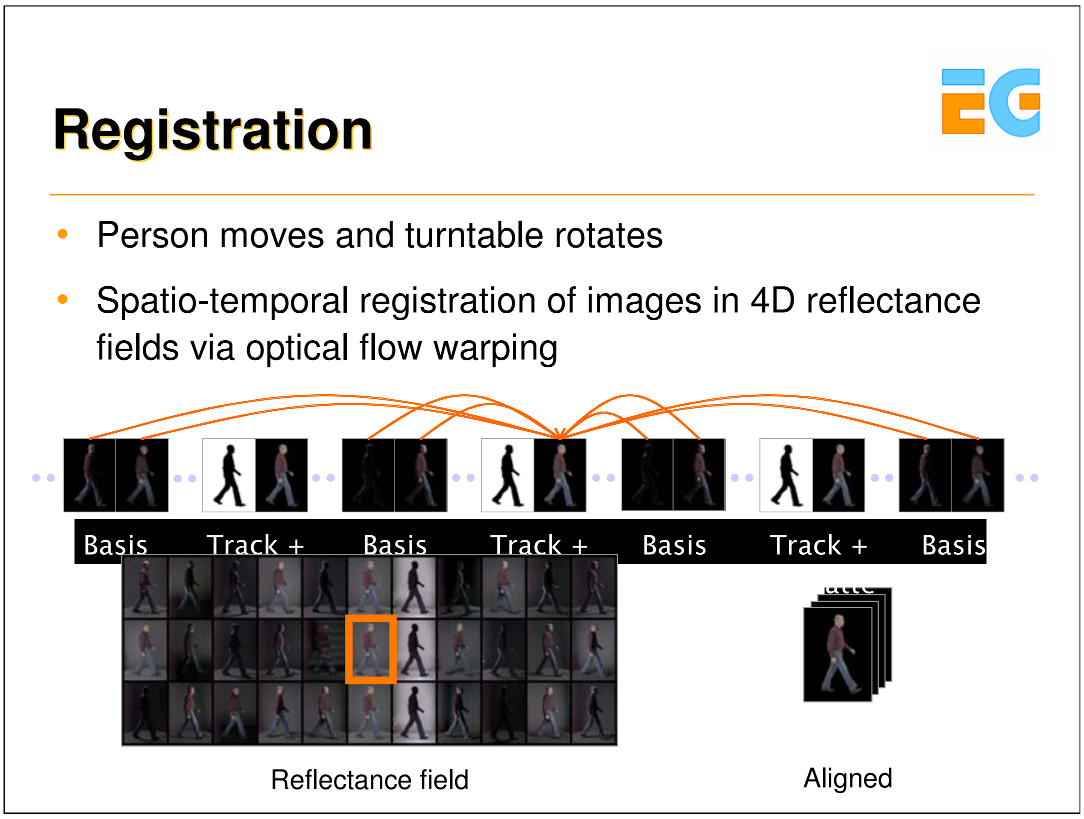
Sum of weighted basis frames

=

Relit frame

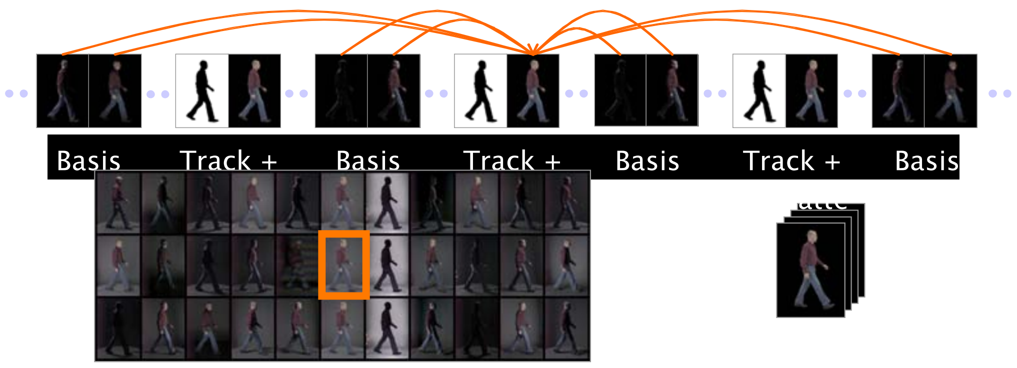
Light Stage 6 → Reflectance field for each pose

As mentioned before, a reflectance field transforms incident light fields to exitant light fields (see [9] for a definition of light field). Before the person is rendered under a novel illumination condition, the new illumination is projected into the lighting basis and the basis images in all 4D reflectance fields are accordingly scaled. A novel view of the actor is obtained by weightedly summing up the scaled basis images. By this means the flowed reflectance fields originally captured turn into a flowed light field, i.e. a set of relit images showing the person from each possible camera view and at each possible moment of the walking cycle. Please refer to [8] for details.



Registration

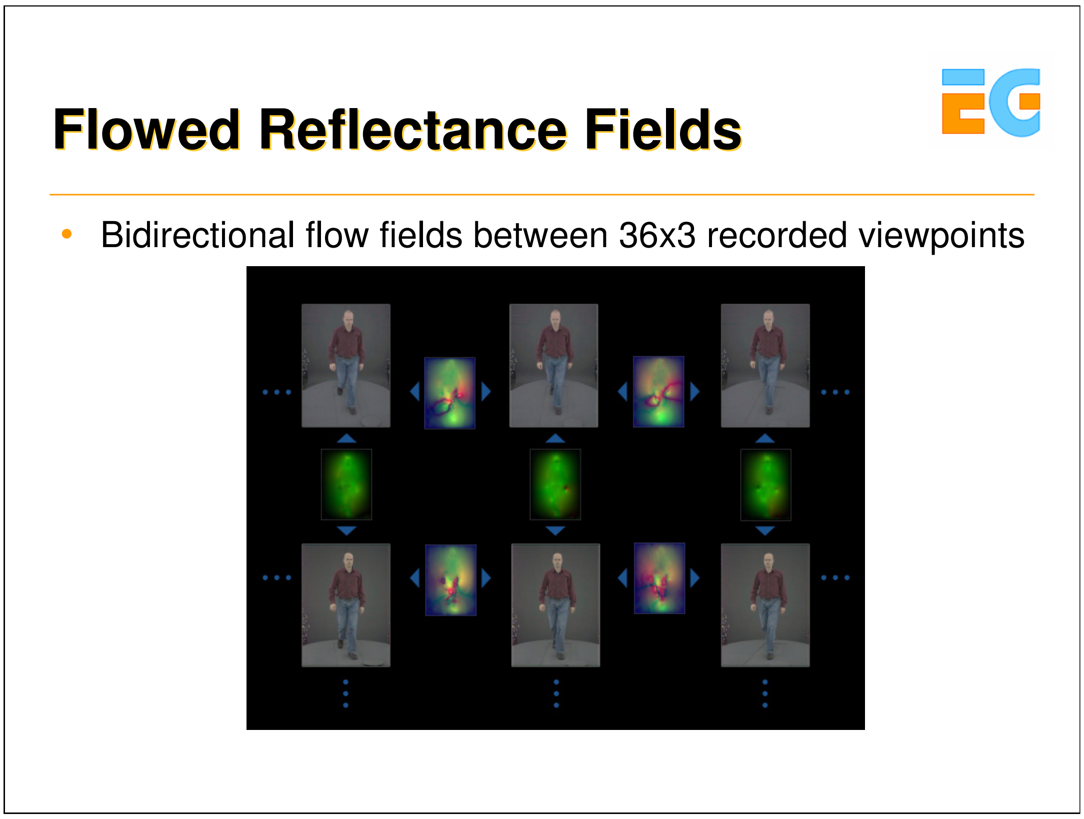
- Person moves and turntable rotates
- Spatio-temporal registration of images in 4D reflectance fields via optical flow warping



Reflectance field

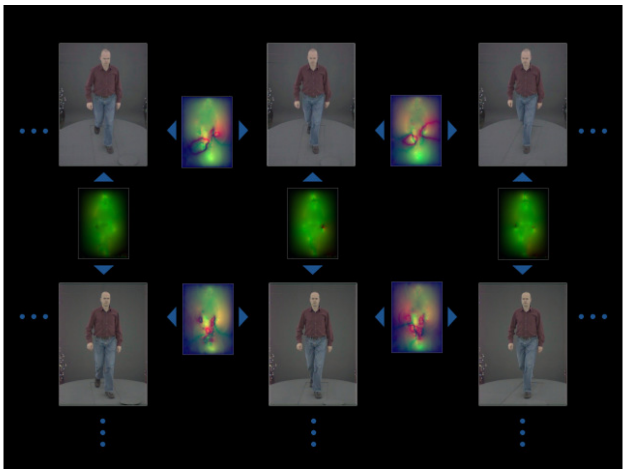
Aligned

Since the turntable is rotating all the time, and the person is permanently moving a couple of registration problems have to be solved. For each complete lighting cycle (1/30s), a complete 4D reflectance field is acquired. However, as the person is permanently moving and the turntable is permanently rotating, the acquired images in the reflectance field are not properly aligned. Therefore, in a post-processing step all images captured during one lighting cycle (i.e. for each pose and each rotational increment) are aligned with the center frame by using an optical flow based warp correction. By this means, a so-called flowed reflectance field is obtained.

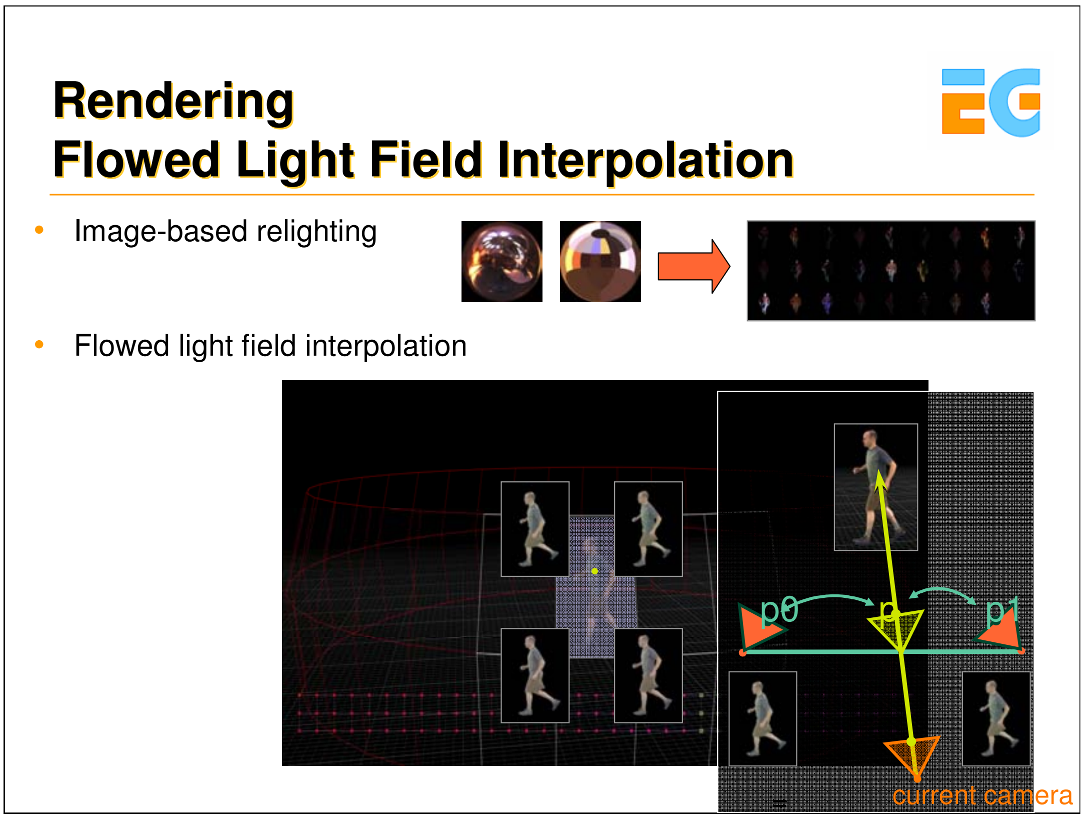


Flowed Reflectance Fields

- Bidirectional flow fields between 36x3 recorded viewpoints

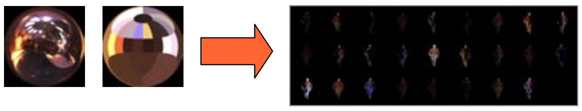


During recording, new viewpoints are captured at 10 degree rotational increments of the turntable. Therefore, 36x3 virtual viewpoints of the scene are obtained. In order to be able to later generate arbitrary viewpoints in-between recording cameras, optical flow fields between adjacent 4D reflectance fields in the set of 108 viewpoints are generated. By this means a so-called flowed reflectance field is obtained.

