

Video-based Rendering

Marcus Magnor
MPI Informatik
Saarbrücken, Germany

Tutorial
3DPVT 2004



The Ultimate Goal



Scene from "The Matrix", © Warner Bros. 2000

Behind the Scenes ...



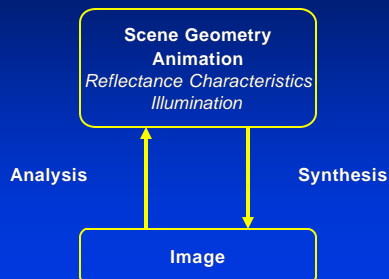
<http://whatisthematrix.warnerbros.com/>

Photography vs. Computer Graphics



- Easy acquisition
- Fast display
- Natural appearance
- No flexibility
- Unlimited flexibility
- Time-consuming modeling
- Expensive image synthesis
- Non-realistic appearance

Image-based Modeling & Rendering



Real-World Acquisition

	<i>visual appearance</i>	<i>3D shape</i>
<i>static object</i>	<i>photo</i>	<i>active scanning</i>
<i>dynamic scene</i>	<i>video</i>	<i>?</i>

Video-based Rendering

Goals

- Render dynamic, real-world object from arbitrary viewpoint
 - Real-time display frame rate
 - Online processing: live display, or
 - Offline: best-possible rendering quality

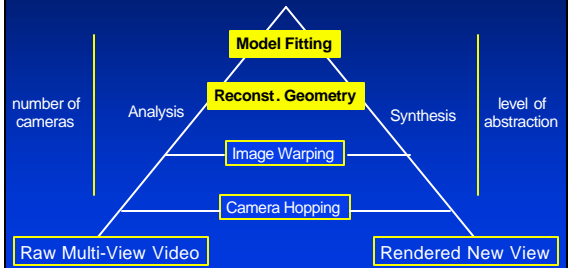
Means

- Reconstruction of dynamic 3D shape from images
- Fast, realistic rendering from shape + video

Scenario

- Acquisition: handful of synchronized video cameras
- Display: consumer-market PC hardware

Video-based Scene Analysis & Synthesis



Tutorial Outline

Multi-Video Acquisition

On-line Processing

Model-based Analysis and Synthesis

Outlook & Discussion

Visual Hull Rendering

Voxel-based

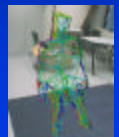
- Discretize volume into voxels
- Discard voxels outside silhouette cones

Rapid octree construction from image sequences [Szeliski 1993]

Polyhedron-based

- Generate boundary representations of silhouette cones from silhouette polygons
- Intersect 3D polyhedral cones

Polyhedral visual hulls for real-time rendering [Matusik et al. 2001]



Visual Hull Rendering

Implicit reconstruction

- Render desired view directly from reference views
- Reconstruction implicitly performed during rendering

Image-based visual hulls [Matusik et al. 2000]

On graphics hardware

- Reconstruction and rendering in one single pass

Hardware-accelerated visual hull reconstruction and rendering [Li et al. 2003]

Multi-View Video Acquisition

Studio

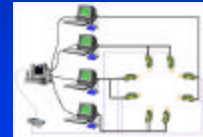
- Stage size 2x2x3 meters
- Calibrated cameras

Acquisition

- 8 Sony DFW-500 cameras
- 320x240 RGB, 15 fps
- External trigger
- IEEE1394 downlink

Processing

- Clients: four 1GHz Athlon PCs
 - 2 cameras / client
- Server: 1.7GHz P4 PC



Volumetric Visual Hull Reconstruction

MPI INFORMATIK

Camera Camera Camera Camera Sony DFV-V500

Client 1 Client 2

Background subtraction Partial volume

RLE encoded volume RLE encoded volume

Server

Complete volume reconstruction 1.7 GHz P4

Voxel visibility computation

Video-based Rendering 3DPV/T04 Tutorial

Background Subtraction

MPI INFORMATIK

- For each background pixel compute mean color $\mathbf{m}(x, y)$ and color standard deviation $S(x, y)$
- New pixel $p(x, y)$ classified by 3 criteria
 - Surely foreground : $\|p(x, y) - \mathbf{m}(x, y)\| > T_n \cdot S(x, y)$
 - Surely background : $\|p(x, y) - \mathbf{m}(x, y)\| < T_b \cdot S(x, y)$
 - Else shadow test : Hue difference $\Delta > T_{angular} \Rightarrow$ foreground, else background

