Acquiring Detailed 3D Models from Images and Video

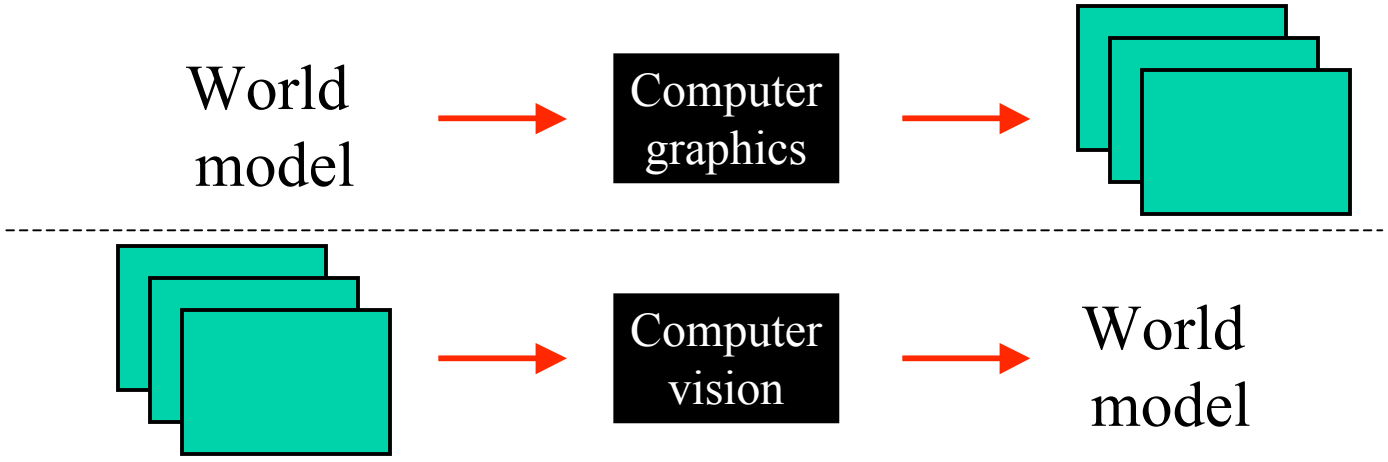
NIPS 2004 Tutorial

Microsoft Research

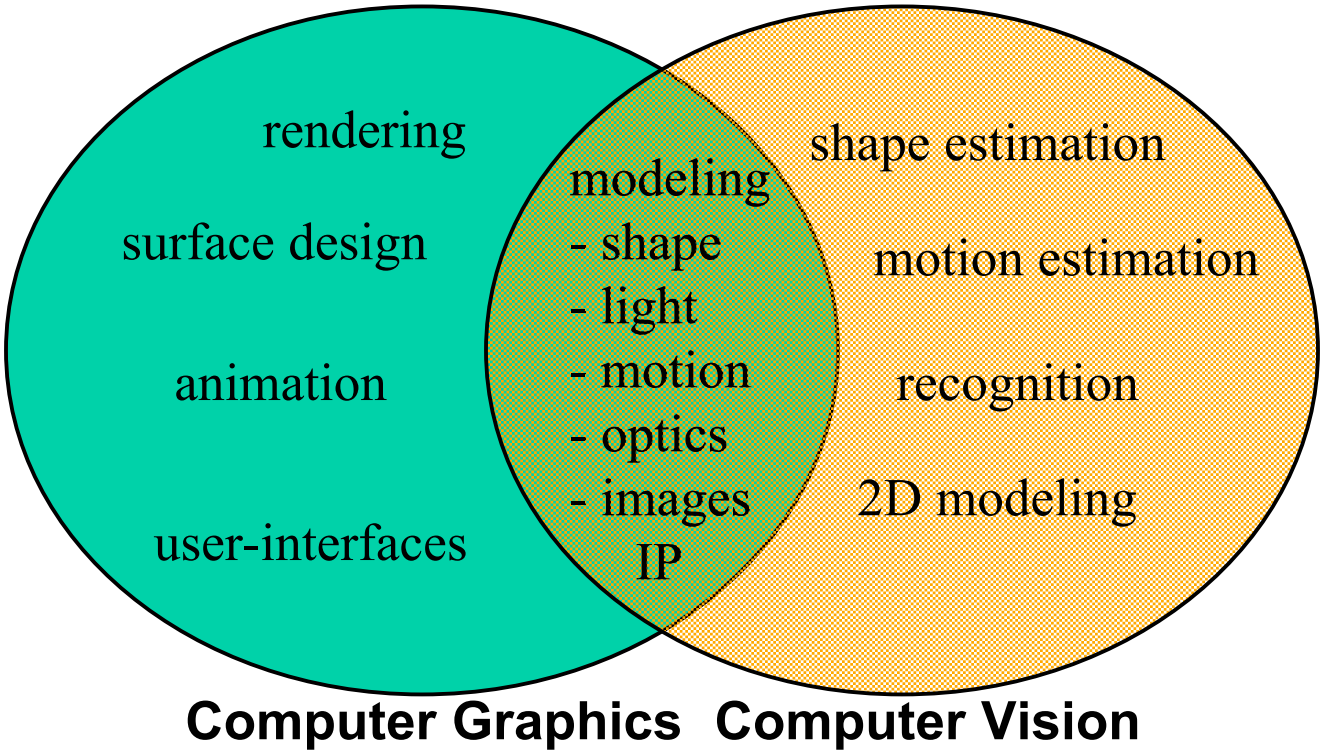
What is Computer Vision?

What is Computer Vision?

- Image Understanding (AI, behavior)
- Computer emulation of human vision
- A sensor modality for robotics
- Inverse of Computer Graphics



Intersection of Vision and Graphics



Computer Vision [Trucco&Verri'98]

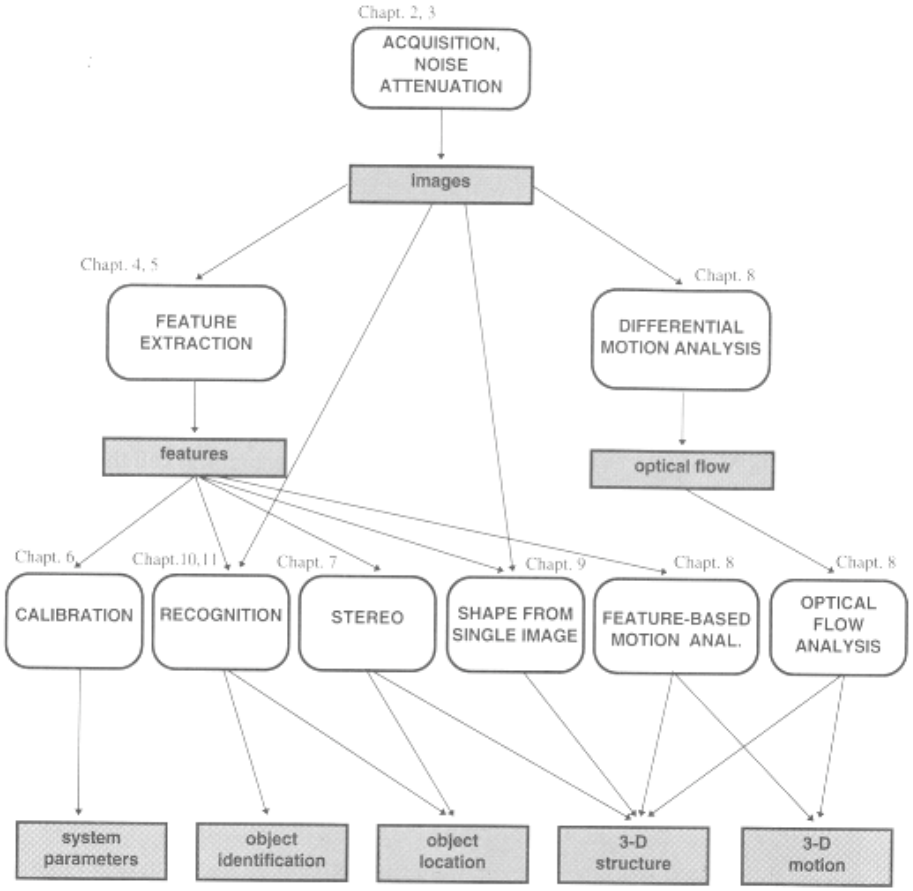
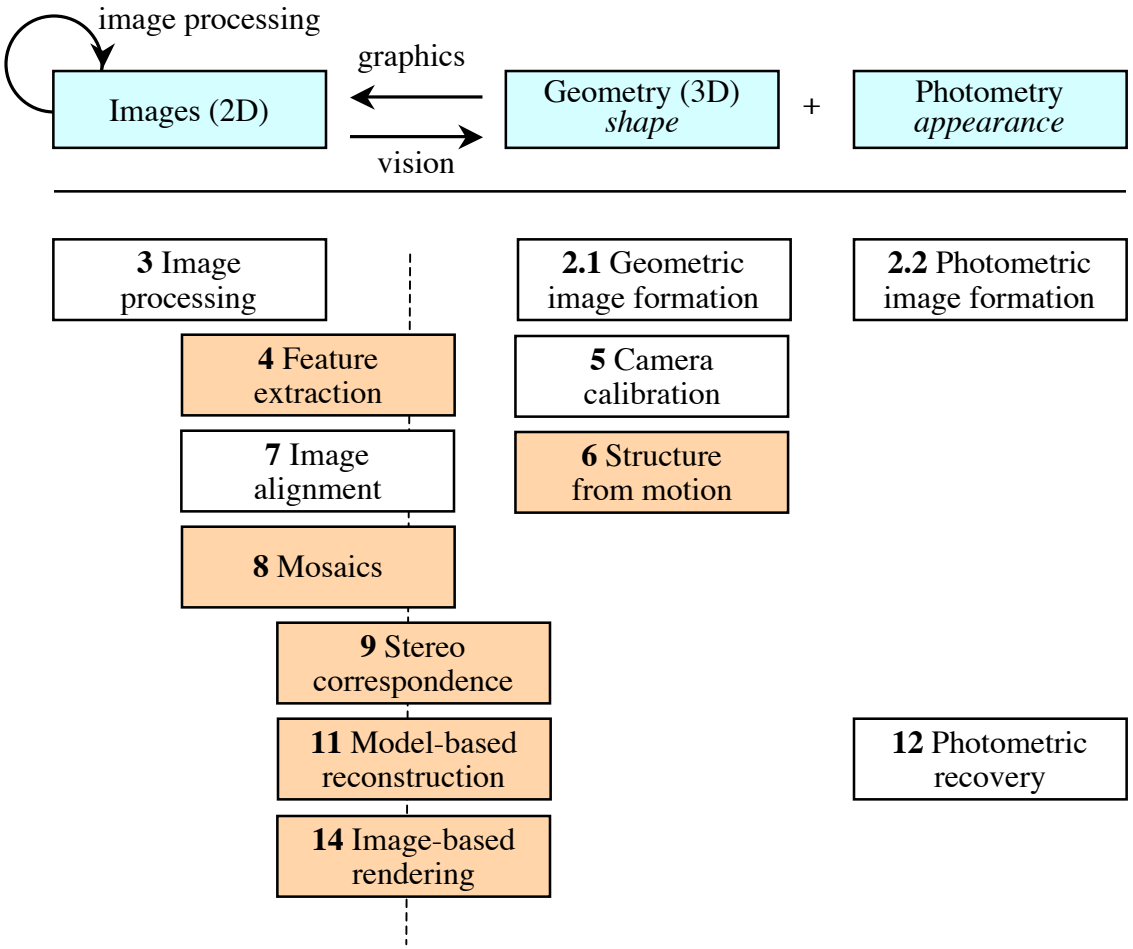