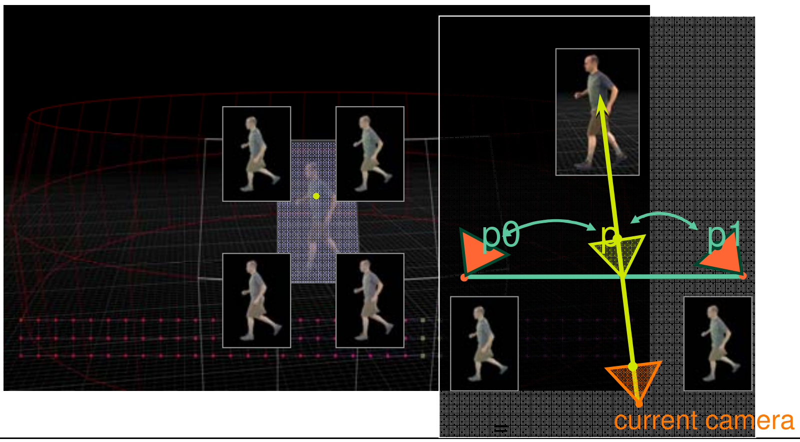


Rendering
Flowed Light Field Interpolation

- Image-based relighting



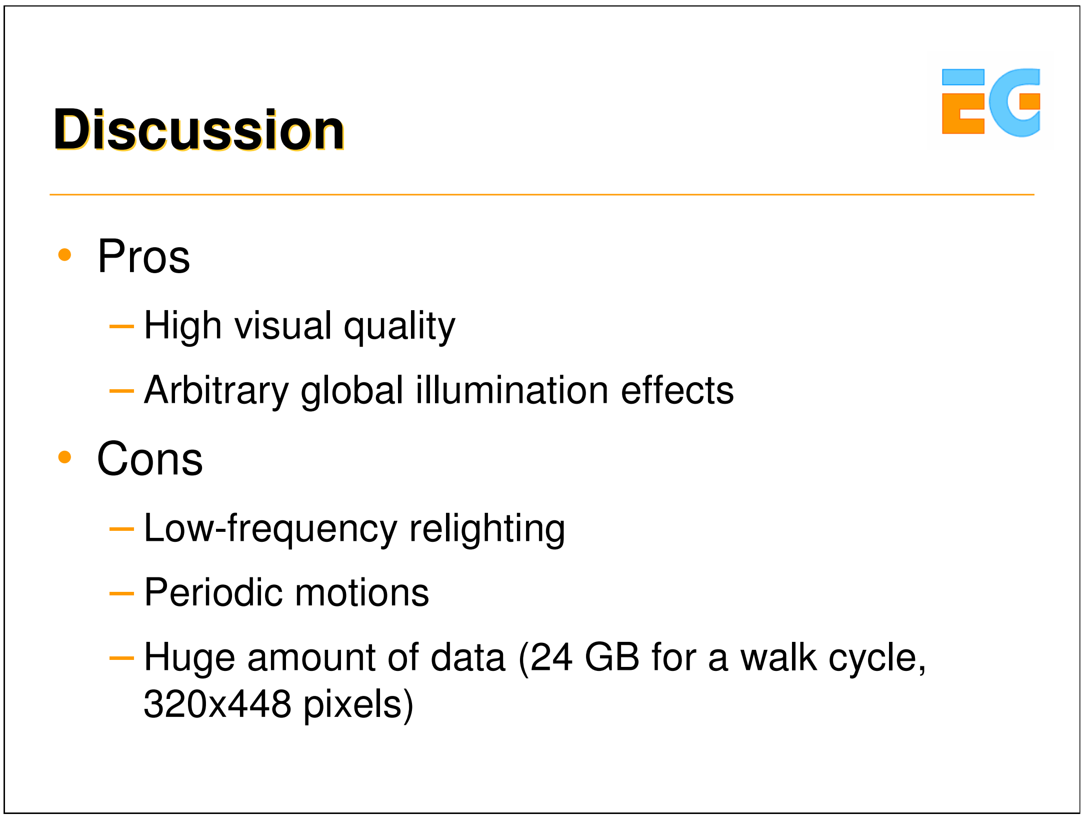
- Flowed light field interpolation



Finally, a novel viewpoint of a person in-between a recording camera viewpoint and under novel lighting conditions is obtained by using a process termed flowed light field interpolation:

First, the set of captured data is relit according to the new synthetic lighting conditions using the image-based relighting approach presented before [8] which yields a flowed light field.

Second, to generate any arbitrary viewpoint in-between true recording cameras, a warping-based approach is used to combine pixel information from the closest nearby relit images in the flowed light field. The pixel information from the closest nearby true cameras is combined using a light field rendering process similar to [9, 10]. To this end, geometric relations between the virtual camera and the four surrounding recording cameras are used to determine how much influence the pixels from each camera get in the final output view. Proper pixel locations for blending are looked up in the true camera views by following the previously computed bidirectional flow fields in a backward direction.

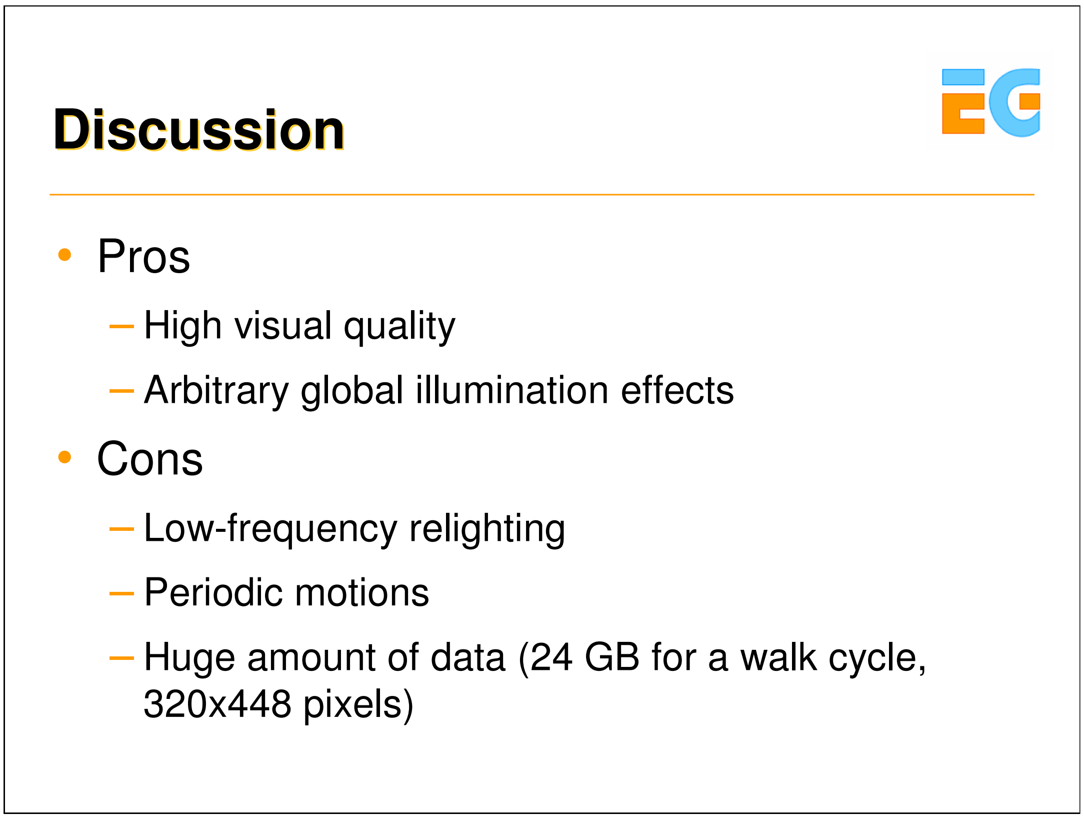


Discussion

- Pros
  - High visual quality
  - Arbitrary global illumination effects
- Cons
  - Low-frequency relighting
  - Periodic motions
  - Huge amount of data (24 GB for a walk cycle, 320x448 pixels)

The data-driven relighting approach has a couple of advantages. First, the quality of the rendered and relit views is fairly high if the array of recording cameras is sufficiently dense. In addition to that, being based on the captured images themselves, the data-driven representation can inherently reproduce both local and global lighting effects under any novel incident illumination.

On the other hand, the proposed method also has a couple of disadvantages. First, only low frequency lighting effects can be reproduced as the employed lighting basis is very coarse. Furthermore, the huge amount of data even at low image resolutions, the high engineering overhead and the current limitation to very short cyclic motion may make the approach infeasible for many applications.



Discussion

- Pros
  - High visual quality
  - Arbitrary global illumination effects
- Cons
  - Low-frequency relighting
  - Periodic motions
  - Huge amount of data (24 GB for a walk cycle, 320x448 pixels)

References

•[1] R. L. Carceroni and K. Kutulakos. *Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape, and reflectance*. In Proc ICCV 2001.

•[2] B.-T. Phong. Illumnation for computer generated pictures. *Communications of the ACM*, pages 311.317, 1975.

•[3] C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H.-P. Seidel, *Seeing People in Different Light: Joint Shape, Motion, and Reflectance Capture*, to appear in IEEE Transactions on Visualization and Computer Graphics, 13(4), p. 663-674, 2007.

•[4] Lafortune, E. P., Foo, S., Torrance, K. E., and Greenberg, D. P. 1997. *Non-linear approximation of reflectance functions*. In Proc. SIGGRAPH. ACM Press, 1997, p. 117-126.

•[5] N Ahmed, C. Theobalt, M. Magnor, H.-P. Seidel, *Spatio-Temporal Registration Techniques for Relightable 3D Video*, Proc. of IEEE ICIP, 2007, to appear.

•[6] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich and Hans-Peter Seidel. *Image-based Reconstruction of Spatial Appearance and Geometric Detail*. In ACM Transactions on Graphics, 22(2), 2003, pages 234-257.

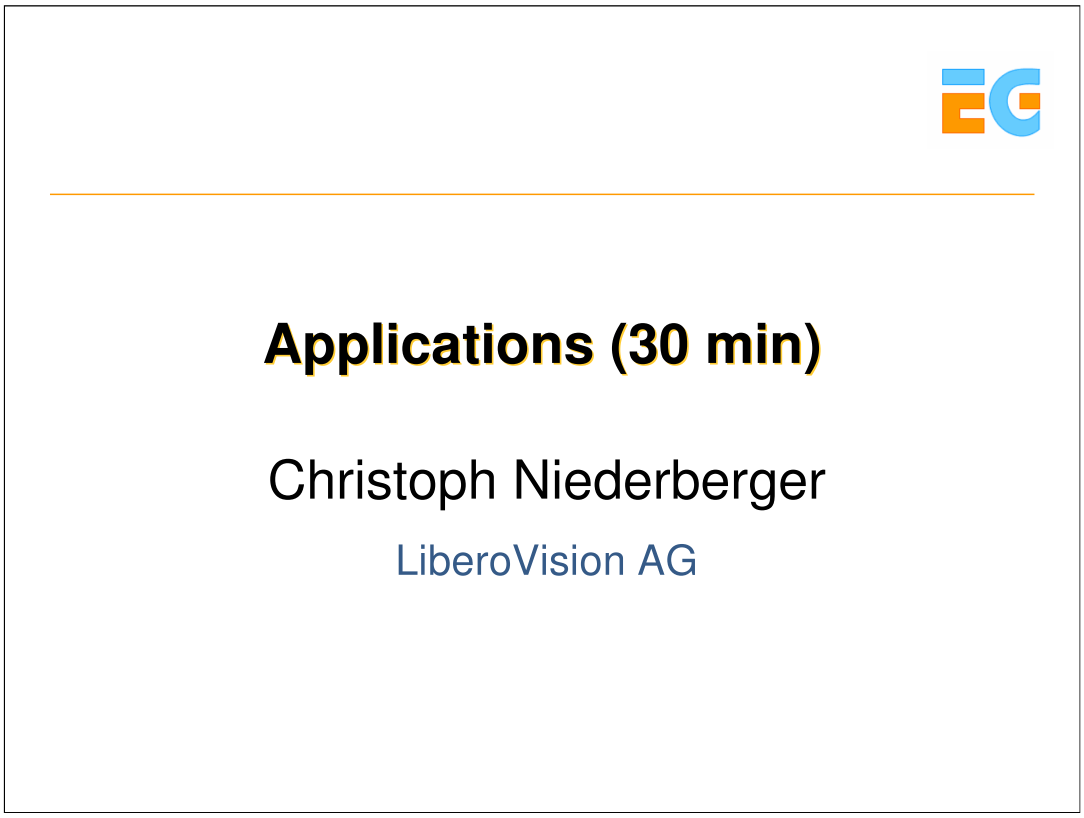
•[7] Einarsson P., Chabert C., Jones A., Lamond B., Ma A., Hawkins T., Sylwan S., Debevec P. *Relighting human locomotion with flowed reflectance fields*. Proc. of EGSR 2006

•[] Wenger, A., Gardner, A., Tchou, C., Unger, J., Hawkins, T., and Debevec, P. 2005. Performance relighting and reflectance transformation with time-multiplexed illumination. In *ACM SIGGRAPH 2005 Papers, p. 756-764* .

•[8] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, Mark Sagar. *Acquiring the Reflectance Field of a Human Face*, Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, pp. 145-156 (July 2000). ACM Press

•[9] Levoy, M. and Hanrahan, P. Light field rendering. In Proc. of SIGGRAPH '96.

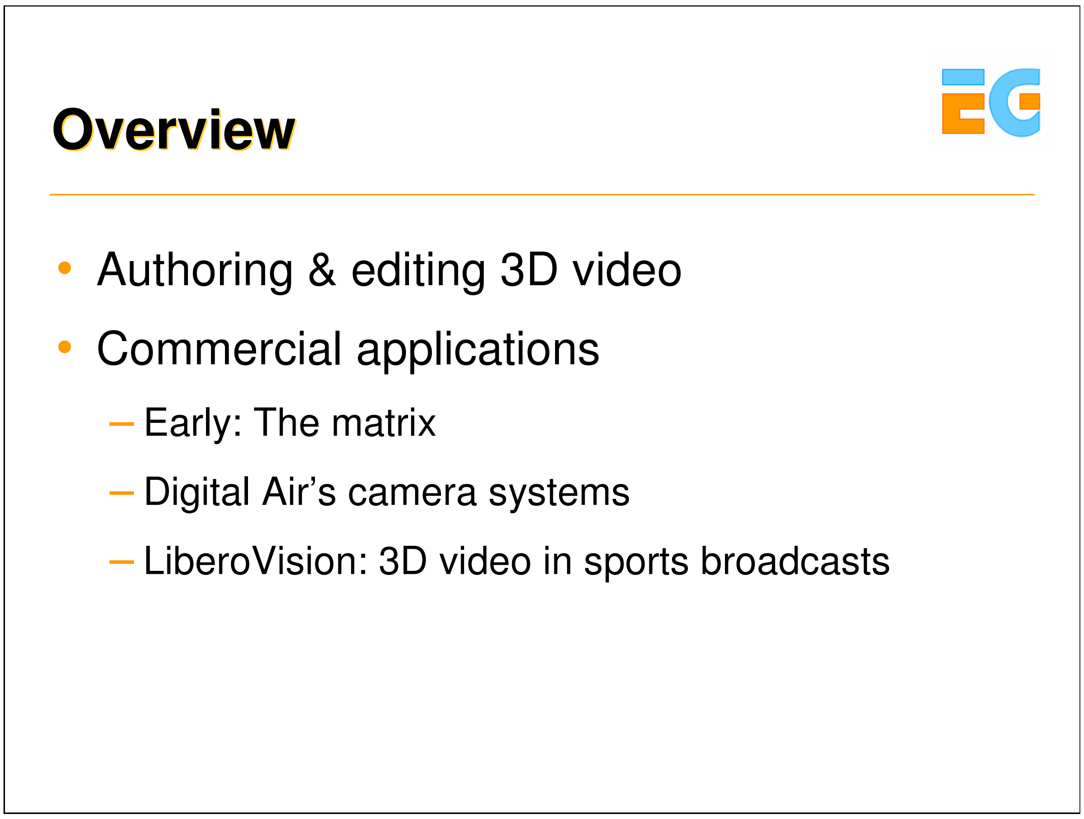
•[10] Buehler, C., Bosse, M., McMillan, L., Gortler, S., and Cohen, M. Unstructured lumigraph rendering. In Proc. SIGGRAPH 2001.