Cheung et al. 2000

Hole Filling : Dilate / Erode

Video-based Rendering 3DPV/T04 Tutorial

Volume Reconstruction

MPI INFORMATIK

Shape-from-silhouette

- Scene : Subdivided into cubic volume elements
- Silhouette defines generalized cone
- Intersection of cones builds voxel-based approximation to *visual hull*
- On Client : Re-project voxels into silhouette view (precomputed)
- Each client computes partial hull for 2 views
- RLE encode partial hulls
- Complete hull on server by intersection

Video-based Rendering 3DPV/T04 Tutorial

Billboard Rendering

MPI INFORMATIK

[Yamasazi et al., Eurographics 2002]

- Small rectangle parallel to the image plane of the novel viewpoint
- Centered in grid cell
- Size chosen so that it completely covers the cell
- Performed entirely on graphics card

Video-based Rendering 3DPV/T04 Tutorial

Billboard Texturing

MPI INFORMATIK

- Texture patches from source views are back-projected onto the billboard
- Blend contributions from best matching cameras which "see" the cell

Video-based Rendering 3DPV/T04 Tutorial

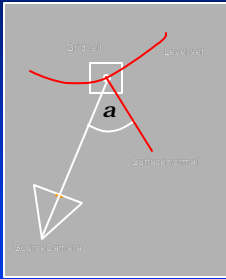
Visibility

MPI INFORMATIK

- A camera is not used for texturing if the textured cell is occluded in the camera's view by other occupied cells

Video-based Rendering 3DPV/T04 Tutorial

Selecting Source Cameras



- Select cameras in which the voxel is visible
- Compute angle a between the line of sight of source camera and normal
- Blending weight for each of the cameras is

$$w := \exp(-ca)$$

Real-Time Implementation

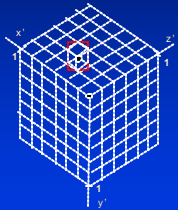
Balance CPU \leftrightarrow Vertex Pr. \leftrightarrow Pixel Pr.

CPU

- For cell bordering the surface:
 - Compute two texturing source cameras and weights
 - Send (x,y,z) coordinate of current cell to graphics hardware
 - Invoke display list to render a unit square
- Data transfer to graphics hardware is minimized

Real-Time Implementation

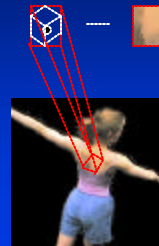
GPU tasks I: Vertex Processing



- Rotate unit square to face viewer and move to cells (x,y,z) position
- Project into novel view
- Project into source views to assign texture coordinates

Real-Time Implementation

GPU tasks II: Pixel Processing



- Use multiple texture units to blend contributions from different source cameras
- Last texture unit: Level set function as 3D alpha texture



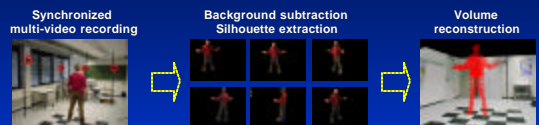
Volumetric Reconstruction & Rendering



Hardware accelerated rendering: 30 fps

- Input: 6 x 320x240 RGB, 15 fps, 2GHz P4, GeForce4
- Reconstructed visual hull: 64x64x64 voxel resolution

Volumetric VH Reconstruction: Processing Pipeline



Multi-Video Recording: 15 fps

- 6 cameras, 320x240 RGB

Silhouette extraction: >15 fps

Volume reconstruction: >15 fps

- 32x32x32 voxels

Visibility computation: 5 fps

- Non-optimized

Results

Complexity

- Rendering setup: Pre-process and transfer source camera images to graphics hardware, linear in number of views
- Rendering time approximately linear in number of surface cells

Frame-rates

- Pentium IV 1.8GHz, GeForce 4 Ti 4600
- 8 source cameras => 2.4 MByte image data
- 30 fps at a grid resolution of 64 x 64 x 64