Figure 1.7 The book at a glance: method classes (white boxes), results (grey boxes), their interdependence, and where to find the various topics in this book.

Image-Based Modeling [Sz'0?]



Outline

- Motivating applications
- Problem setting: “inverse graphics”
- Feature detection (see also NIPS’03)
- Image mosaics: simple 2D modeling
- Structure from motion (sparse 3D)
- Stereo reconstruction (dense 3D)
- Model-based reconstruction
- Image- and video-based rendering
- *Opportunities for inference/learning*

Some applications

Panoramic Mosaics


+

+ … +

=

Concentric Mosaics

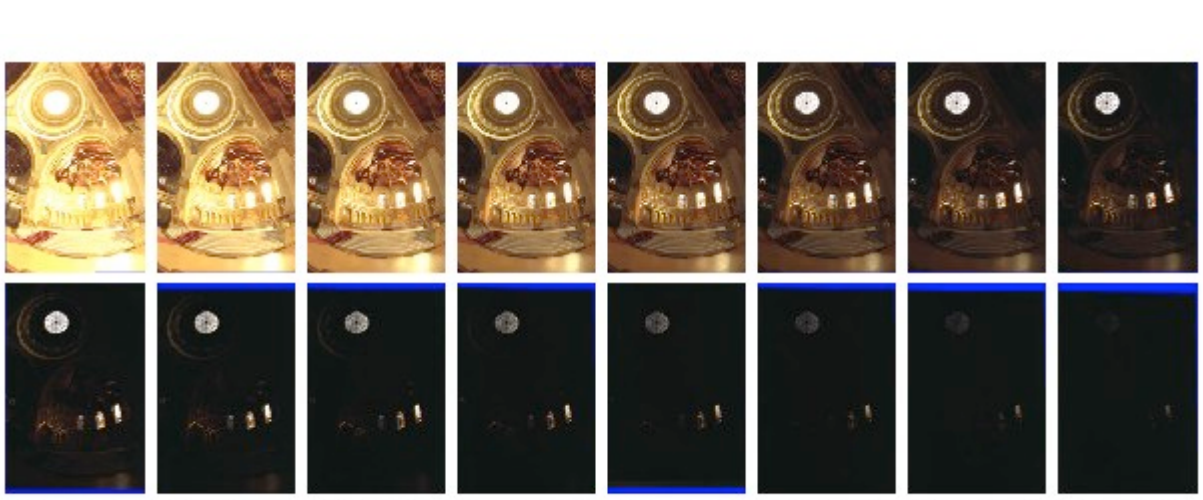
Interpolate between several panoramas to give a 3D depth effect

[Shum & He, SIGGRAPH'99]

Image Enhancement

High dynamic range photography
[Debevec *et al.*'97; Mitsunaga & Nayar'99]

- combine several different exposures together



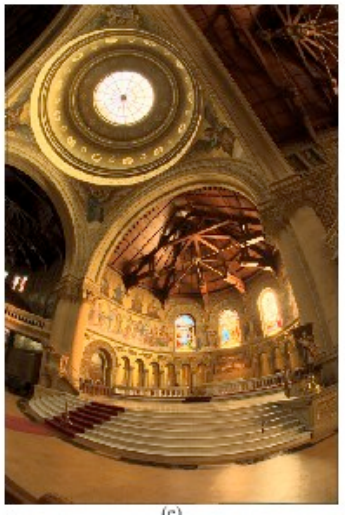
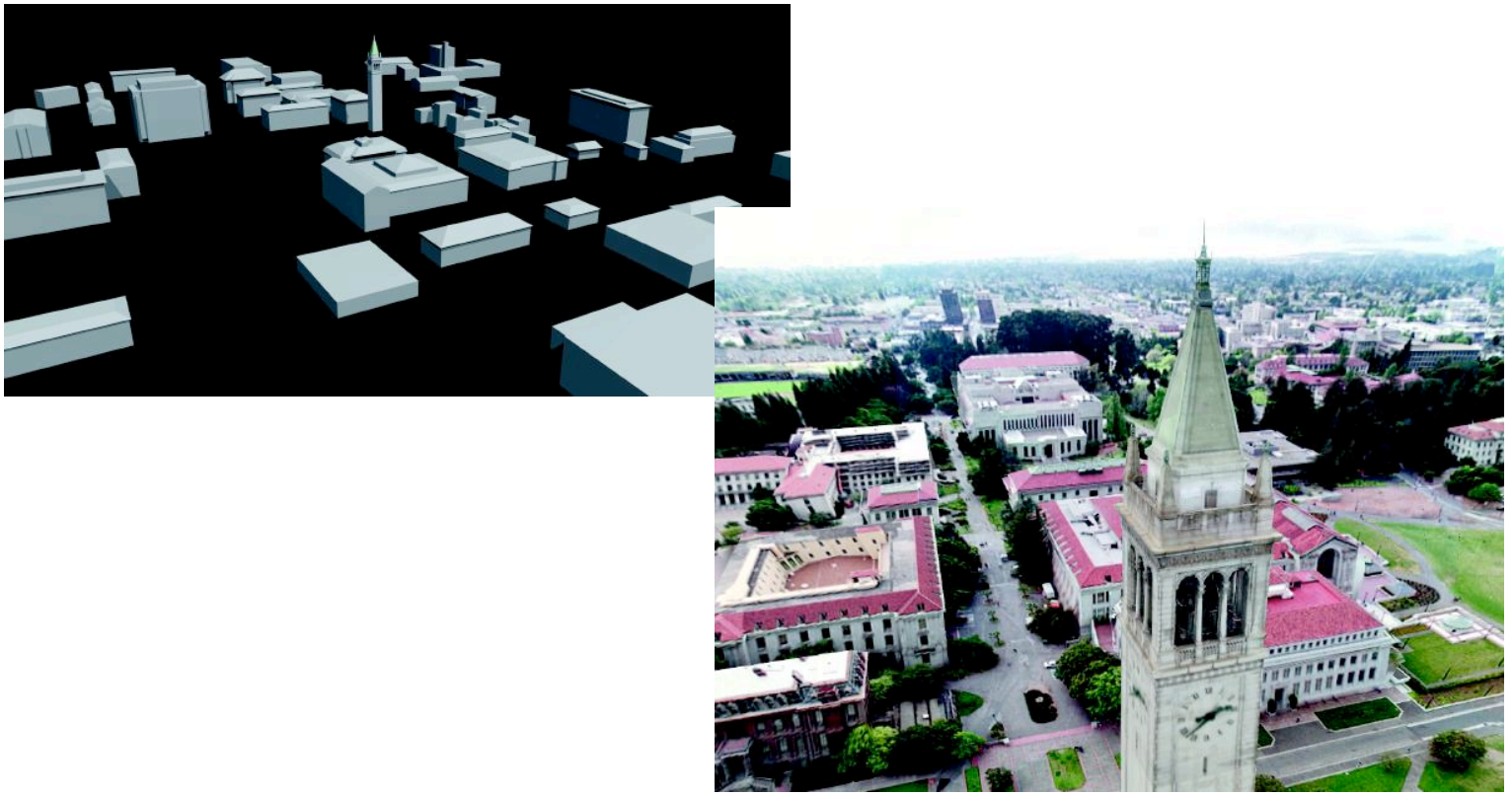