Applications (30 min)

Christoph Niederberger

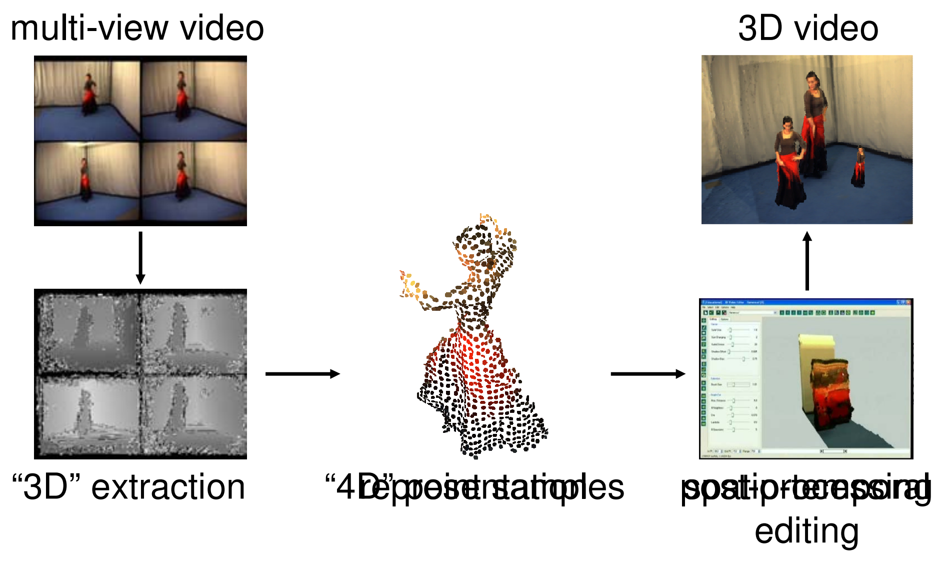
LiberoVision AG



Overview

- Authoring & editing 3D video
- Commercial applications
  - Early: The matrix
  - Digital Air’s camera systems
  - LiberoVision: 3D video in sports broadcasts

Need for Authoring Tools!

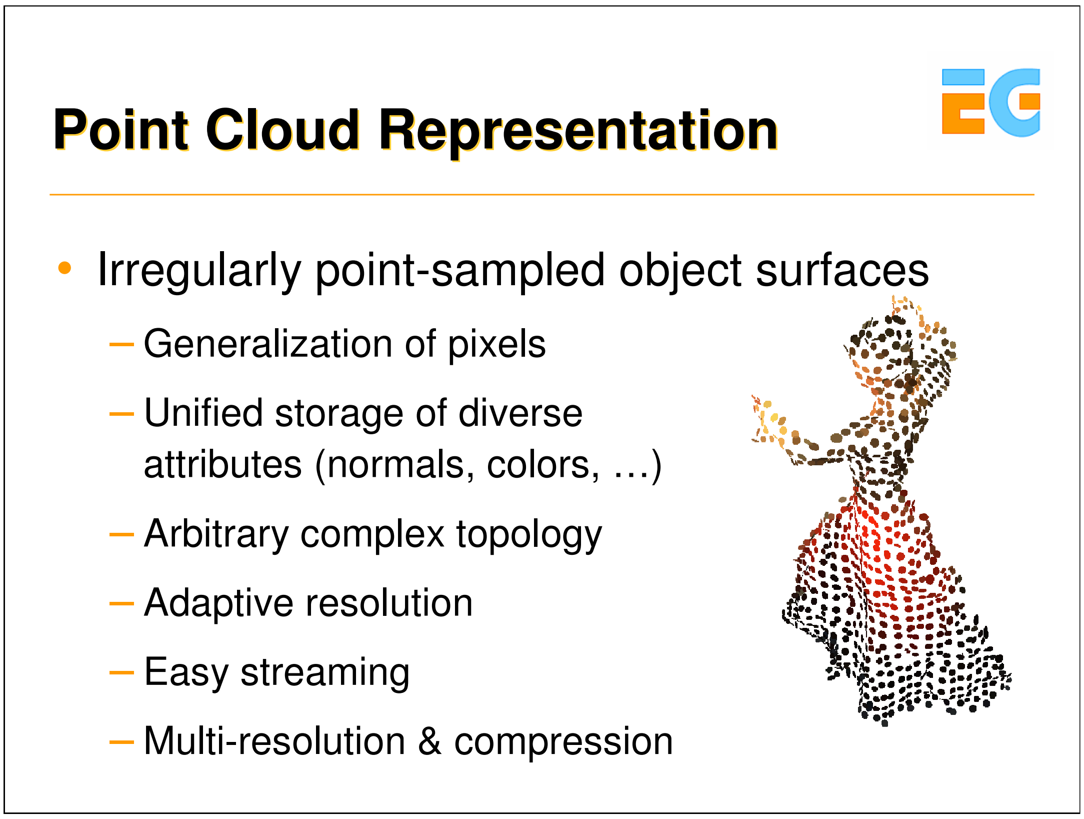


If we have captured a scene from multiple viewpoints, authorign tools are necessary to edit this content. In this slide, an overview of the processing pipeline is shown.

Based on a multi-view video acquisition of a scene, a 4D representation is generated based on the extraction of the three-dimensional data. Usually, this relies on the extracted depth information.

To edit and author such 3D video data, we need a new type of authoring tools. Such a tool must be able to extract objects or parts of a scene over a certain time and place it into another 3D video environment. Thus, such a tool can be envisioned as a cross-over between a video editor and a 3D modeling tool.

Finally, the edited scene can be previewed such that the final trajectory of the camera can be set up.



Point Cloud Representation

- Irregularly point-sampled object surfaces
  - Generalization of pixels
  - Unified storage of diverse attributes (normals, colors, …)
  - Arbitrary complex topology
  - Adaptive resolution
  - Easy streaming
  - Multi-resolution & compression

To represent the scene geometry, we are not using triangle meshes but we suggest to use points where each point is a sample of a surface in the scene.

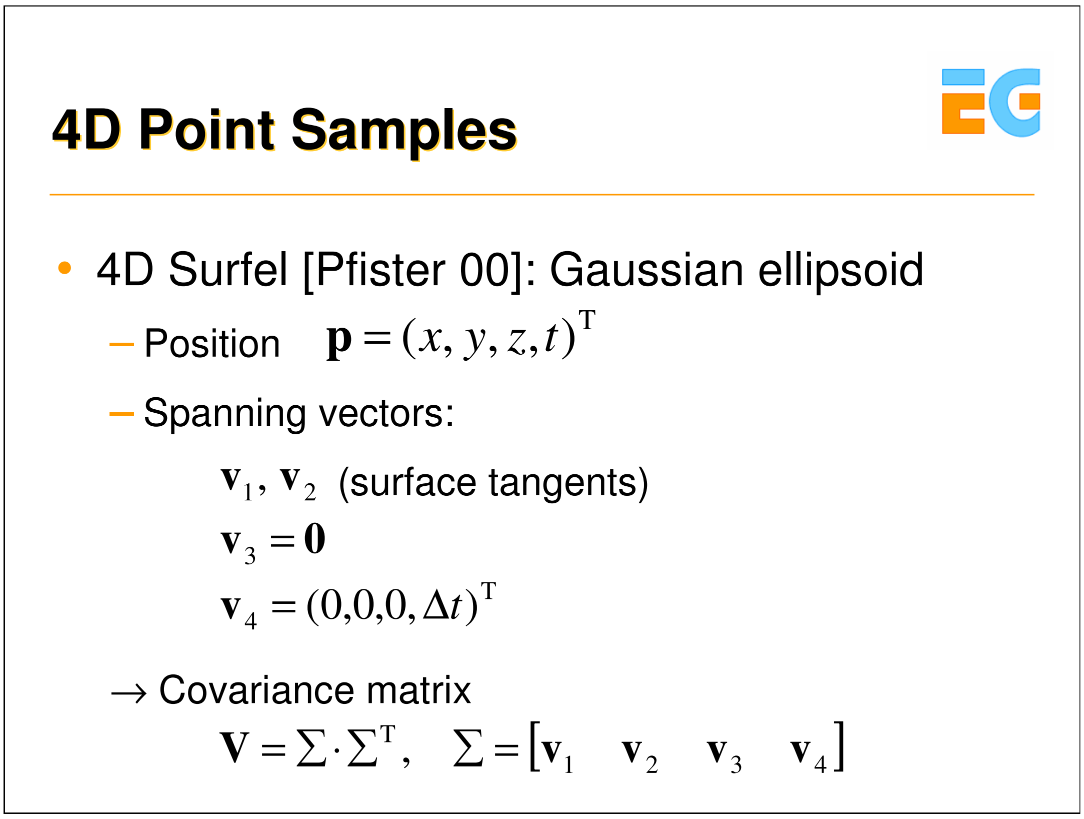
Such a point cloud can be generated quite easily from depth maps by back-projecting all depth pixels into a common 3D world coordinate system. As such, points can be considered as a generalization of 2D image pixels.

If the acquisition system does not produce depth maps but uses another approach to for example directly construct a mesh, this can be also converted quite easily into a point cloud by re-sampling.

Unlike meshes, points provide a unified storage container for all the data you need. Apart from geometry, they can carry information like surface normals, colors, or other material properties. You do not need an additional data structure like a texture to represent such attributes.

Points cannot represent the topology of a scene. This is a clear advantage in our case because this allows us for handling arbitrary complex topologies. More specifically, we can even represent dynamically changing topologies in the 3D video without much effort.

Unlike for example voxel grids, points only represent object surfaces and not their interior or the free air. As a consequence, we are not limited in scene resolution. But we can locally adapt the resolution to the scene complexity by irregular sampling.



4D Point Samples

- 4D Surfel [Pfister 00]: Gaussian ellipsoid
  - Position $\mathbf{p} = (x, y, z, t)^{\mathrm{T}}$
  - Spanning vectors:

    $\mathbf{v}_1, \mathbf{v}_2$ (surface tangents)

    $\mathbf{v}_3 = \mathbf{0}$

    $\mathbf{v}_4 = (0,0,0,\Delta t)^{\mathrm{T}}$

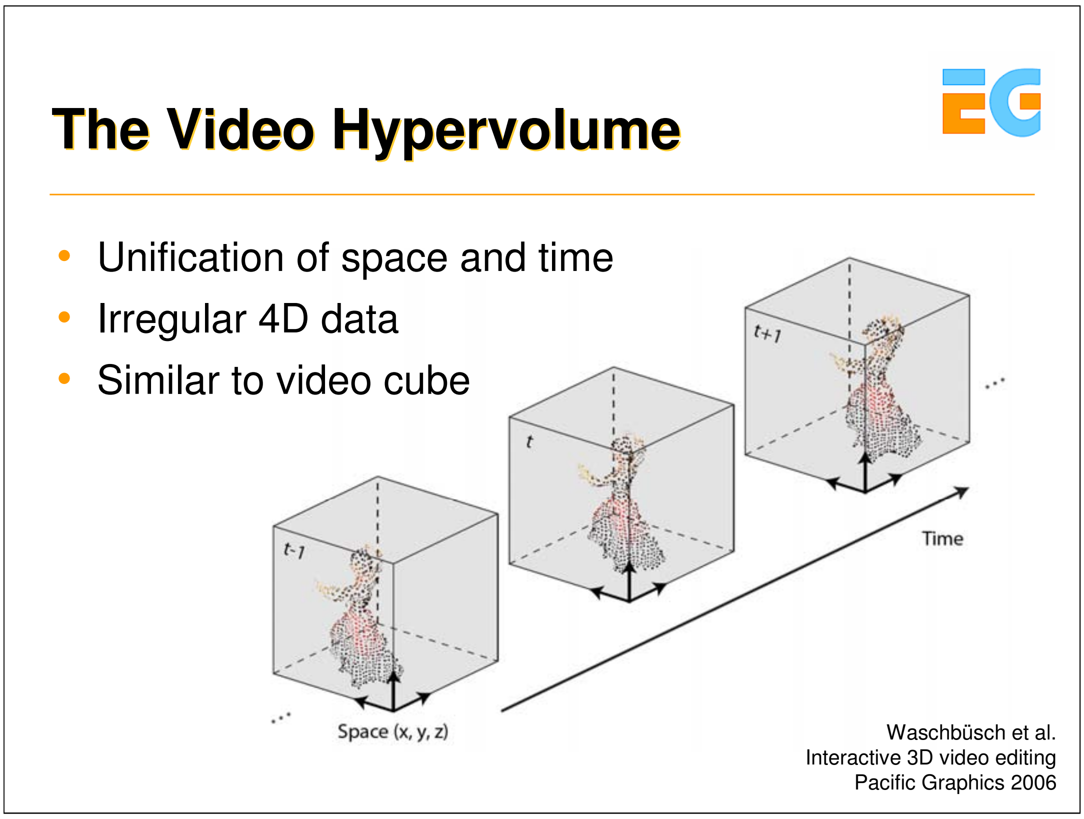
  $\rightarrow$ Covariance matrix

  $$\mathbf{V} = \Sigma \cdot \Sigma^{\mathrm{T}}, \quad \Sigma = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 \end{bmatrix}$$

We extend those Surfels to our 4D setting by adding a temporal coordinate t.