Quality Improvements



Problem

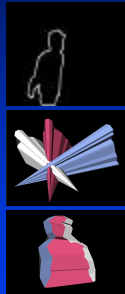
- Segmentation errors
- Seams along boundary lines between two textures

Partial Solution

- Render all background pixels transparent
- Add some transparency along the foreground boundary of the textures

The Visual Hull

- Approximate shape representation
 - Envelope of true geometry
- Reconstructed from multiple silhouettes
 - Silhouette projection into 3D space: silhouette cones
 - Polygonal approximation
 - Intersection of silhouette cones: Visual Hull
- Fast acquisition and visualization of arbitrary foreground geometry

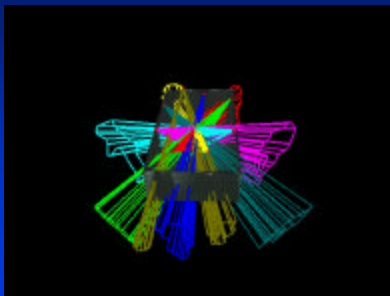


Polygonal Visual Hull Reconstruction

- Silhouette contours approximated as 2D polygons
 - Precision adjustment: variable # polygon vertices
- Contours define polyhedral 3D cones
- Intersection of cones: Polygonal Visual Hull
- On Client
 - silhouette segmentation
 - polygonal contour approximation
 - Encode contour polygon
- On Server
 - 3D polygonal cone intersection



Visual Hull Reconstruction

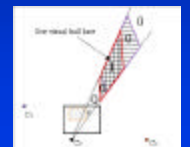
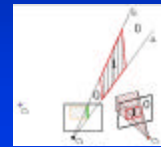


Hardware-Accelerated Visual Hull Reconstruction

[Li et al., Graphics Interface 2003]

- Perform 3D intersections on rasterized silhouette face
- Silhouette mask encoded in texture alpha channel
- Project all textures onto silhouette face
- Exploit multiple texture units

$$A = \prod_{k=1}^N A_k$$



Hardware-Accelerated Visual Hull Rendering



View-dependent texture mapping

- Weight of view k for face f
 - Per-face visibility evaluation
 - View-dependent weight

$$W_{k,f} = V_{k,f} \cdot W_k$$

$$V_{k,f} = \begin{cases} 1, & \mathbf{v} \cdot \mathbf{n} < 0 \\ 0, & \text{otherwise} \end{cases}$$

$$W_k = 1 / \cos^{-1}(q_{k,c})$$

- Weight normalization

$$\hat{W}_{k,f} = W_{k,f} / \sum_{i=1}^N W_{k,f}$$

Per-fragment color computation

- On graphics hardware

$$C_f = \sum_{k=1}^N \hat{W}_{k,f} \cdot T_k$$

C_f : output fragment color for face f , T_k : color texel value from view k

Results



- Acquisition: 8 cameras, 15 fps, 320x240 pixels
- Rendering: 640x480 pixels, 40 fps

Experiments & Performances



- Rendering window size : 640 x 480 pixels
- Each silhouette cone: 100-120 silhouette faces

Graphics hardware	GeForce 3	Radeon 9700 Pro
Reference view number	4	8
Frame rates(fps)	84	44

Previous fastest system performance for 4 views : 30 fps [Matusik et al. 2001]

Enhanced VH Rendering



Projective Texture Mapping

- Problem: texturing-through
- Solution: Shadow Mapping**
 - Render from camera's point-of-view
 - Generate depth map
 - Render from eye's point-of-view
 - For each fragment, determine 3D distance to camera
 - Compare depths
 - Assign 0/1 to (in)visible fragments



Enhanced VH Rendering



Opaque Projective Texture Mapping

- Multiple textures
 - Tex0: Color texel, Tex1: Shadow texel
 - Final fragment: Tex0*Tex1
- Enable alpha testing to remove occluded parts

Problem

- Shadow leak due to insufficient depth map resolution



Enhanced VH Rendering



Extended Opaque Projective Texture Mapping

- Higher depth map resolution
 - Increased rendering time
 - More texture memory
- Render "fatter" version of depth map
 - 2-pass depth map rendering
 - 1st Pass: Line mode using thick line width
 - 2nd Pass: Normal mode