Figure 6: *Sixteen photographs of a church taken at 1-stop increments from 30 sec to —— sec. The sun is directly behind the rightmost stained glass window, making it especially bright. The blue borders seen in some of the image margins are induced by the image registration process.*

3D Shape Reconstruction

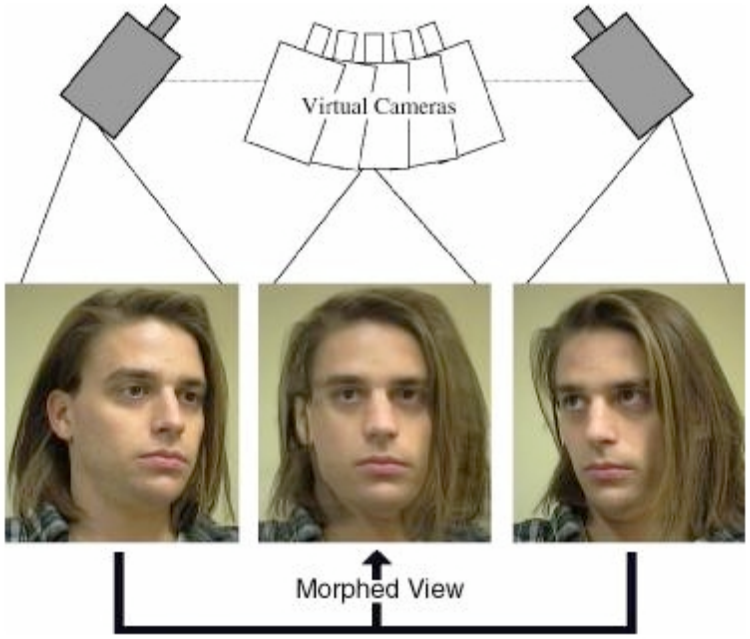


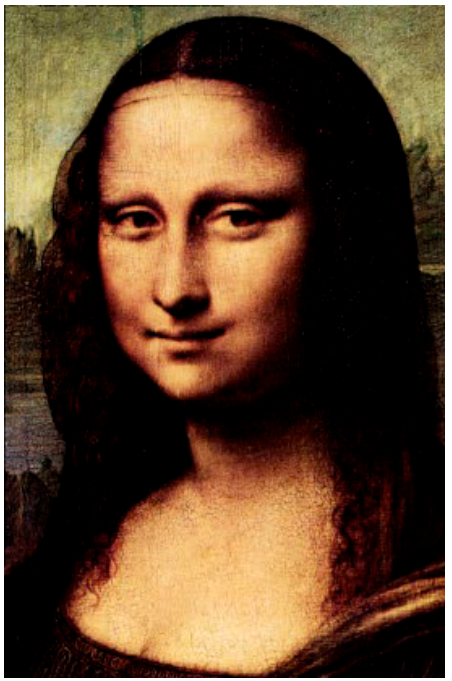
Debevec, Taylor, and Malik, SIGGRAPH 1996

Face Modeling

View Morphing

Morph between pair of images using epipolar geometry [Seitz & Dyer, SIGGRAPH'96]





Z-keying: mix live and synthetic

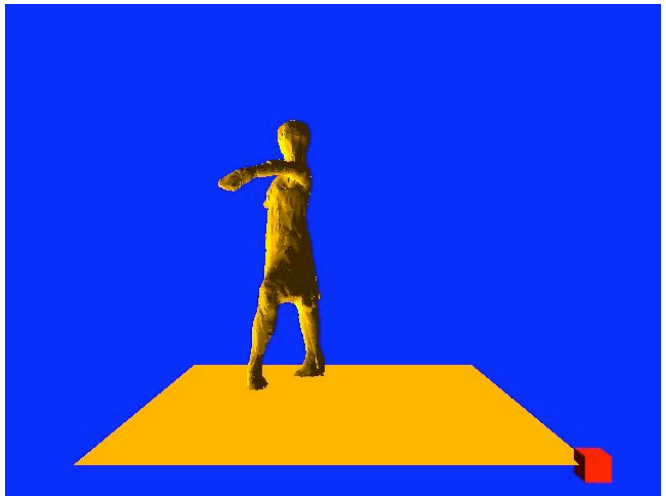
Takeo Kanade, CMU (Stereo Machine)

Virtualized Reality™

Takeo Kanade, CMU

- collect video from 50+ stream

reconstruct 3D model sequences

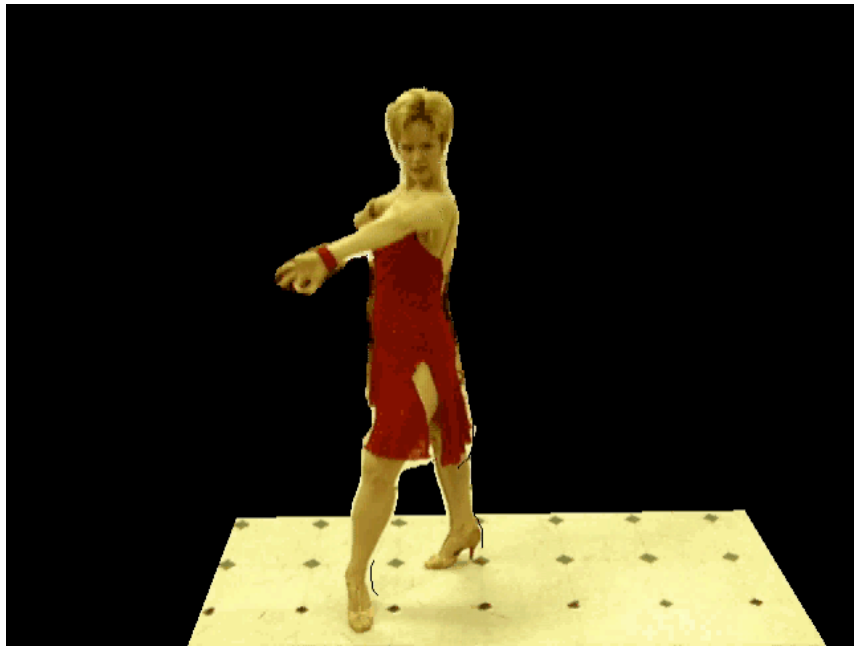


http://www.cs.cmu.edu/afs/cs/project/VirtualizedR/www/VirtualizedR.html

Virtualized Reality™

Takeo Kanade, CMU

- generate new video



- steerable version used for SuperBowl XXV "eye vision" system

Applications

- Panoramic mosaics, high-dynamic range (real estate, virtual tourism)
- 3D environment reconstruction (digital sets, place recognition)
- Face modeling and tracking
- Special effects (ILM, RealVis,2D3)
- Image and video editing (Avid, Adobe)
- Vehicle safety (MobilEye)

“Inverse graphics”

“Inverse graphics”

Barrow & Tennenbaum, 1978

- *Intrinsic Images:* separate an image into its constituent components:
  - distance
  - reflection
  - orientation
  - illumination