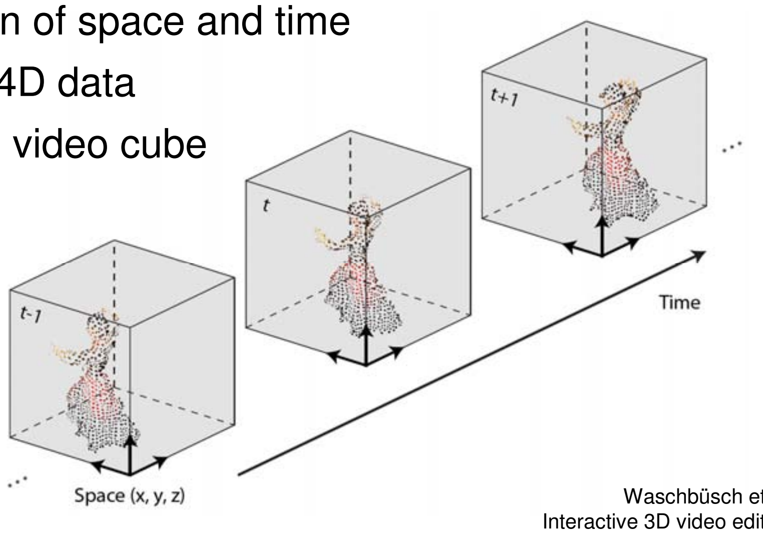
For the computation of the covariance matrix we introduce a fourth vector parallel to the time axis. Its length corresponds to the distance of two successive frames. This gives us smooth renderings if we want to visualize the temporal domain of the video hypervolume.

Alternatively, the covariance matrix can provide a probabilistic model of the geometric uncertainty, as has been introduced in our previous paper. This allows to model inaccuracies in the scanning due to noise or calibration errors.



The Video Hypervolume

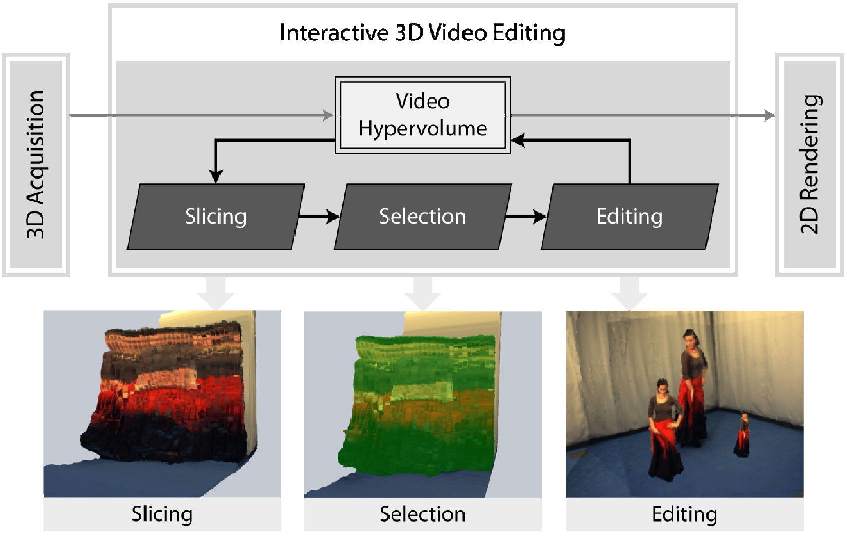
- Unification of space and time
- Irregular 4D data
- Similar to video cube



Waschbüsch et al.
Interactive 3D video editing
Pacific Graphics 2006

The 3D video hypervolume is a representation unifying point clouds over time. For each moment in time t-1, t, t+1, …, we have a complete 3D space irregularly filled with point samples representing the objects in the scene. Thus, the hypervolume becomes a four-dimensional representation of the scene.

Editing Pipeline



When interactively editing a video hypervolume, three different operators help the user to edit a scene with many degrees of freendom.

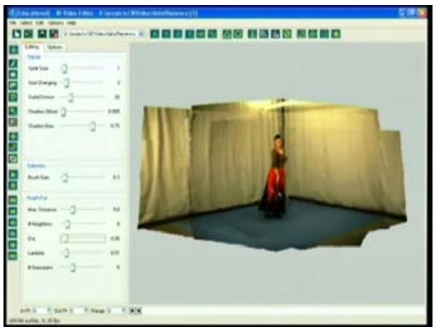
First, slicing the hypervolume results in a 3D volume not necessarily representing a 3D space but rather a 2D plane over time.

Then, the user can select a subvolume of the sliced part of the hypervolume. This intuitive approach is known from many editing programs for 2D images or 3D models.

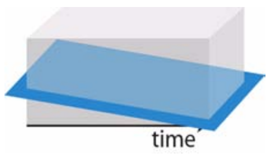
Finally, the editing operators allow the user to cut and paste, resize or reposition a previously selected part of the scene.

Slicing

- Visualize the hypervolume
- Navigate in space-time



- Intersect hypervolume with hyperplane

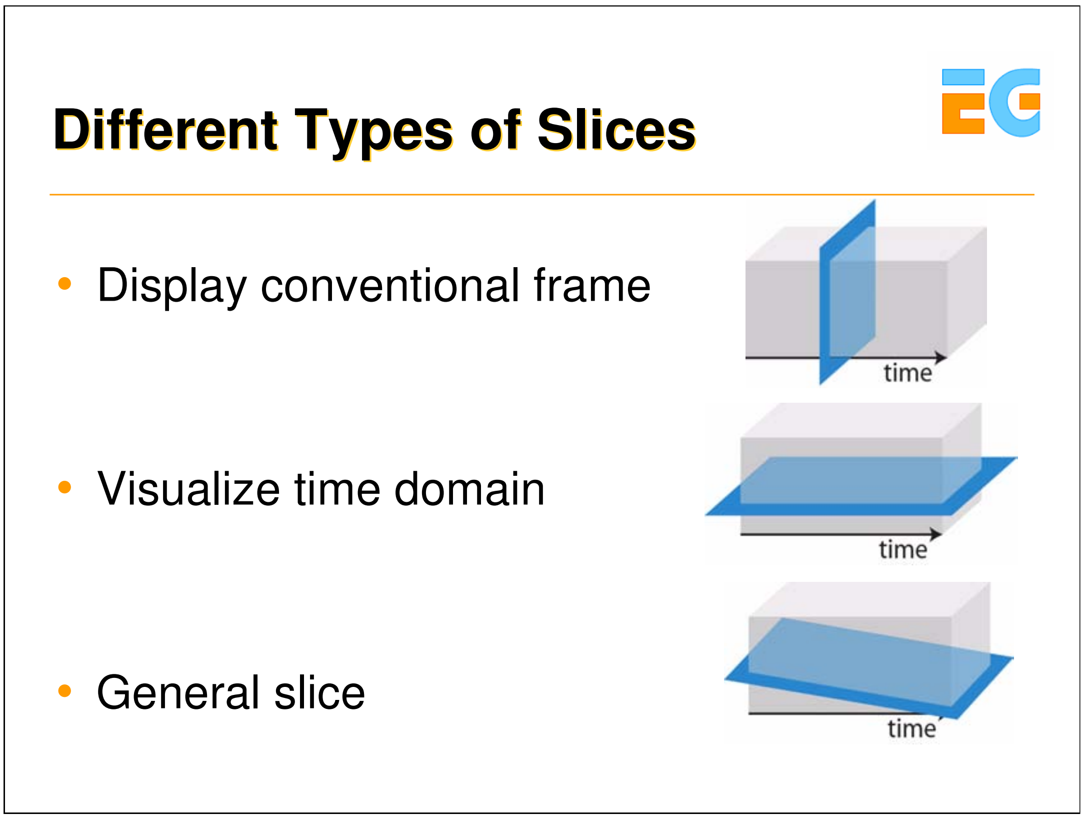


Via slicing, the user can define a part of the four-dimensional hypervolume which should be visualized on the screen.

The slicing process basically intersects the hypervolume with a hyperplane, reducing the dimensionality by one from 4D to 3D.

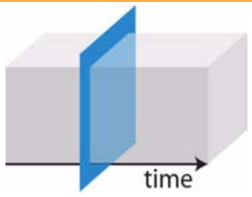
The resulting 3D point cloud is displayed on the screen and can be viewed from all sides using a trackball navigation interface.

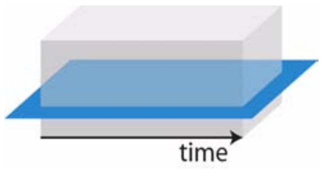
By freely choosing the slice orientation, the user can visualize both spatial and temporal aspects of the 3D video.

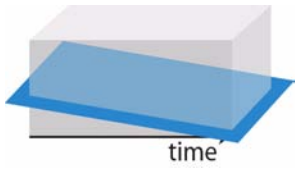


Different Types of Slices

- Display conventional frame
- Visualize time domain
- General slice







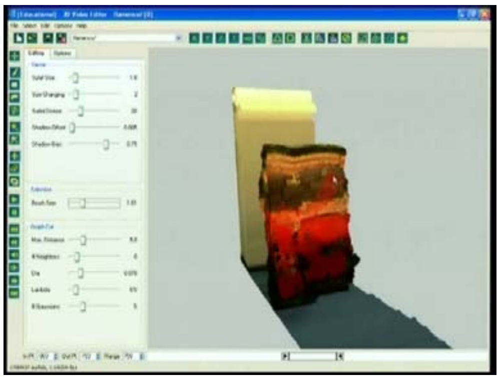
The simplest slice is oriented orthogonal to the time axis. This corresponds to viewing a conventional frame of the 3D video.

By orienting the slice parallel to the time axis, the user can define views which visualize the temporal domain of the video. This can be used for example to select a moving object over multiple frames.

In general, the user can define arbitrary slice orientations which can be used for example to follow object trajectories.

Selection

- Select points in current slice
- Selection tools
  - Bounding box
  - 3D paintbrush
  - Graph cut segmentation

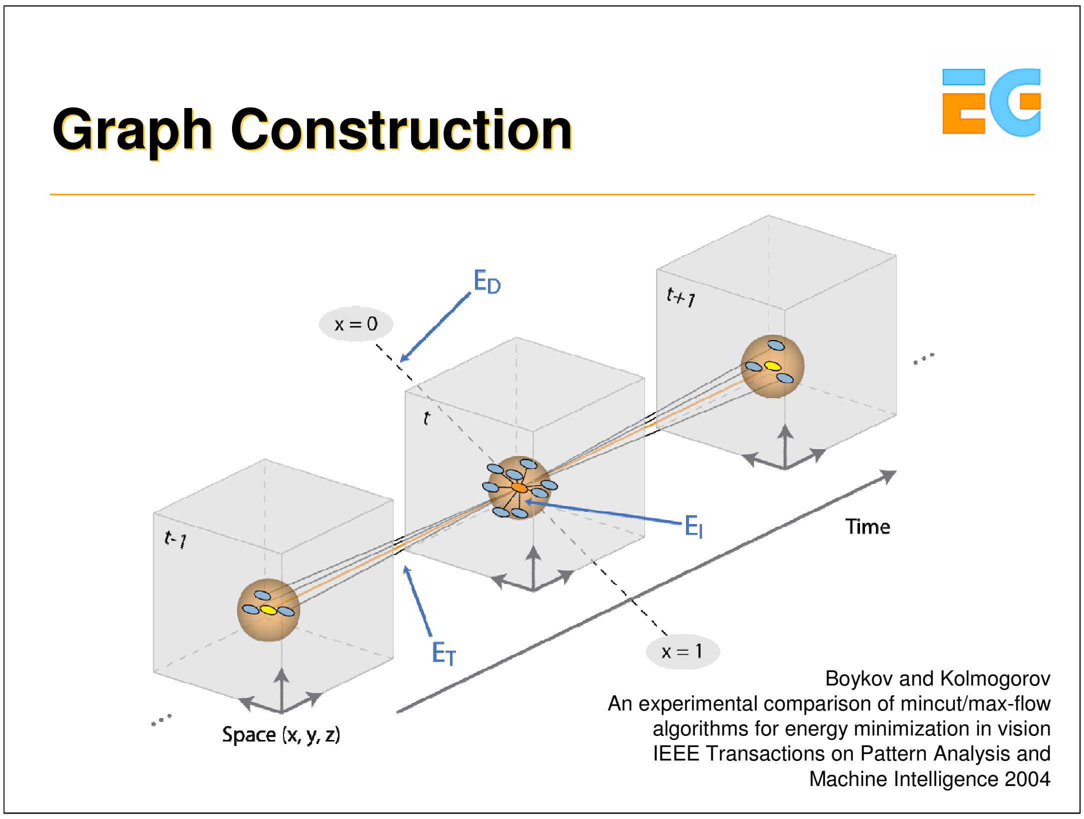

graph cut selection

In order to select parts of the current slice, the user can use different simple approaches. Each of these tools results in a partition of the points in the current slice.