Polygonal VH Reconstruction: Processing Pipeline

MPI
INFORMATIK

Synchronized multi-video recording

Background subtraction
Silhouette extraction

Visual Hull reconstruction

Silhouette extraction: >15 fps

- 320x240 RGB

Polygonal VH reconstruction: 15 fps

- 400-500 triangles
- 6 cameras

Video-based Rendering 3DPVT04 Tutorial

Results

MPI
INFORMATIK

Video-based Rendering 3DPVT04 Tutorial

Model-based Free-Viewpoint Video

MPI
INFORMATIK

Multi-view video recording

Silhouette extraction

Generic model adaptation

Multi-view texturing
Interactive rendering

Silhouette-based model-fitting
[Carranza et al., Siggraph 2003]

Video-based Rendering 3DPVT04 Tutorial

Human Body Model

MPI
INFORMATIK

Generic body model

- 16 body segments
- 21422 triangles total

Initialization

- Adaptation to actor's proportions
- 16 non-uniform deformation parameters per segment

Motion capture

- 15 joints
- 35 pose parameters
- Special limb parameterization

Video-based Rendering 3DPVT04 Tutorial

Energy Function

MPI
INFORMATIK

Scaling & Pose Parameter Estimation

- Criterion: overlap between image silhouettes and body model projection
 - Area of intersection
 - Robust against silhouette inaccuracies
- Graphics hardware acceleration
 - Model rendering
 - Pixel-wise XOR (stencil buffer)
 - 8 silhouettes / frame buffer read-write
 - 105 pose evaluations / sec. (GeForce 3)

Video-based Rendering 3DPVT04 Tutorial


Model Matching

MPI
INFORMATIK

8-bit stencil buffer => 1 frame buffer read/write

Video-based Rendering 3DPVT04 Tutorial

Initialization



Initialization pose → Initial model positioning → Body Pose Estimation


Body Pose Estimation → Uniform skeleton rescaling → Non-uniform segment scaling → Body Pose Estimation

Non-uniform segment scaling: Four Bézier scale local vertex coordinates per segment => find control values


Uniform skeleton rescaling

Video-based Rendering 3DPVT04 Tutorial

Body Pose Estimation





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



Pose Parameters at $t-1$ → Grid search → Pose Parameters at t


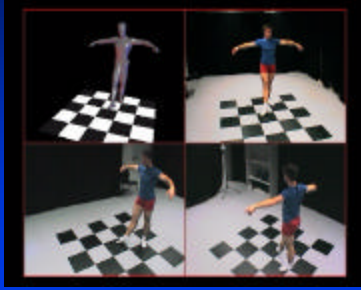
Video-based Rendering 3DPVT04 Tutorial

Model-to-Silhouette Matching


Video-based Rendering 3DPVT04 Tutorial

Motion Capture

Video-based Rendering 3DPVT04 Tutorial

Model Fitting Acceleration



Performance bottlenecks


- Energy function
 - Transfer bandwidth (frame buffer read/write)
 - Rendering model geometry

Energy function modification

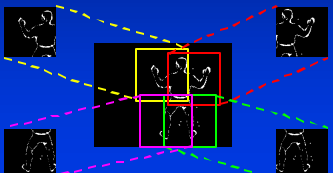
- Optimize body parts using sub-windows of image plane
- Pre-render un-changing body parts
- Distributed implementation

Video-based Rendering 3DPVT04 Tutorial

Sub-Window Optimization



- Render energy function in sub-area of image plane
- Conservative sub-window-size
- Centered around projected COG of body part
- Reduces memory-bandwidth



Video-based Rendering 3DPVT04 Tutorial

Body Part Pre-Rendering

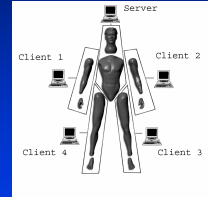
- Reduced number of triangles per function evaluation
 - Render only currently optimized geometry
- Error from unchanging body parts
 - Correct by pre-rendering and masking