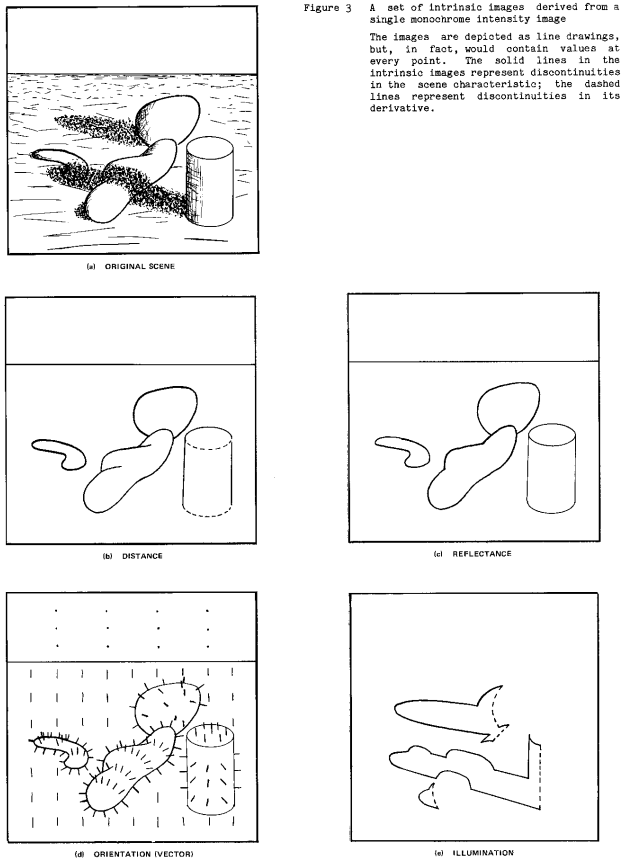
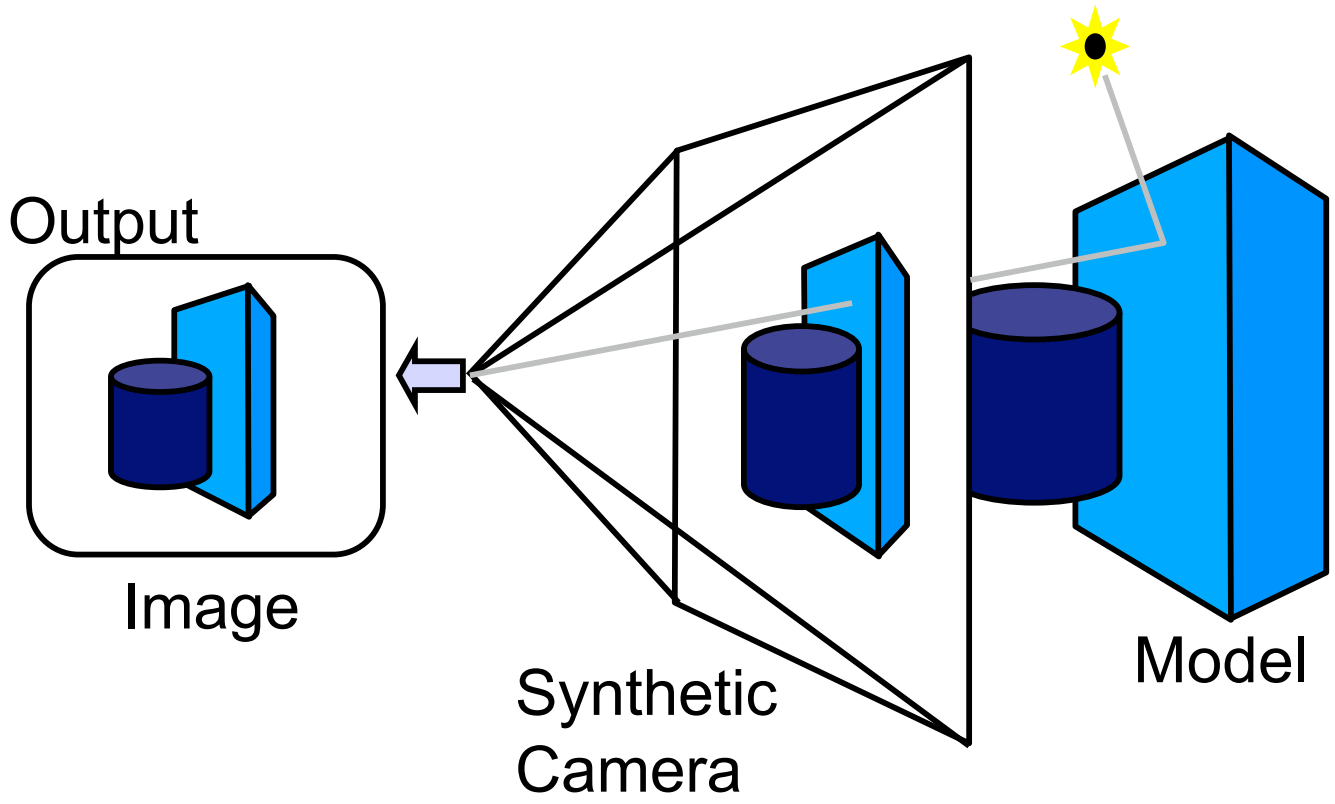


Figure 3 A set of intrinsic images derived from a single monochrome intensity image

The images are depicted as line drawings, but, in fact, would contain values at every point. The solid lines in the intrinsic images represent discontinuities in the scene characteristic; the dashed lines represent discontinuities in its derivative.