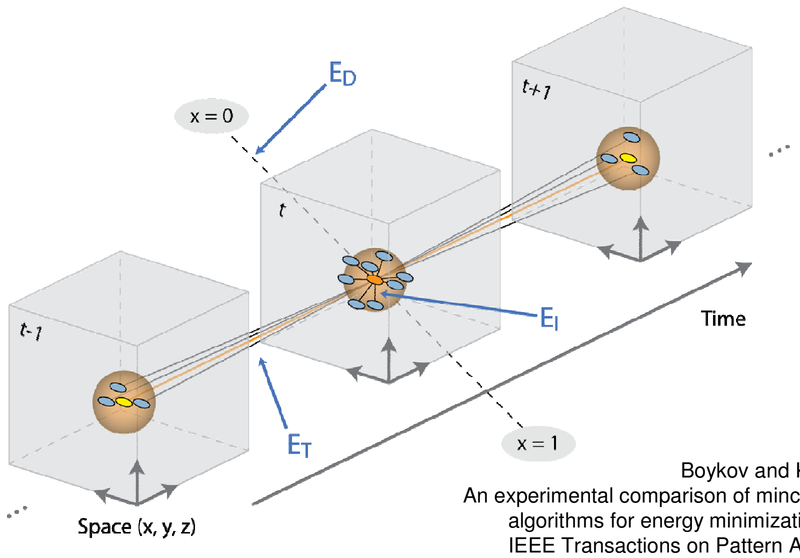
For example, a simple bounding box selects all points inside the volume.

With the 3D paintbrush, the user can select points by simply drawing onto the current 2D projection of the slice. Then, the paintbrush follows the surface of the scene.

Finally, a graph cut segmentation can be used to simply create a partition of the scene.



Graph Construction



Boykov and Kolmogorov
An experimental comparison of mincut/max-flow algorithms for energy minimization in vision
IEEE Transactions on Pattern Analysis and Machine Intelligence 2004

The underlying graph not only connects the points in each frame but also over time. This yields to smooth segmentation/selection results both spatially as well as temporally.

E(d) stores the so-called data energy which gives an indicator of how similar a point's information is to either the foreground or background (x=0 and x=1).

E(i) and E(t) are the so-called link energy (as in the cited literature).

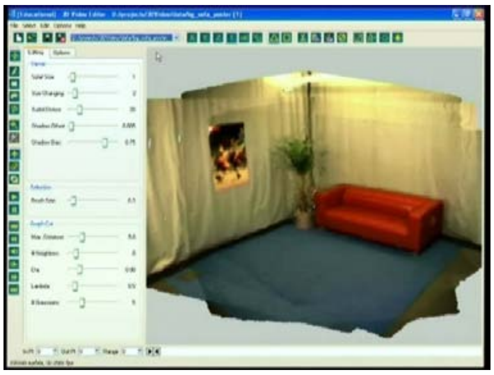
E(i) are the intra-frame connections and calculate the likelihood that two spatially close points belong to the same cluster (foreground or background). A k-nearest neighbor search yields the closest points.

E(t) are the inter-frame connections and calculate the likelihood that two spatially close points in different frames belong to the same cluster (foreground or background). A k-nearest neighbor search in the frames i-1 and i+1 starting from the reference point of frame i yields the closest points.

Besides color we can also exploit geometry information such as the normal similarity of two 3D points for 3D video editing.

Editing Operators

- Cut, copy & paste
  - Object removal
  - Insertion into other scenes
  - Actor cloning
- Translate, rotate, scale
  - Translation in temporal domain → time shift



For editing, our system supports basic operations like copy & paste, translation, or scaling and rotaton.

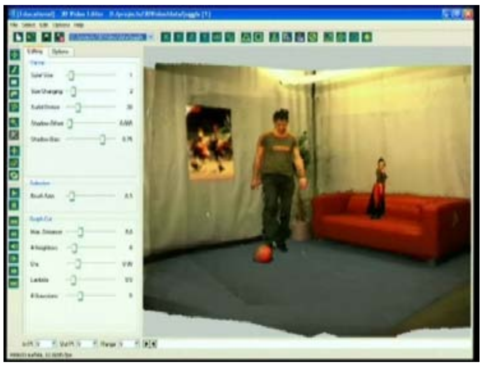
Although those operations are quite simple, they become very powerful in 3D video.

For example, with cut & paste together with our selection tools, you can generate complex effects like object removal or insertion of people into other scenes. After object removal, there are no remaining holes in the background, because we have the complete information of the scene available. Note how difficult such tasks would be with conventional 2D video.

The translation operator benefits from the generality of our hypervolume representation: a time shift operator is identical to the translation operator in the temporal domain.

More Editing Tools

- Shadow mapping
- Insertion of 3D meshes
- Insertion of 2D images and videos


shadow mapping

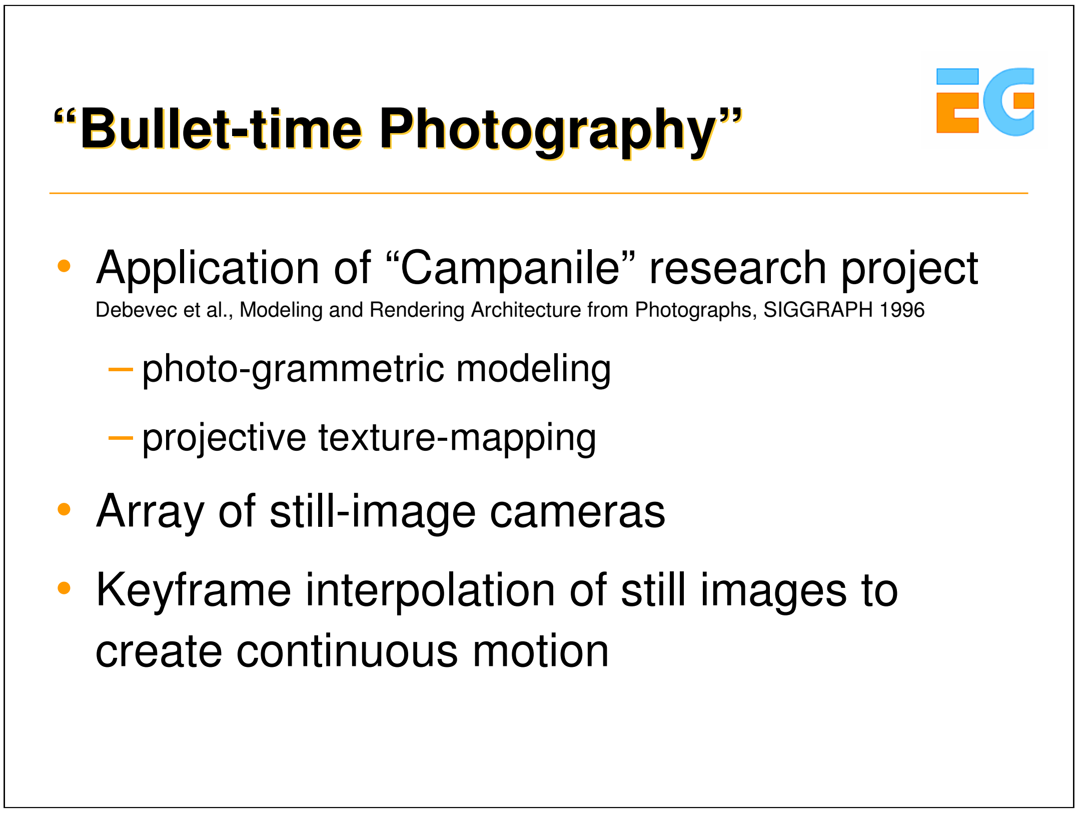
We implemented some further tools to insert 2D images or videos, which are just converted into point samples, and to generate artificial shadows.

The latter is important if novel objects get inserted into the scene because shadows provide important hints to the scene geometry.

3D Video Editing is Fun!

As a final result, let me show you a demo video which we generated with our editor…

You can clearly observe that we still have artifacts at object boundaries. This is a common weakness of the depth from stereo reconstruction algorithm which has difficulties to cope with depth discontinuities. We are continuously working on improving this issue. Let me mention that those artifacts are not an issue of an inaccurate graph cut segmentation, because you can observe them at the whole silhouette both in the source and the edited material. Graph cut has been only applied where the actor was connected to the floor.



“Bullet-time Photography”

- Application of “Campanile” research project
  Debevec et al., Modeling and Rendering Architecture from Photographs, SIGGRAPH 1996
  - photo-grammetric modeling
  - projective texture-mapping
- Array of still-image cameras
- Keyframe interpolation of still images to create continuous motion

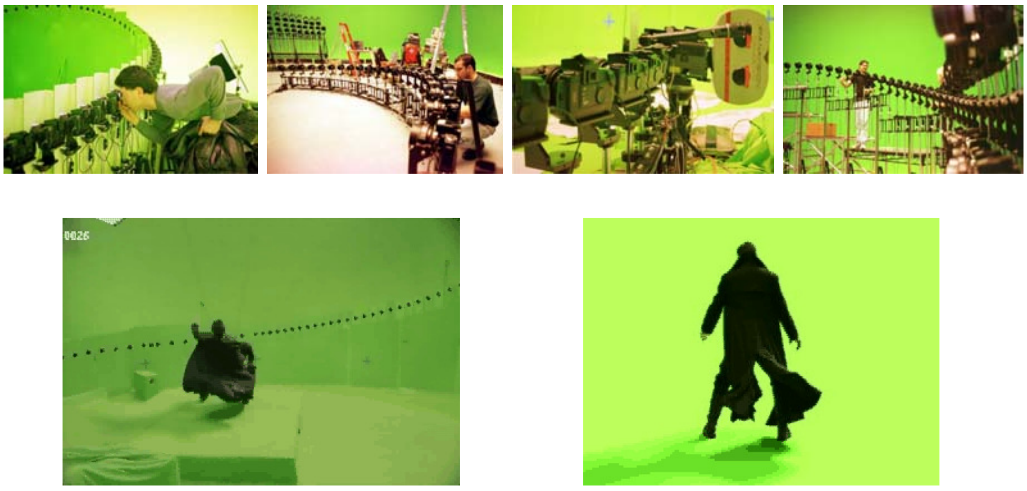
A 3D video application called “Bullet-time Photography” is well known under a differet term: The “Matrix effect”.

The underlying application is a result of Paul Devebec’s 1996 SIGGRAPH Paper.

An array of still image cameras simultaneously takes pictures of a scene from different viewpoints. The location and orientation of each of the cameras has to be determined before the actual shot since the images can’t be changed afterwards.

Using a keyframe interpolation of all these images, the editor can finally create a continuous motion around an object while it is freezed.

Bullet-time Photography in "The Matrix"

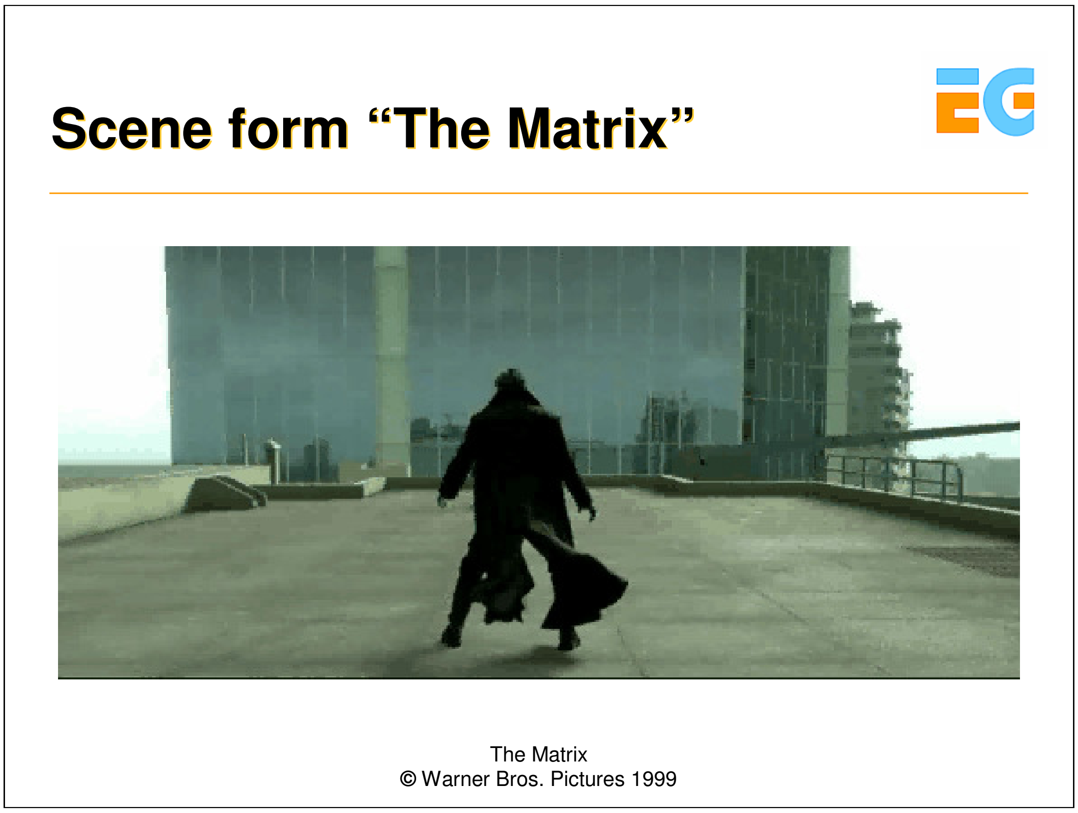


The Matrix
© Warner Bros. Pictures 1999

Here, we see how the matrix special effects were created.

The top row shows how many cameras were necessary and how precisely aligned the array of cameras must have been.