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

Distributed Implementation

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



Results

Average fitting times (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

Refined Model Fitting

Silhouettes

- Robustly detectable
- Sensitive to large-scale motion

Object texture

- Highly detailed
- Sensitive to small movements

► **Exploit texture for fine-tuning model parameters**

2D Optical Flow

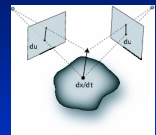
- Temporal image intensity change due to motion \mathbf{u} in image plane $I(\mathbf{p}, t) = I(\mathbf{p} - \mathbf{u} \cdot t, 0)$
- Optical flow constraint $\nabla I(\mathbf{p}, t) \cdot \mathbf{u} + I_t(\mathbf{p}, t) = 0$
 - 1 equation, 2 unknowns
- Smoothness assumption
 - Weighted least-squares fit (Lucas, Kanade)

$$\min_{\mathbf{u}} \left(\sum_{\mathbf{p} \in W} W^2(\mathbf{p}) [\nabla I(\mathbf{p}, t) \cdot \mathbf{u} + I_t(\mathbf{p}, t)]^2 \right)$$

– $W(\mathbf{p})$: Gaussian neighborhood around pixel \mathbf{p}

3D Motion Field

- **Optical Flow** \mathbf{u}_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
 - Solve for 3D motion $d\mathbf{x}/dt$



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

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

3D Motion Field

• 3D motion

- Point visible from N cameras i in sub-sequent time frames
- Determine Optical Flow u_i, v_i
- Solve linear system

$$\begin{bmatrix} \frac{\partial u_i}{\partial x} & \frac{\partial u_i}{\partial y} & \frac{\partial u_i}{\partial z} \\ \frac{\partial v_i}{\partial x} & \frac{\partial v_i}{\partial y} & \frac{\partial v_i}{\partial z} \end{bmatrix} \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \frac{dz}{dt} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ \cdot \\ \cdot \\ u_N \\ v_N \end{bmatrix}$$

• Least-squares solution

- Singular value decomposition
- 3D motion vector for each vertex

Motion Field-guided Model Fitting Algorithm

• Estimate model pose for $t+1$

- Silhouette fitting

• Render image estimates $I'_{j,t+1}$

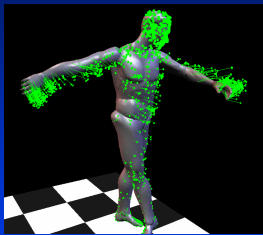
- Texture model with images $I_{j,t}$

• 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

Differential Pose Update



Before motion field enhancement



After registering torso, head, and upper limbs

[Theobalt et al., Pacific Graphics 2003]

Registration Model – Motion Field

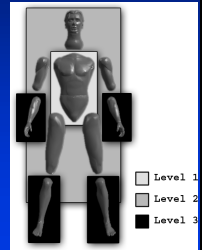
• Absolute orientation problem

$$\sum_i^N \|x_{2,i} - Rx_{1,i} - c\|^2$$

- Least-squares minimization using quaternion representation (Horn)

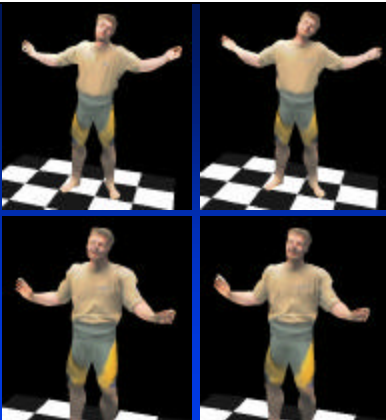
• Hierarchical pose update

- Torso
- Upper arms & legs, head
- Lower extremities



Results

- Silhouette only
 - Disparity mismatch
- Differential pose update
 - Accurate texture projection & blending



Video Texturing

Time-varying texture

- Cloth folds
- Local shadows
- Facial expressions



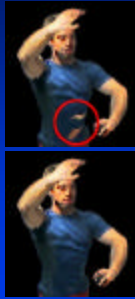
Detail preservation

- Small inaccuracies between geometry and silhouettes
- "Extended soft shadowing"
- Non-linear texture blending



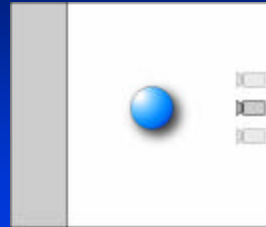
Visibility Computation

- Geometry not perfectly aligned with silhouette
 - Pre-compute visibility from all input cameras
- "Extended Soft Shadowing"
 - Compute visibility from a set of displaced camera views
 - Texture warping: expand texture information at silhouette boundary into background