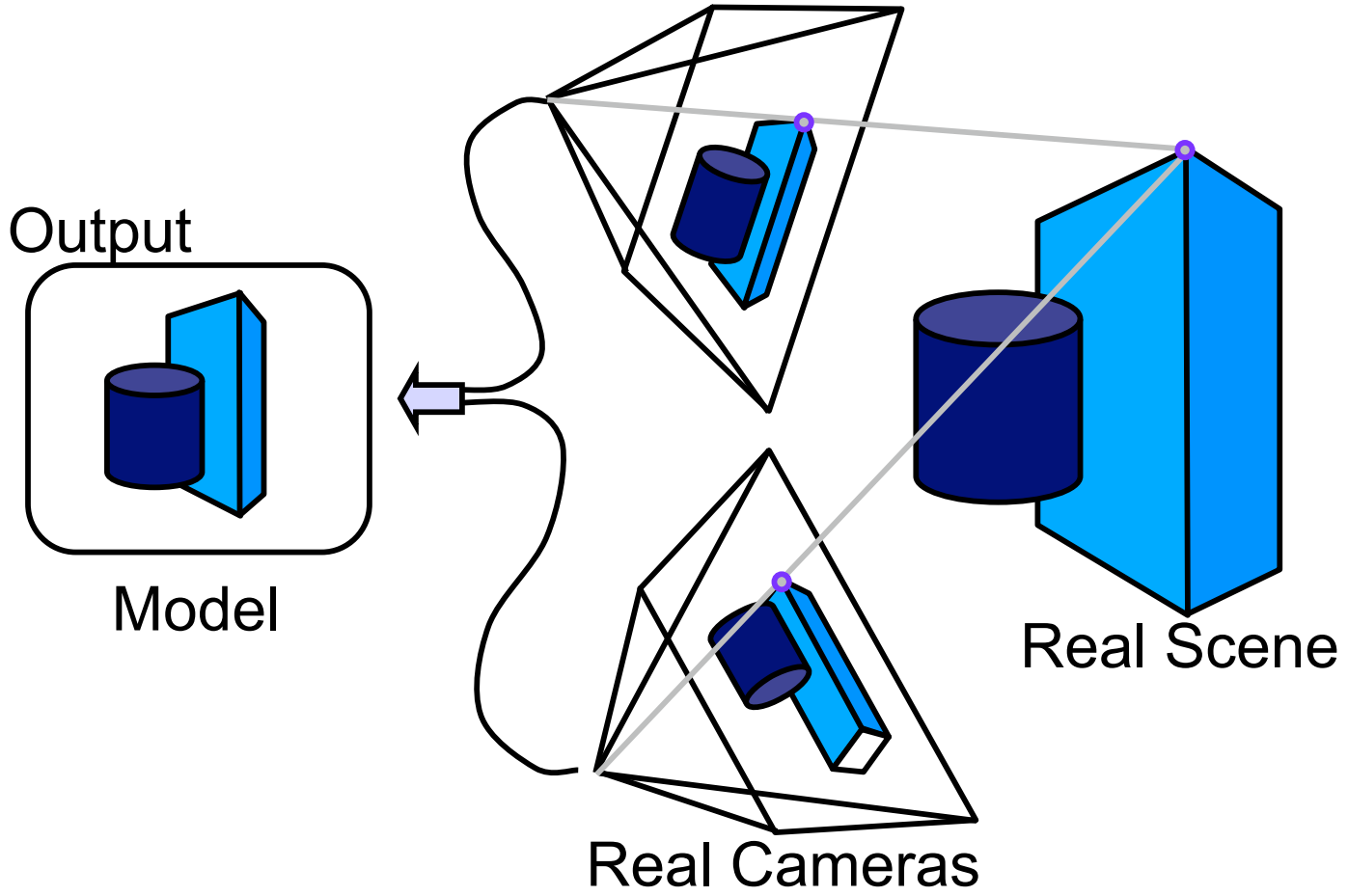
Characteristics from Images

Computer Graphics

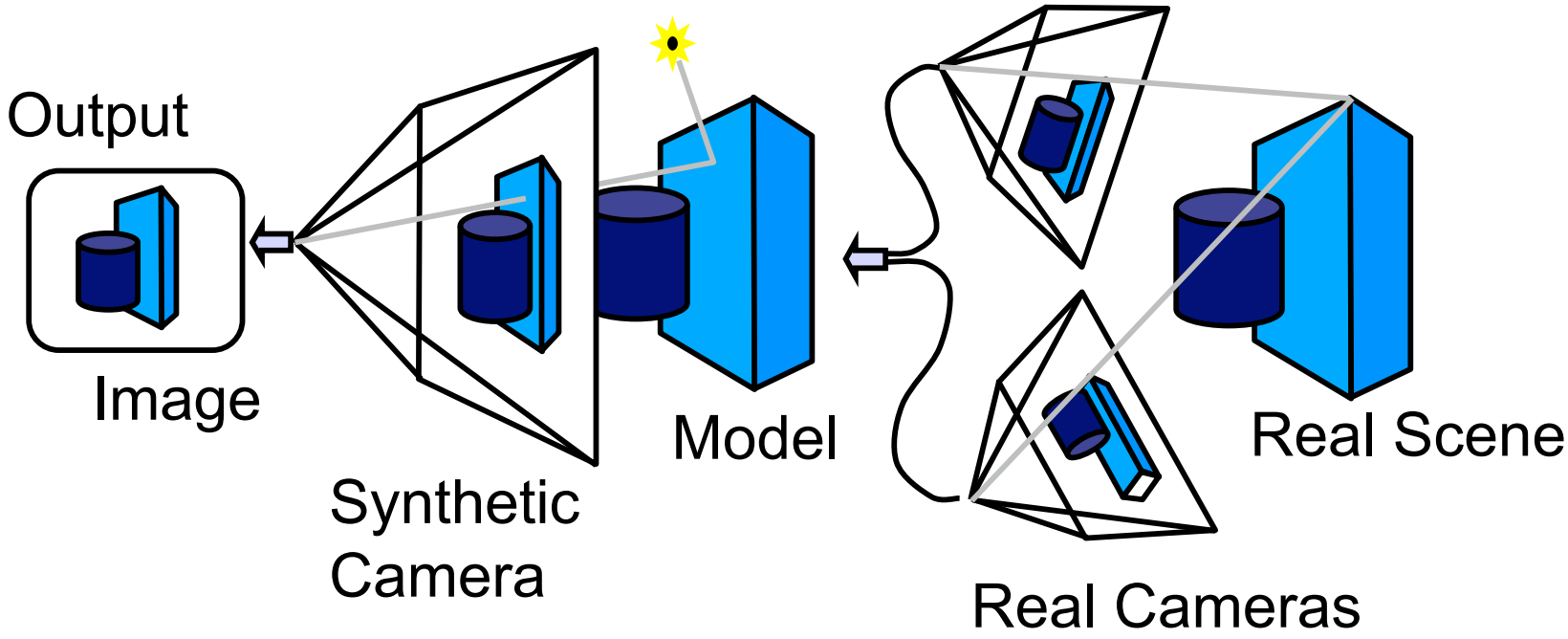


(slides courtesy of Michael Cohen)

Computer Vision



Combined


Output
Image
Synthetic
Camera
Model
Real Scene
Real Cameras

Computer Vision [Trucco&Verri'98]

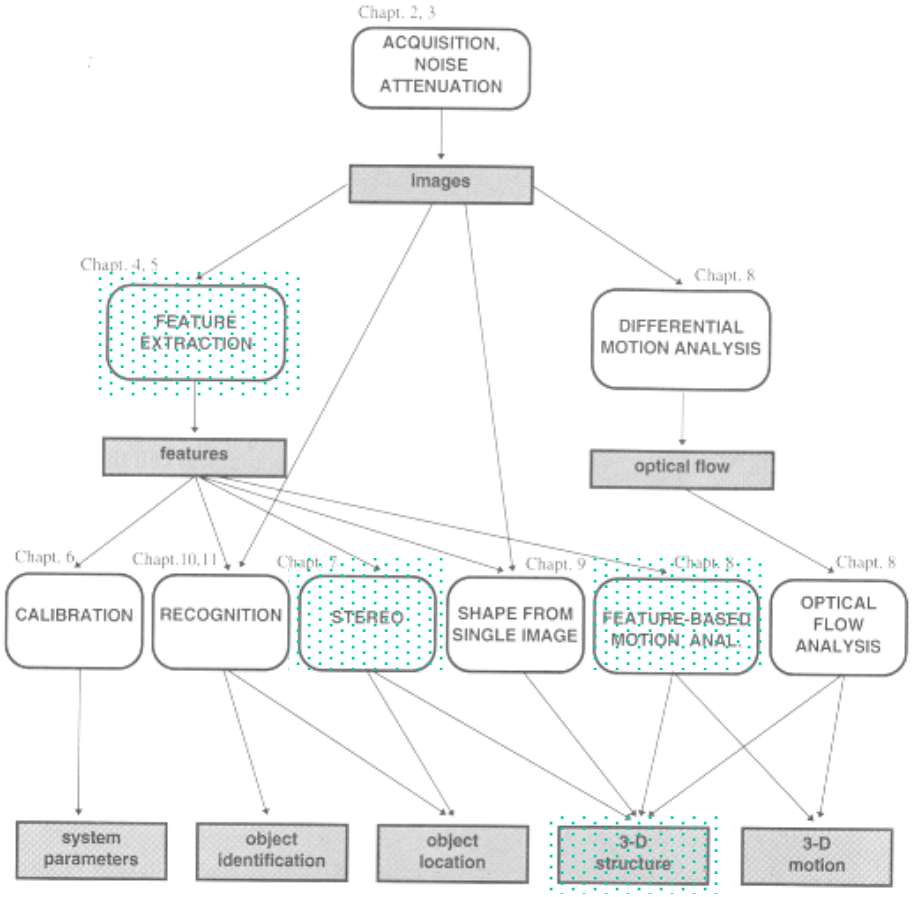
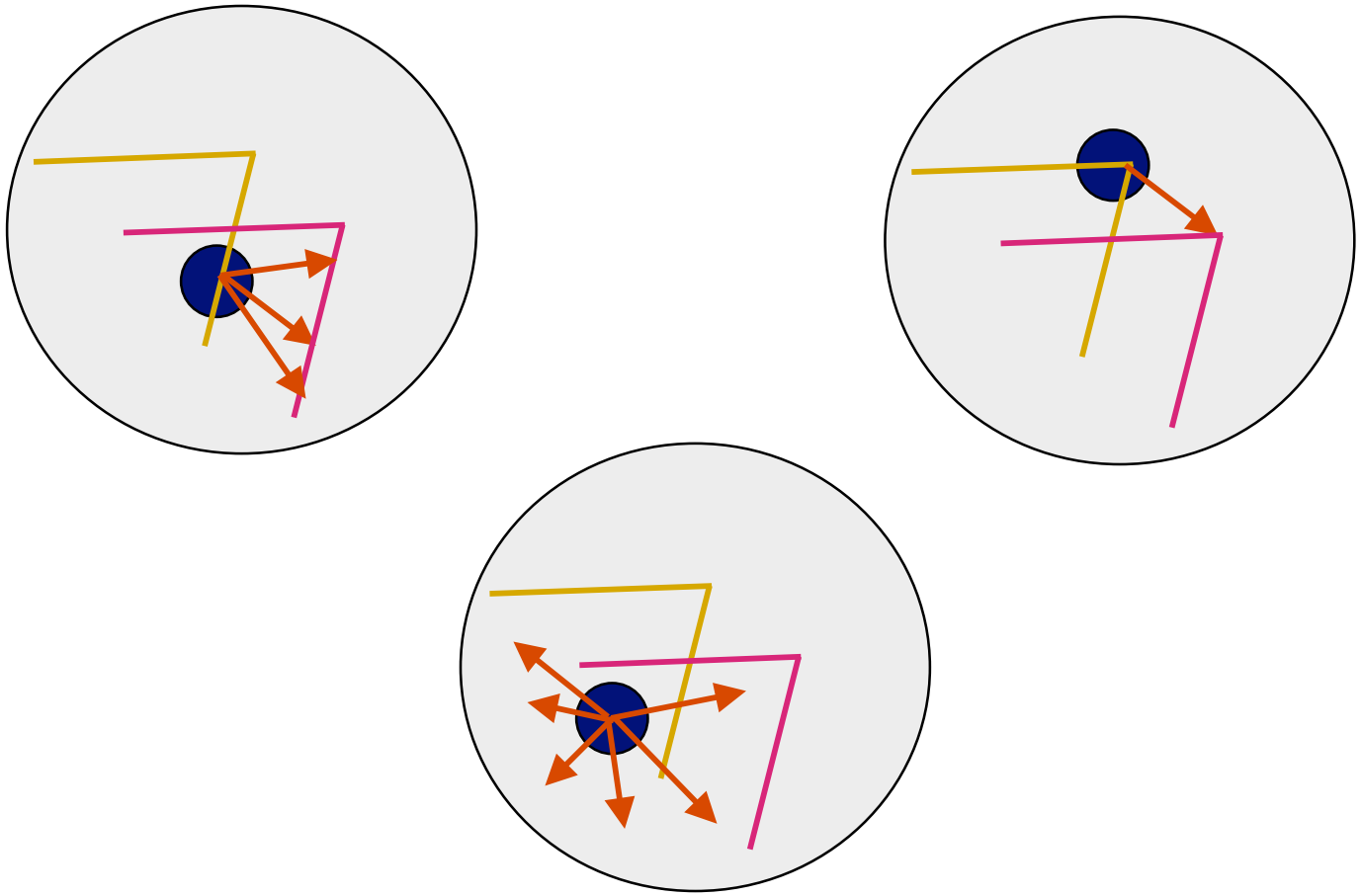


Figure 1.7 The book at a glance: method classes (white boxes), results (grey boxes), their interdependence, and where to find the various topics in this book.