The bottom row shows two intermediate results

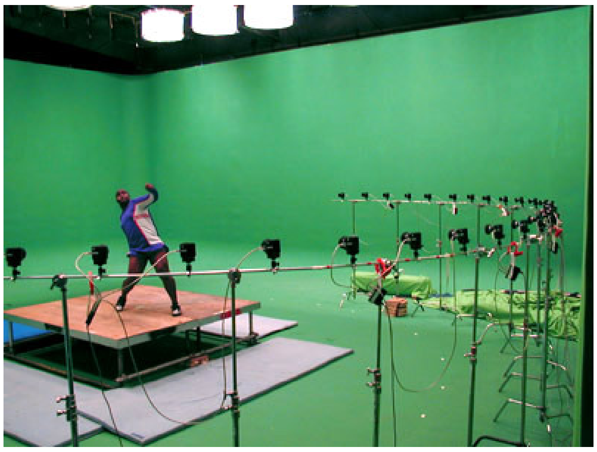


Scene form “The Matrix”

The Matrix
© Warner Bros. Pictures 1999

And this is the final scene. The captured person is placed into a synthetic environment.

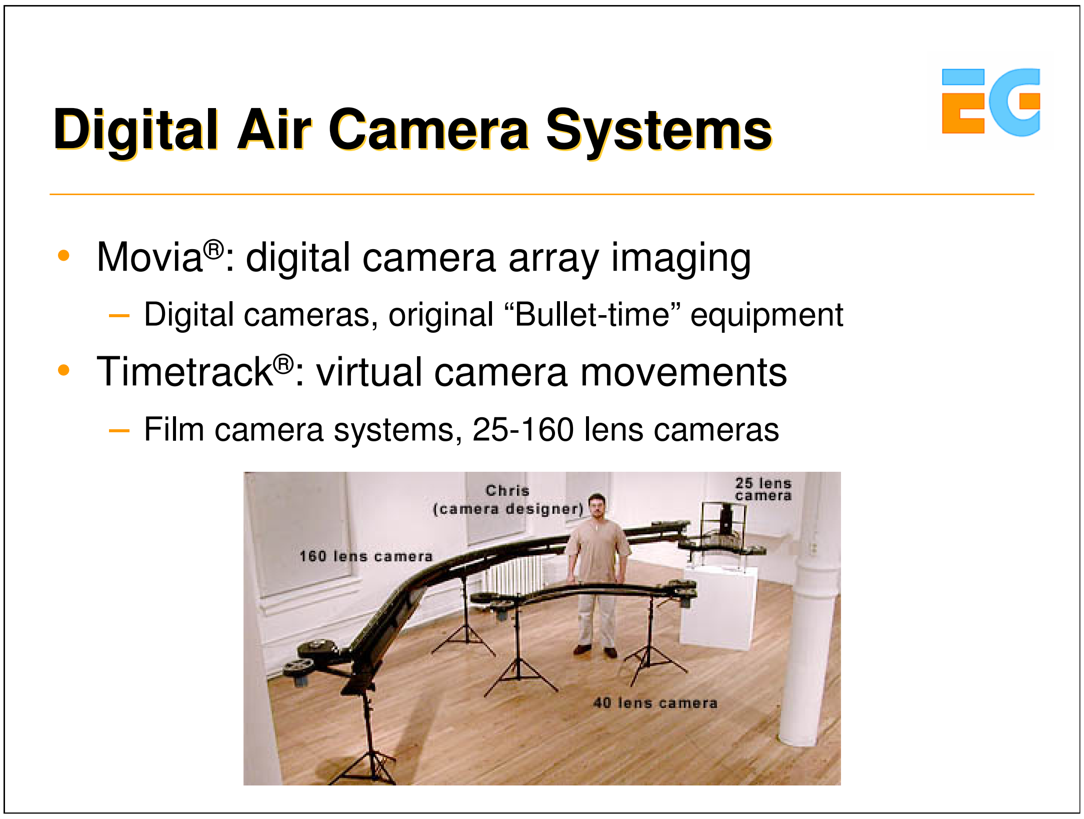
"Bullet-time Photography" Revisited



Movia® camera systems, www.movia.com
Digital Air (Dayton Taylor), www.digitalair.com
Geneva, Switzerland

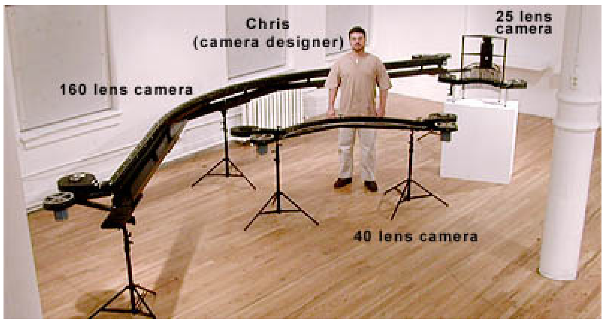
The bullet-time approach of the matrix has been adapted to video cameras by Digital Air in Switzerland. Using an array of many well-aligned video cameras, it is not only possible to fly around a freezed object but also to change the speed and direction of time.

These special effects are used in cinematography, commercial advertisements or trailers as eye-catchers.



Digital Air Camera Systems

- Movia®: digital camera array imaging
  - Digital cameras, original “Bullet-time” equipment
- Timetrack®: virtual camera movements
  - Film camera systems, 25-160 lens cameras



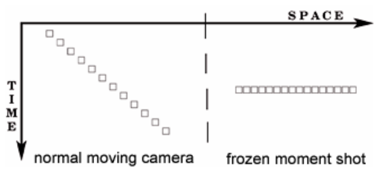
Digital Air offers two different products.

Movia relies on normal still image cameras aligned in an array as we have seen before.

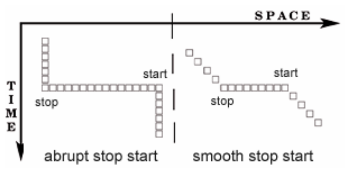
Timetrack replaces the still image cameras with film camera systems where each camera records the scene through 25-160 lenses aligned in an array.

Effects (www.timetrack.com)

- Frozen Moment



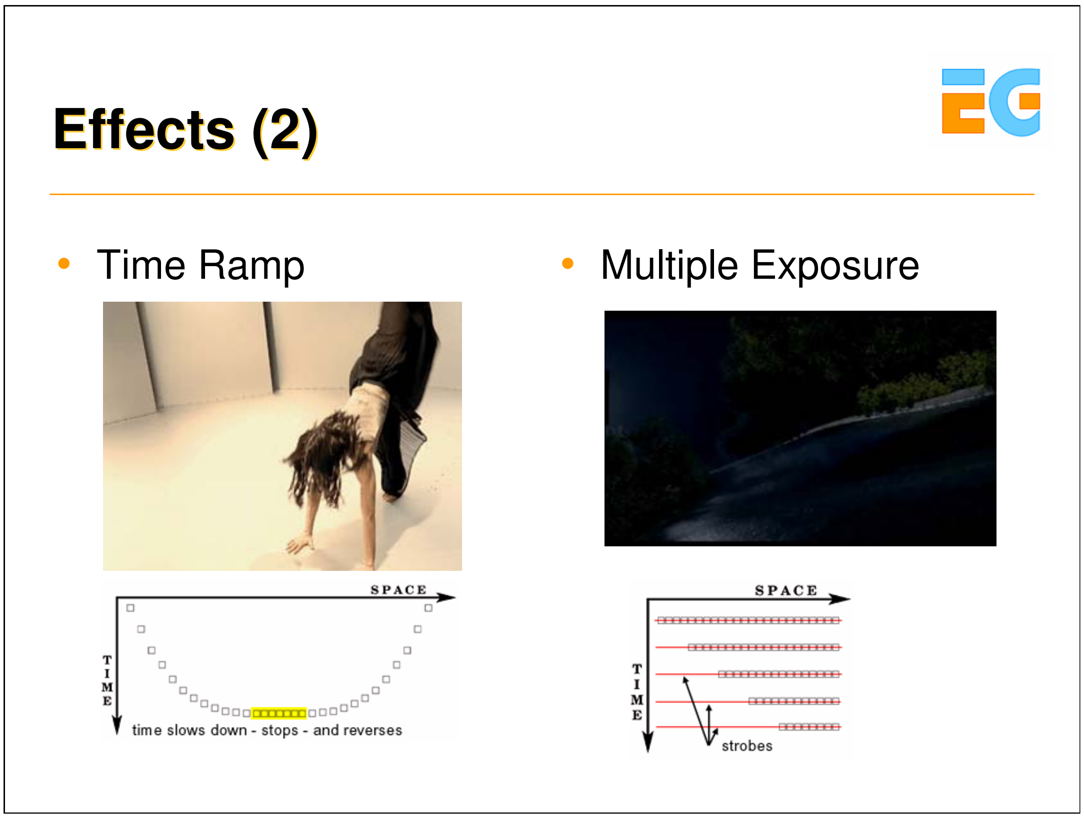
- Stop-Start



As mentioned before, using the timetrack system allows to change the timely behavior of the result, too. The diagrams below the movies depict the editing of the input imagery.

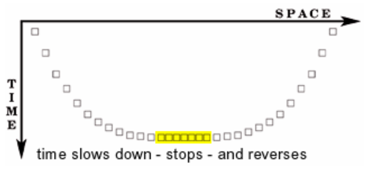
On the left, we see the classic approach where the camera moves in space and over time and the frozen moment where the camera moves in space in a freezed frame.

On the right, two combinations of these approaches are shown. The camera stands still and shows a moving scene, then freezes abruptly or smoothly and rotates around the scene before finally standing still again and showing the remaining sequence from a different location.

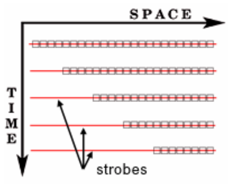


Effects (2)

- Time Ramp

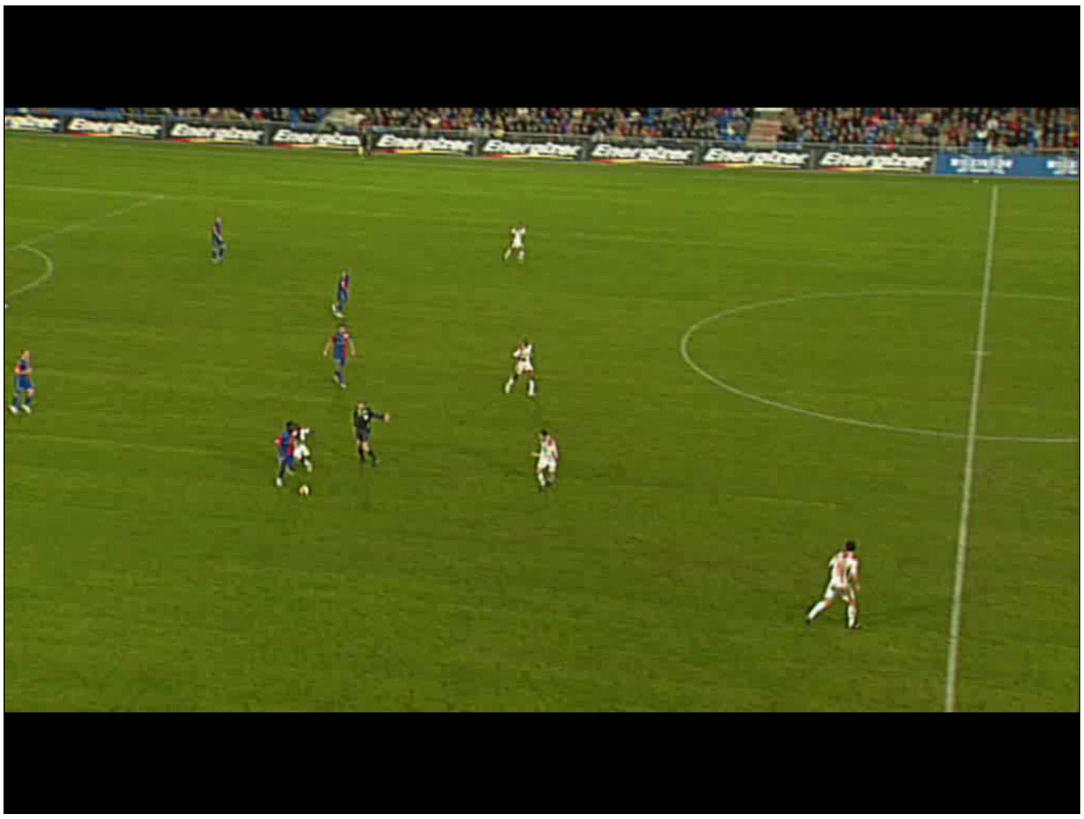


- Multiple Exposure



Another possible special effect is the „time ramp“ where a continuous motion of the camera shows a scene deccelerating, stopping, and reverse accelerating again.

The „multiple exposure“ effect overlays multiple shots of the same scene and creates a stronger feeling of the motion.



A scene of a soccer game with an really tight offside situation. None of the existing cameras can resolve the situation

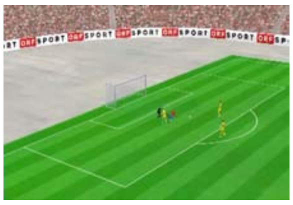
3D Sports Visualization

Orad *VirtualLive*

PVI *EyeVision*

Red Bee Media *Piero*

LifeInMedia *LifeInSports*


Swiss Timing *tvVAT[3D]*

BBC R&D *iview*

Different approaches have been developed to visualize sports scenes from novel views not covered by a camera.