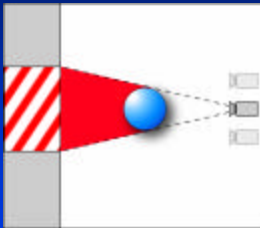
Visibility

Displaced camera views



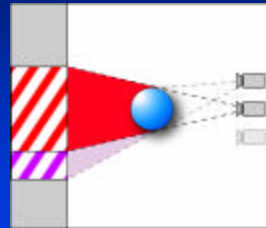
Visibility

Displaced camera views



Visibility

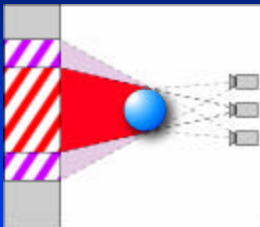
Displaced camera views



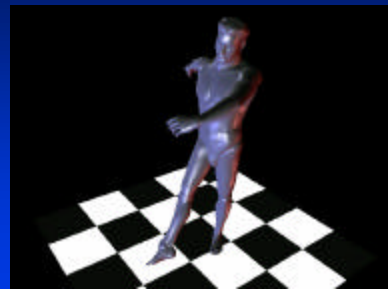
Visibility

Displaced camera views

Zero visibility



Visibility



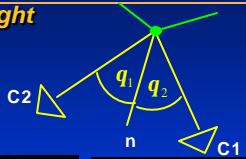

MPI
INFORMATIK

Multi-View Texturing

Surface dependent weight

- No view-dependent effects

$$w_i = \frac{1}{\Theta_i}$$

$$w_i = \frac{1}{(\max(w_i) + 1 - w_i)^a}$$



a = 0 a = 3 a = 15

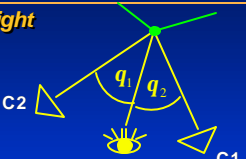

Video-based Rendering 3DPVT04 Tutorial

MPI
INFORMATIK

Multi-View Texturing

Viewpoint-dependent weight

- View-dependent effects

$$r_i = \frac{1}{\Theta_i}$$



Video-based Rendering 3DPVT04 Tutorial

MPI
INFORMATIK

Rendering

- Render model geometry in captured pose
- Dynamic multi-view texture
 - Projective texturing
 - Per-vertex blending
 - Texture information (camera image) $tex_i(j)$
 - Spatial blending weights $w_i(j), r_i(j)$
 - Visibility $Vis_i(j)$



$$c_j = \sum_i^N Vis_i(j) \cdot r_i(j) \cdot w_i(j) \cdot tex_i(j)$$

Video-based Rendering 3DPVT04 Tutorial

MPI
INFORMATIK

Free-Viewpoint Video Viewer


- Pose fitting: 7-12 sec / frame
- Rendering: 30 fps (GeForce3)

Video-based Rendering 3DPVT04 Tutorial

MPI
INFORMATIK

Result



Video-based Rendering 3DPVT04 Tutorial

MPI
INFORMATIK

Summary

- Visual Hull Rendering**
 - Real-time processing
 - Approximate shape
 - Reconstruction & rendering on graphics hardware
 - Improved rendering quality
- Model-based Free-Viewpoint Video**
 - Off-line processing
 - High-quality surface description
 - Temporally consistent
 - Animation representation
 - High visual fidelity

Video-based Rendering 3DPVT04 Tutorial

Outlook

- **Acquisition**
 - Mobile recording
 - Additional illumination, depth, sound, ...
- **Processing / Reconstruction**
 - Reflectance characteristics
 - Non-rigid geometry
 - Illumination
- **Tools**
 - Motion editing
 - Video-based character representation

Online Resources

Visual Hull Rendering

- graphics.lcs.mit.edu/~wojciech/vh
- www.ri.cmu.edu/projects/project_333.html
- vision.kuee.kyoto-u.ac.jp/CDV/PRJ/index.html
- www.grovis.de/ming/visualhull/Visual_Hull_Projects.html

Model-based Analysis & Synthesis

- www.grovis.de/fvv

Thanks for Coming

Questions & Answers

Marcus Magnor magnor@mpii.de