Feature detection

Local Patch Analysis



Patch Translation

Assume a single velocity for all pixels within an image patch

$$E(u,v) = \sum_{x,y \in \Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t \right)^2$$

Minimizing

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$

$$\left(\sum \nabla I \nabla I^T \right) U = -\sum \nabla I I_t$$

On the LHS: sum of the 2x2 outer product tensor of the gradient vector

Singularities & Aperture Problem

Let $M = \sum (\nabla I)(\nabla I)^T$ and $b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$

- Algorithm: At each pixel compute U by solving $MU = b$

- M is singular if all gradient vectors point in the same direction
  -- e.g., along an edge
  -- of course, trivially singular if the summation is over a single pixel
  -- i.e., only *normal flow* is available (aperture problem)

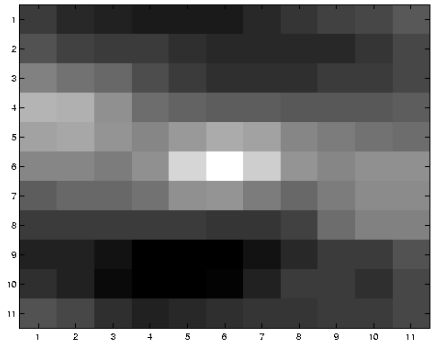
- Corners and textured areas are OK

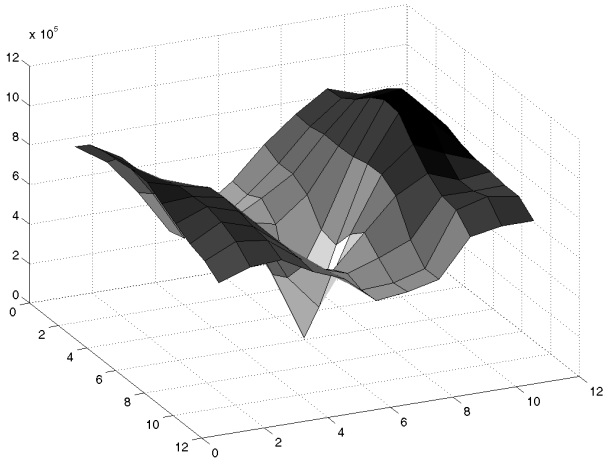
Correlation and SSD

For larger displacements, do template matching

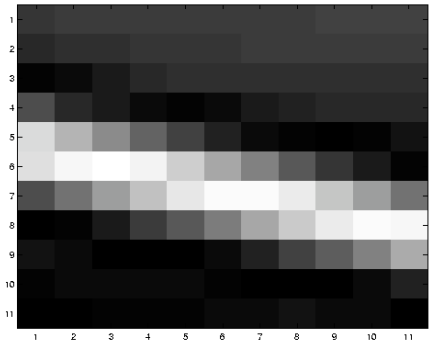
- Define a small area around a pixel as the template
- Match the template against each pixel within a search area in next image.
- Use a match measure such as correlation, normalized correlation, or sum-of-squares difference
- Choose the maximum (or minimum) as the match
- Sub-pixel interpolation also possible

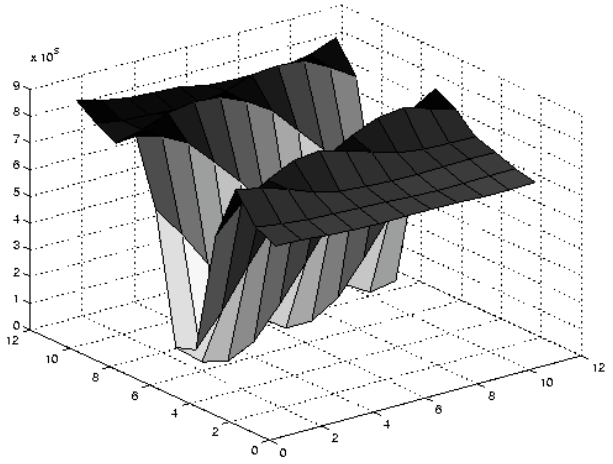
SSD Surface – Textured area



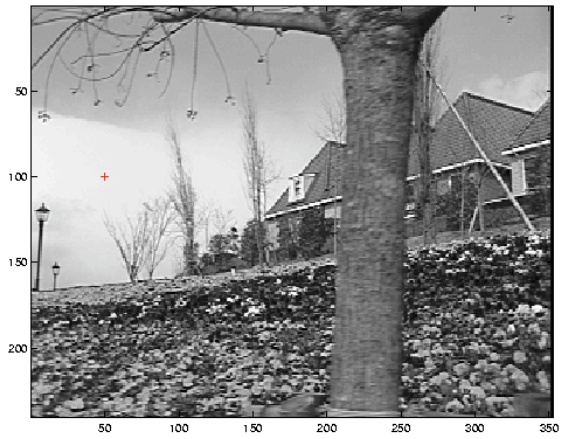


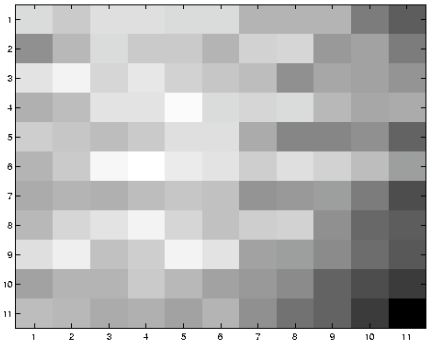
SSD Surface -- Edge

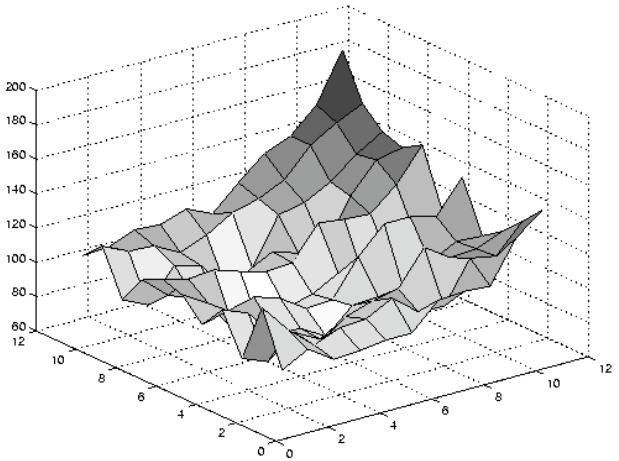




SSD Surface – homogenous area







Shi-Tomasi feature tracker

1. Find good features (min eigenvalue of 2×2 Hessian)
2. Use Lucas-Kanade to track with pure translation
3. Use affine registration with first feature patch
4. Terminate tracks whose dissimilarity gets too large
5. Start new tracks when needed

Tracking results

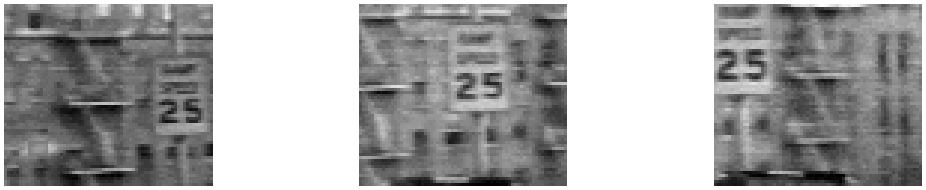


Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

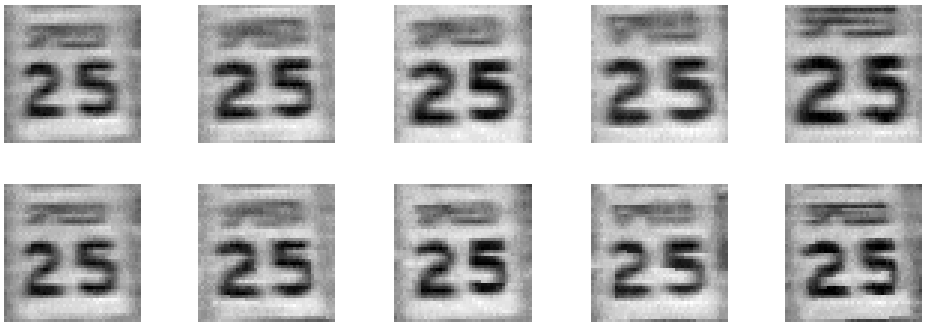


Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

Tracking results

Figure 13: Labels of some of the features in figure 11.