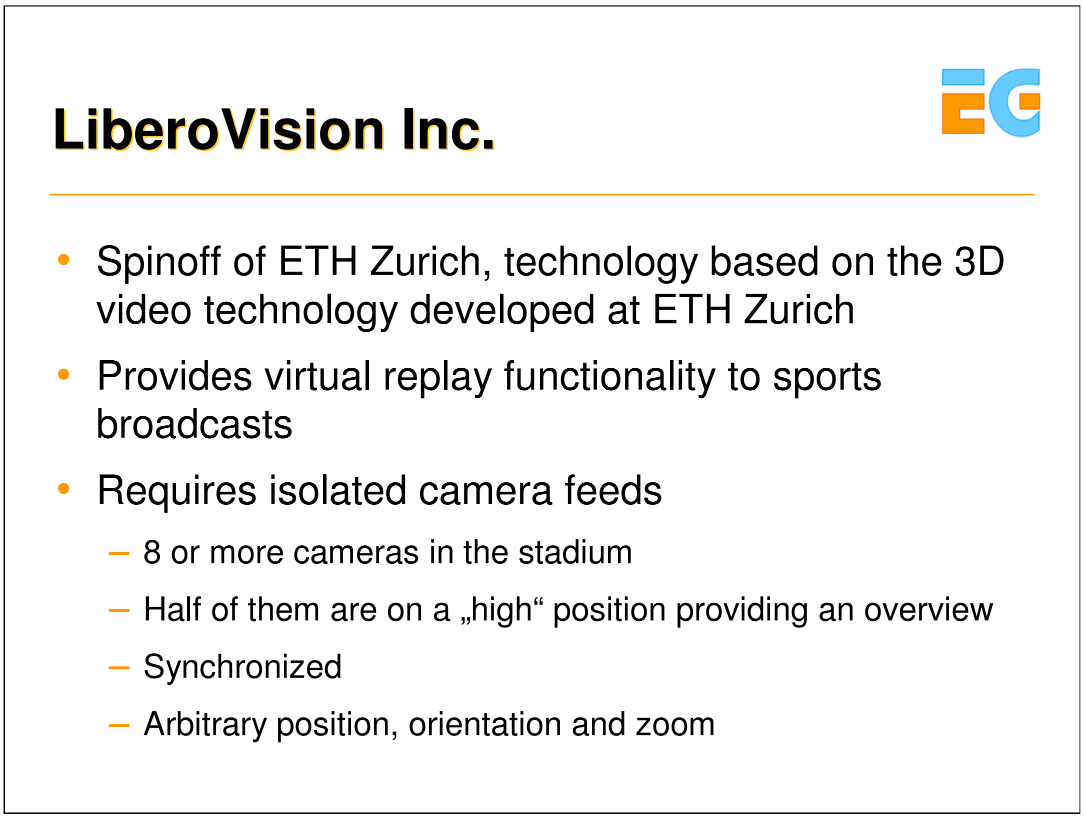
VirtualLive and LifeInSports are based on a 3D model which is syntheticaly rendered, thus, allowing a total freedom of possible viewpoints but a non-realistic looking result.

EyeVision captures a scene with over 30 cameras positioned around the pitch. These cameras are controlled by an operator and are always focused on one spot on the pitch. The final result is a blending through the camera views resulting in a fly-around of the scene.

Piero and iview generate a model of the players based on the original camera feeds. These representations are then placed into a synthetic environment offering more freedom in terms of possible viewpoints.

LiberoVision is one step beyond the previously seen examples by offering a full freedom of viewpoints resulting in a realistic virtual view of the scene.

The video shows the same scene as before. In the moment of the offside decision, the playback stops and the virtual camera moves onto the offside line and shows the view of the referee on the side-line.



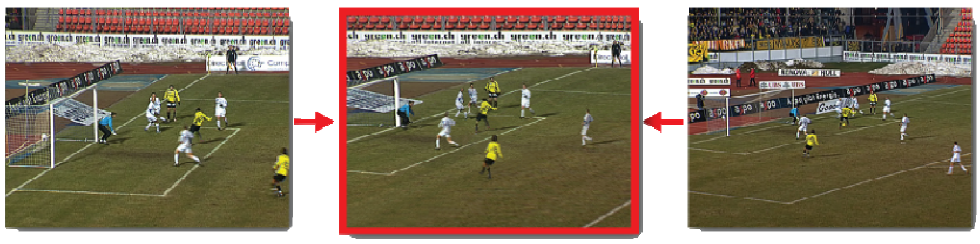
LiberoVision Inc.

- Spinoff of ETH Zurich, technology based on the 3D video technology developed at ETH Zurich
- Provides virtual replay functionality to sports broadcasts
- Requires isolated camera feeds
  - 8 or more cameras in the stadium
  - Half of them are on a „high“ position providing an overview
  - Synchronized
  - Arbitrary position, orientation and zoom

The technology of LiberoVision is based on the developments at ETH Zurich, Switzerland.

It requires the isolated feeds of the cameras positioned in the stadium and does not require any additional infrastructure inside the stadium. Usually, more than 8 cameras are used to produce a sports event such as a soccer game, approximately half of them are on a high position and provide an onverview of the game. Synchronization of the cameras is a requirement of the broadcast production and makes it easier to recreate a scene. However, the cameras are not fixed but can change their position, orientation and zoom.

Based on existing camera feeds only



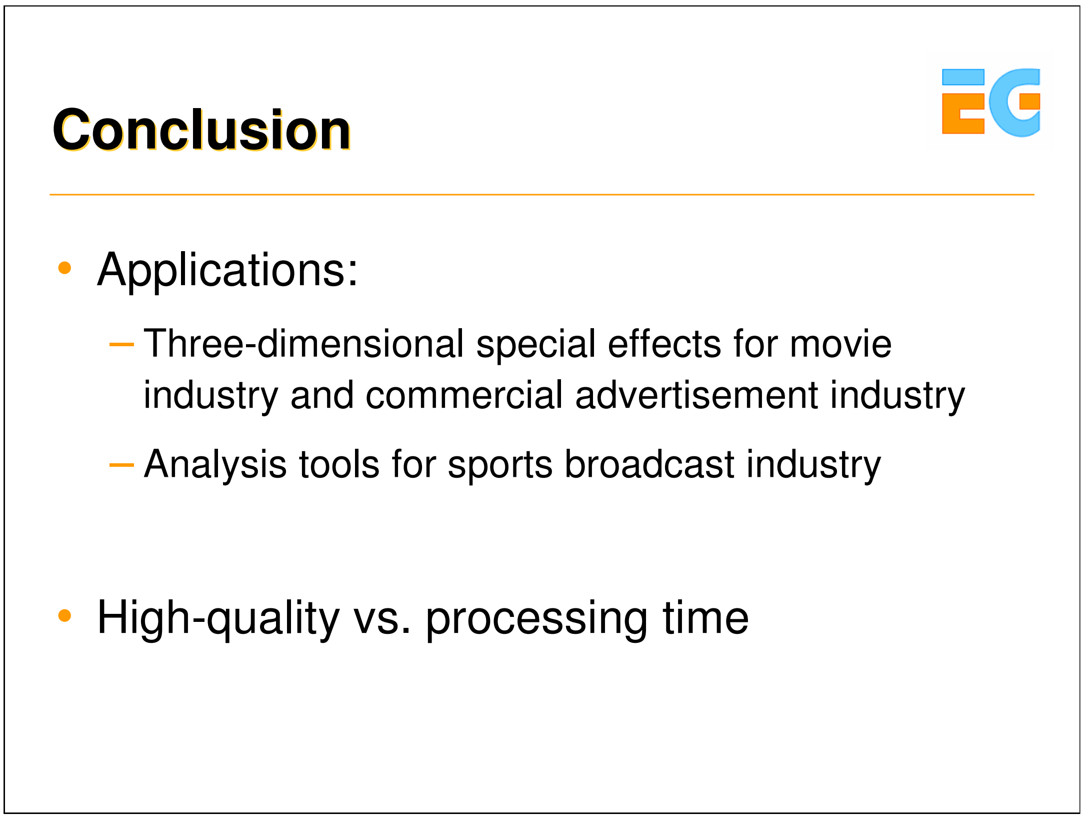
Penalty box camera (left)

LiberoVision (virtual camera)

Lead camera (middle line)

On the left, we see a shot of the penalty box camera and on the right, a shot from the lead camera in the middle of the pitch. Based only on both these images, we can create the virtual image inbetween showing the scene from a viewpoint not covered by the physical cameras.

Another example of the LiberoVision technology. A goal situation is shown and the virtual camera provides a birds-eye view to tactically analyze the scene.



Conclusion

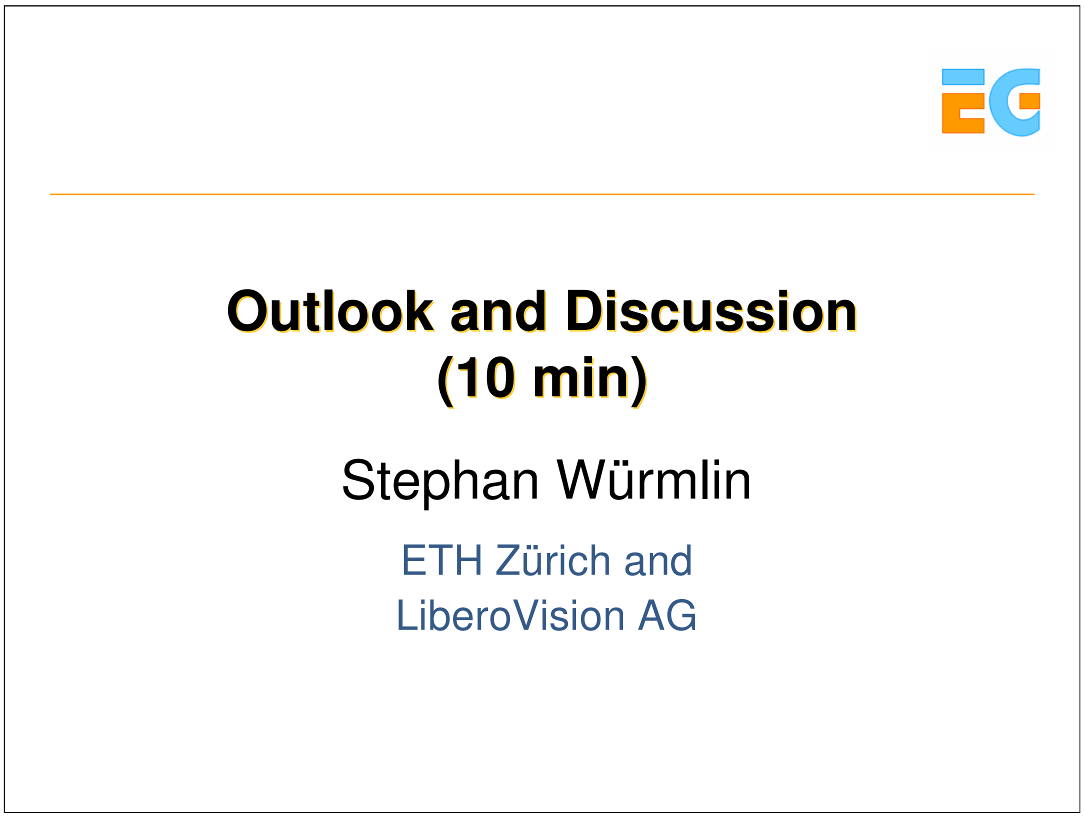
- Applications:
  - Three-dimensional special effects for movie industry and commercial advertisement industry
  - Analysis tools for sports broadcast industry
- High-quality vs. processing time

We have shown two different applications of 3D video.

First, the three-dimensional special effects for the movie industry such as the "Matrix effect" and derivatives of the same technology which is used in the commercial advertisement industry as eye-catchers.

A second application are tools for the analysis of sports scenes where virtual views are generated providing perspectives not captured by the available cameras.

Both these industries have different requirements regarding the results. While the movie industry can afford a rather long processing time, the result must be of a really high quality. In sports broadcast on the other hand, the result must be available within minutes or even seconds.

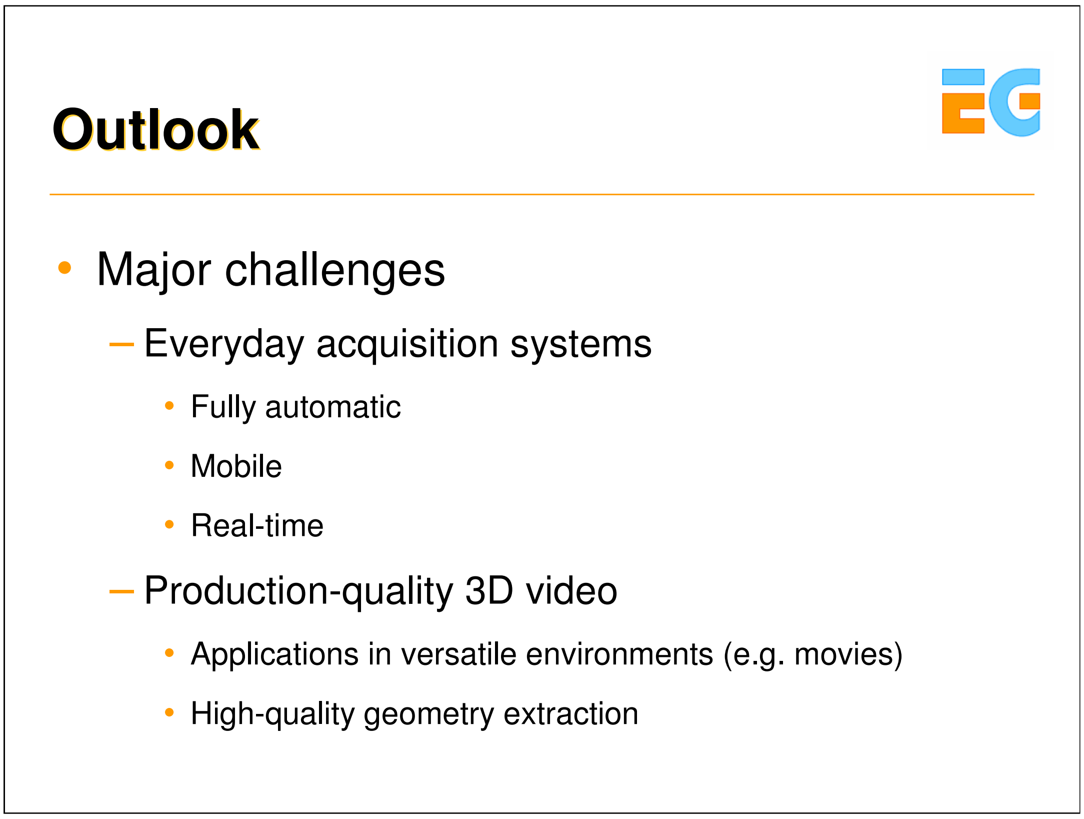


Outlook and Discussion (10 min)