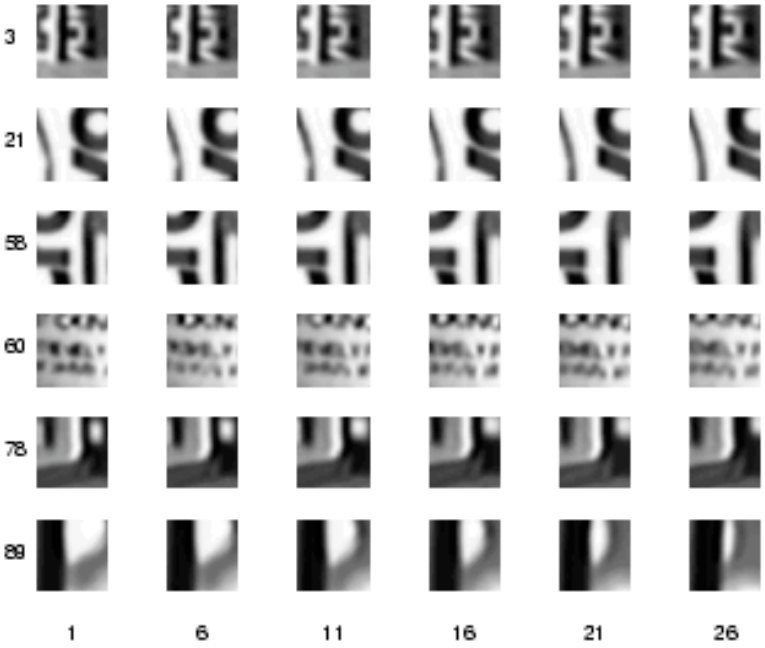


Figure 14: Six sample features through six sample frames.

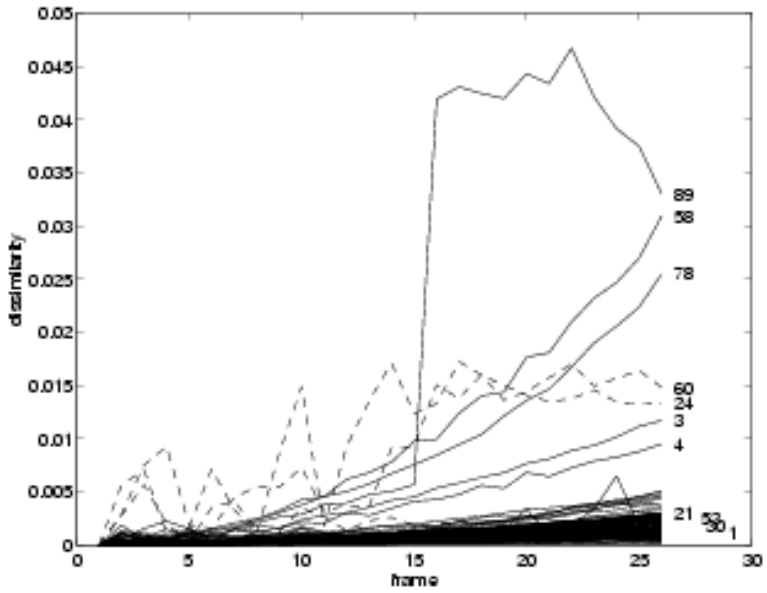


Figure 15: Affine motion dissimilarity for the features in figure 11. Notice the good discrimination between good and bad features. Dashed plots indicate aliasing (see text).

Features 24 and 60 deserve a special discussion, and

Multi-Scale Oriented Patches

Interest points

- Multi-scale "Harris" corners $\left(\sum \nabla I \nabla I^T\right) U = -\sum \nabla I I_t$
- Orientation from blurred gradient
- Geometrically invariant to similarity transforms

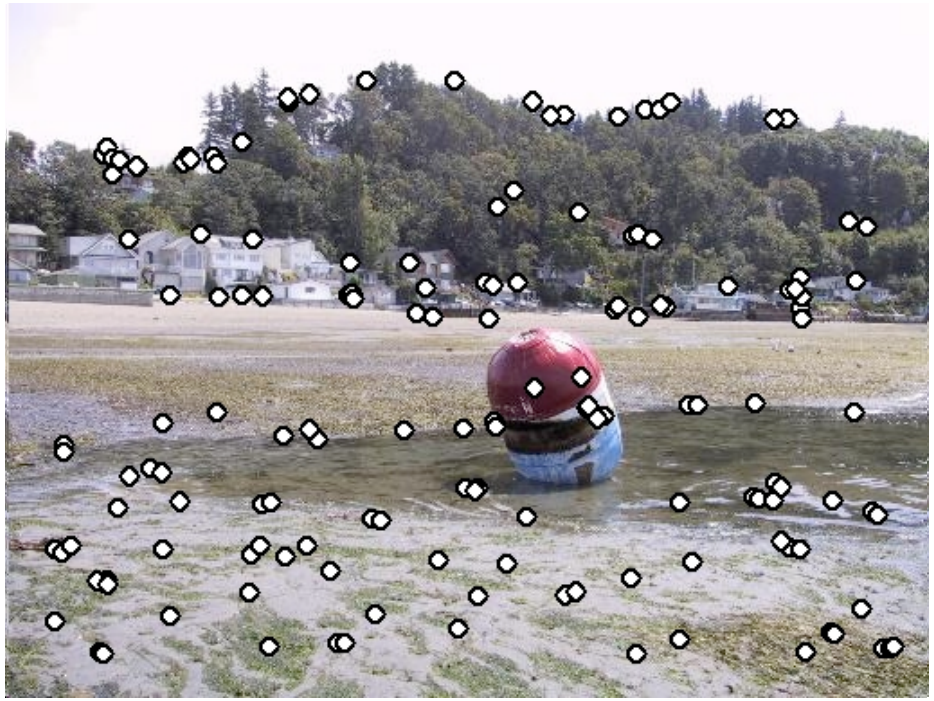
Descriptor vector

- Bias/gain normalized sampling of local patch (8x8)
- Photometrically invariant to affine changes in intensity

[Brown, Szeliski, Winder, CVPR'05?]