Stephan Würmlin

ETH Zürich and
LiberoVision AG

478

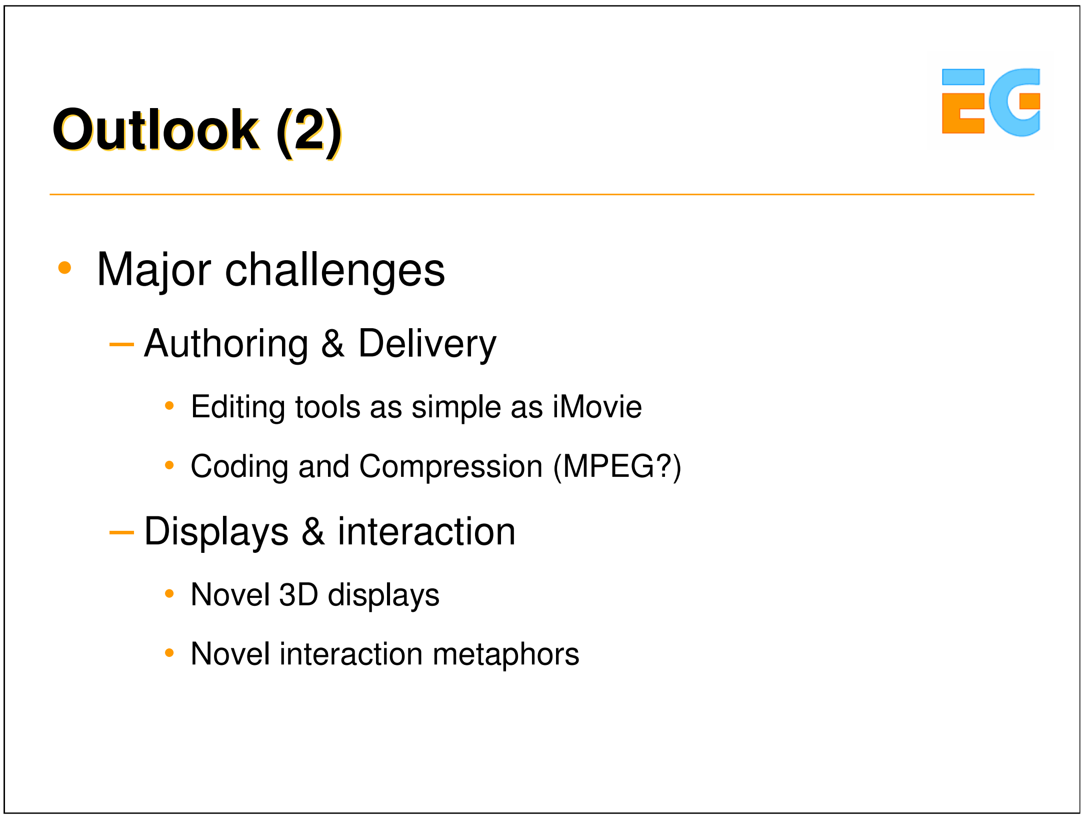


Outlook

- Major challenges
  - Everyday acquisition systems
    - Fully automatic
    - Mobile
    - Real-time
  - Production-quality 3D video
    - Applications in versatile environments (e.g. movies)
    - High-quality geometry extraction

Acquisition and processing of 3D video is still very time consuming, cumbersome and needs years of training and experience. However, for reaching the masses, acquisition needs to be as simple as possible, meaning fully automatic, mobile installations (such as on a mobile equipped with a camera and a GPS) and should at least acquire and process the data in real-time.

Despite the just mentioned commercial applications of 3D video, production-quality 3D video can only be achieved by either focusing on one application (such as soccer or sports in general) or by using special equipment, such as highly sophisticated camera systems and the like. The toughest problem is the extraction of high-quality geometry information out of the images alone. This is probably the biggest challenge of it all!



Outlook (2)

- Major challenges
  - Authoring & Delivery
    - Editing tools as simple as iMovie
    - Coding and Compression (MPEG?)
  - Displays & interaction
    - Novel 3D displays
    - Novel interaction metaphors

Major challenges also include authoring and delivery of 3D video data.

1) On the application side, 3D video gives us novel possibilities for creating special effects but also requires novel editing operations. It is desirable to have an editing software being as user-friendly as today's 2d video editing programs, such as iMovie.

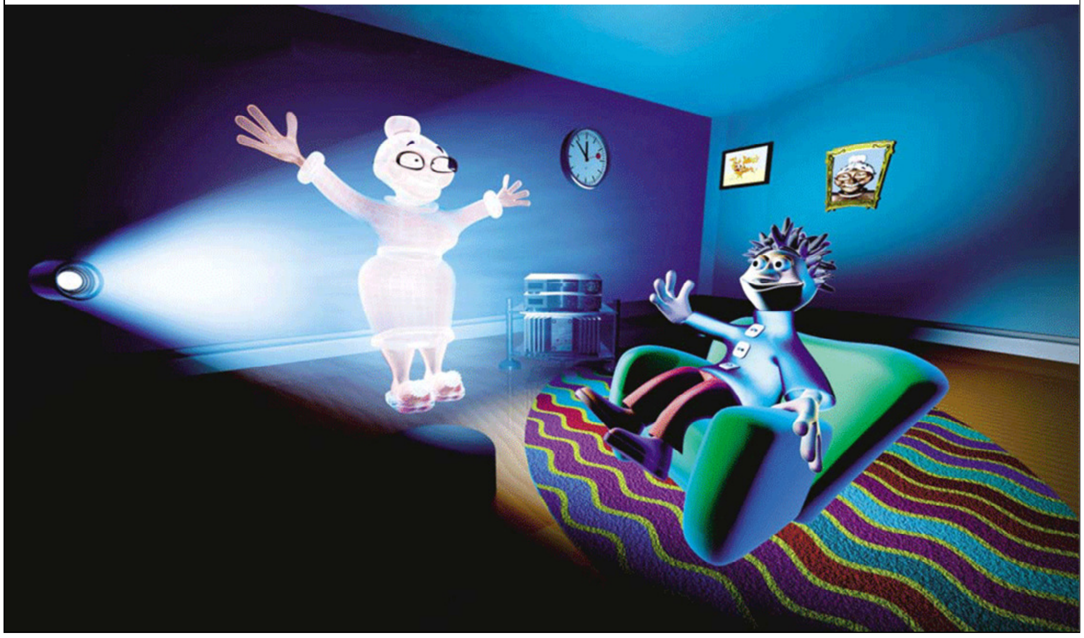
2) To bring 3d video to the masses there is also the interesting question of spatio-temporal coding of the data stream. Those issues are currently investigated by the MPEG committee.

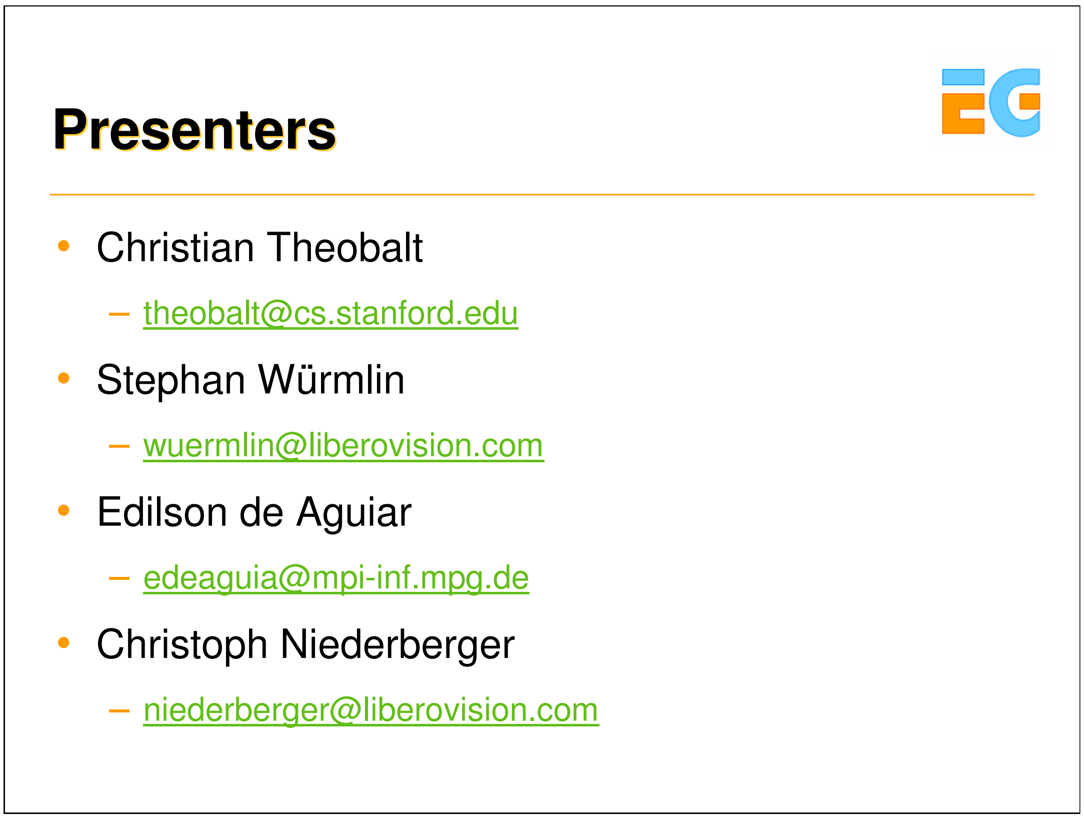
Current display technology does not fully exploit the interactive and three-dimensional nature of 3D video.

1) Hence, novel displays need to be developed such as autostereoscopic TVs or mobile projectors

2) Furthermore, interaction with the mouse is cumbersome in most daily environments, novel user interfaces such overcome this limitation. Microsoft Surface is one possible way of solving these problems.

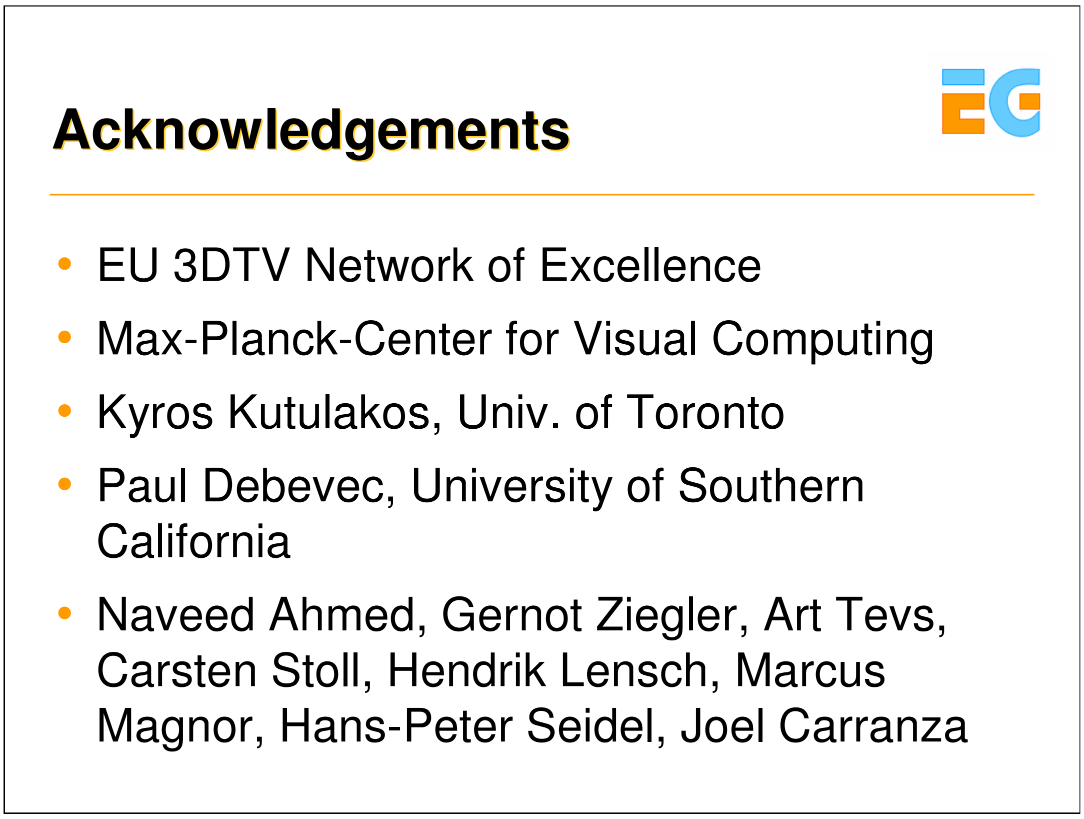
Questions?





Presenters

- Christian Theobalt
  - theobalt@cs.stanford.edu
- Stephan Würmlin
  - wuermlin@liberovision.com
- Edilson de Aguiar
  - edeaguia@mpi-inf.mpg.de
- Christoph Niederberger
  - niederberger@liberovision.com



Acknowledgements

- EU 3DTV Network of Excellence
- Max-Planck-Center for Visual Computing
- Kyros Kutulakos, Univ. of Toronto
- Paul Debevec, University of Southern California
- Naveed Ahmed, Gernot Ziegler, Art Tevs, Carsten Stoll, Hendrik Lensch, Marcus Magnor, Hans-Peter Seidel, Joel Carranza

Acknowledgements

- E. Lamboray, M. Waschbüsch, D. Cotting, F. Sadlo, M. Gross
- M. Zwicker, C. Lee, W. Matusik, H. Pfister
- T. Weyrich, N. Kern, P. Kaufmann, S. Böhler, R. Küng, A. Smolic, D. Y. Kwon, S. Rondinelli
- C. Niederberger, R. Keiser, M. Germann, M. Feriencik
- ETH research grant No. 0-21020-04 (blue-c-II poly-project)