Feature density

Distribute points evenly over the image



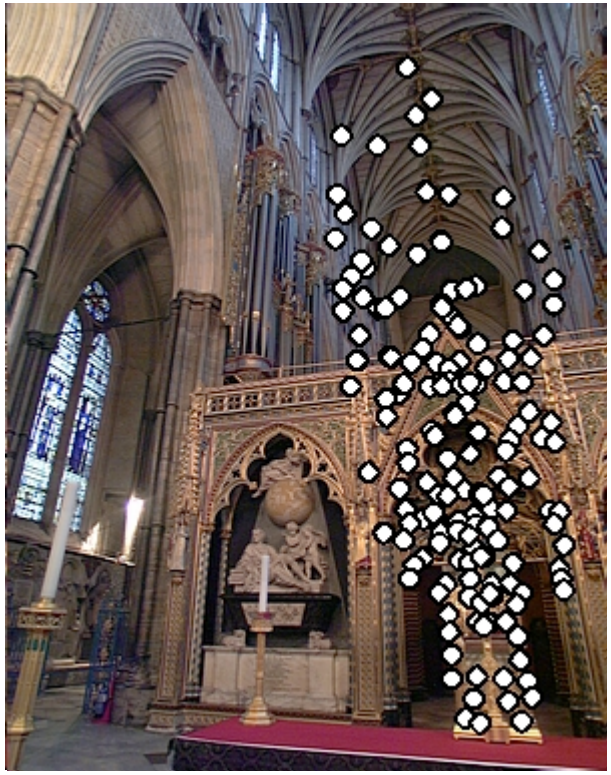
Descriptor Vector

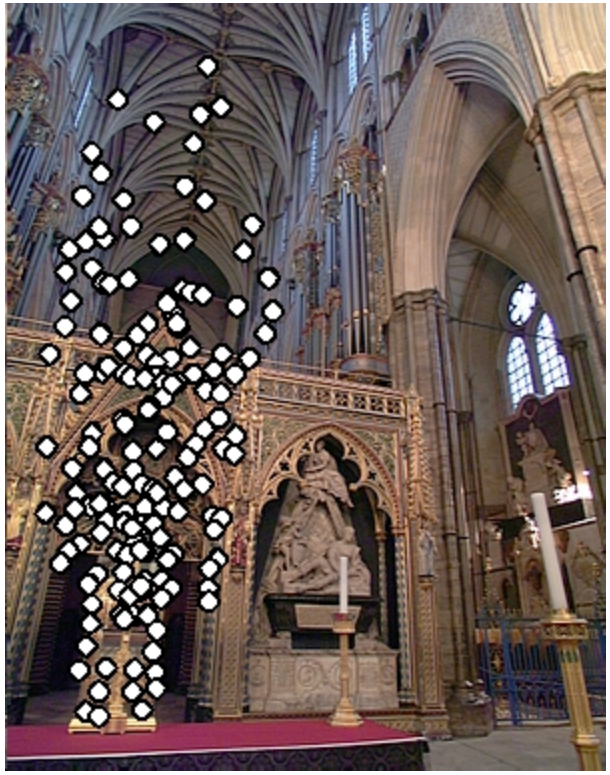
Orientation = blurred gradient

Similarity Invariant Frame

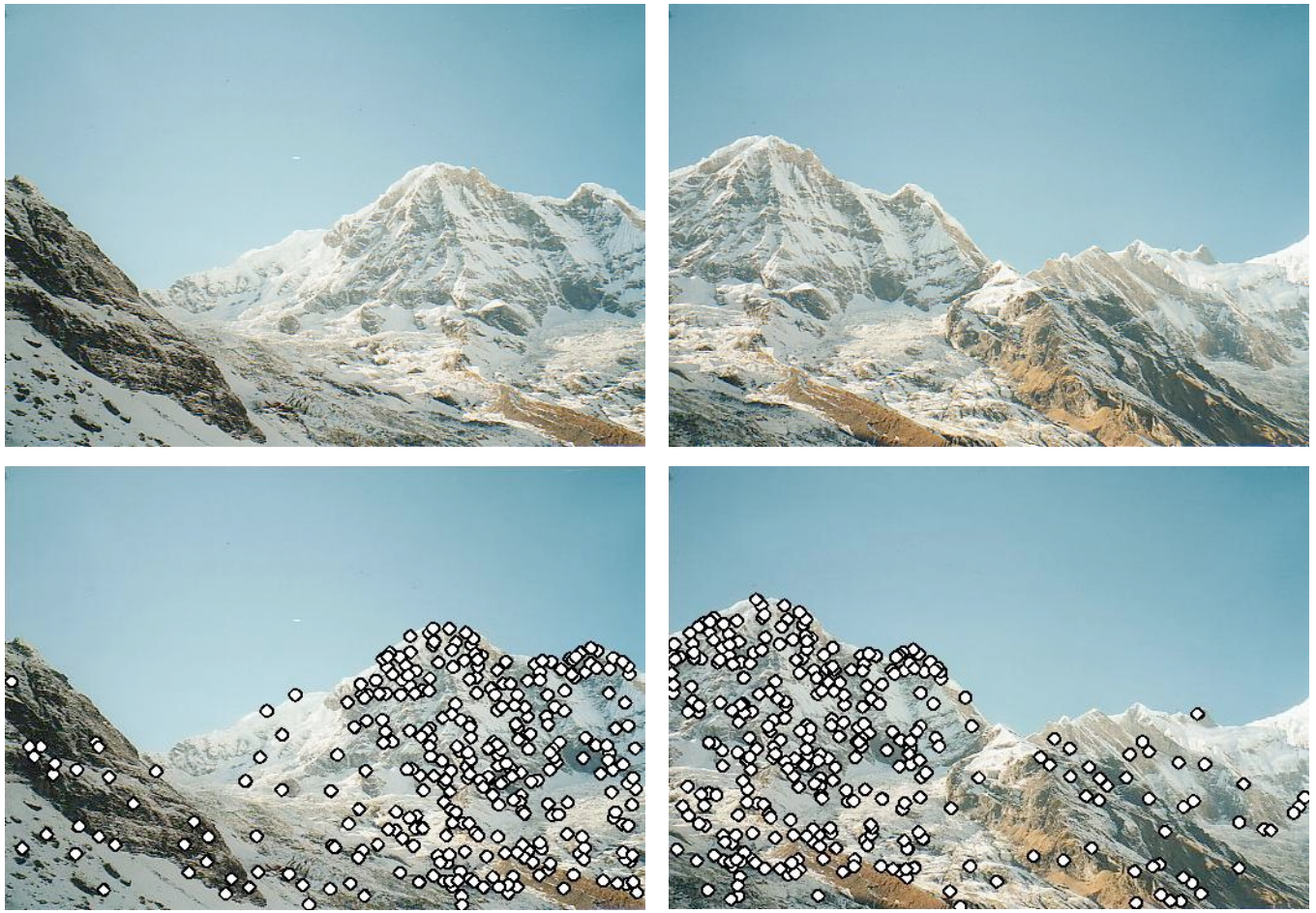
- Scale-space position (x, y, s) + orientation (θ)

Probabilistic Feature Matching

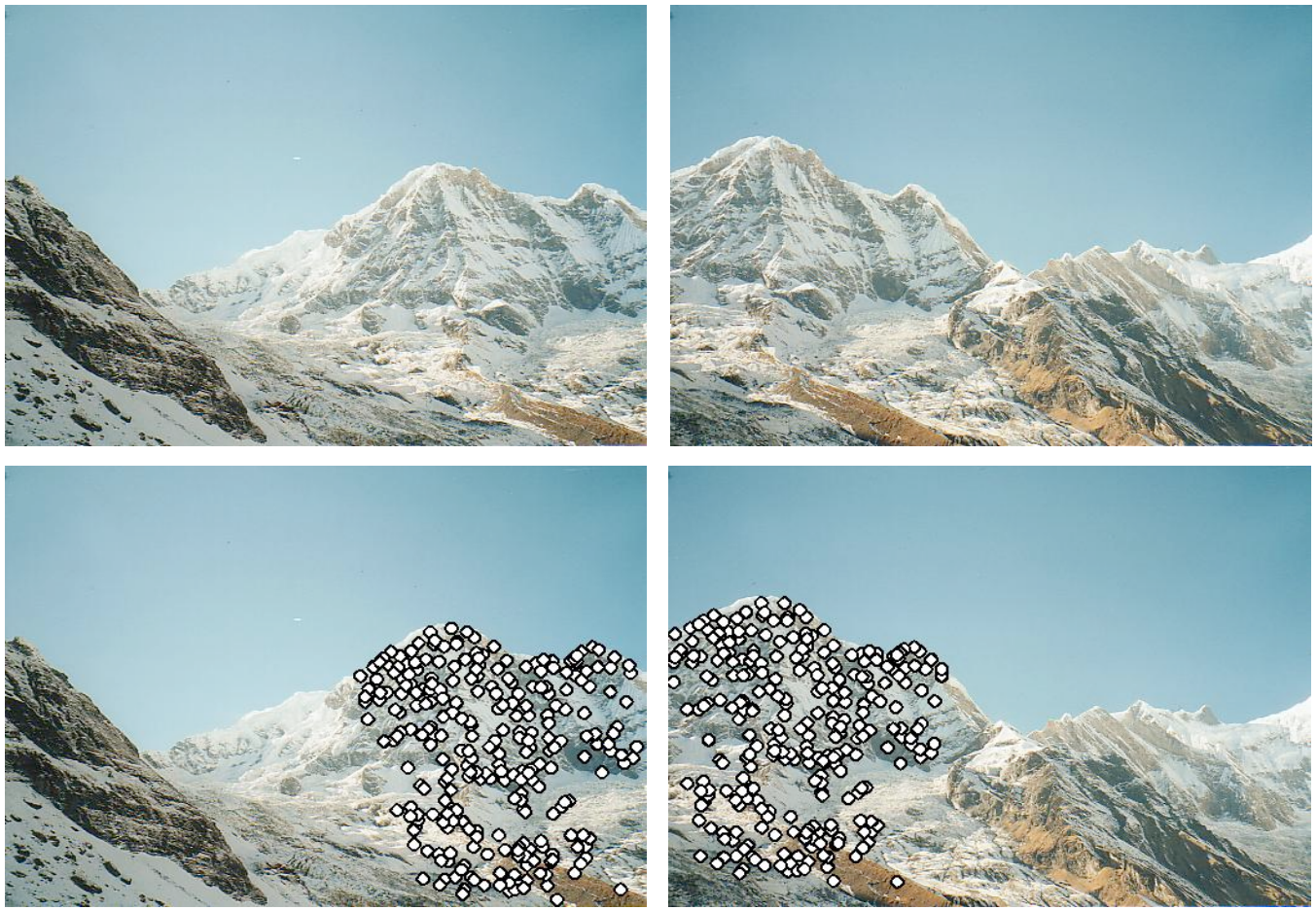




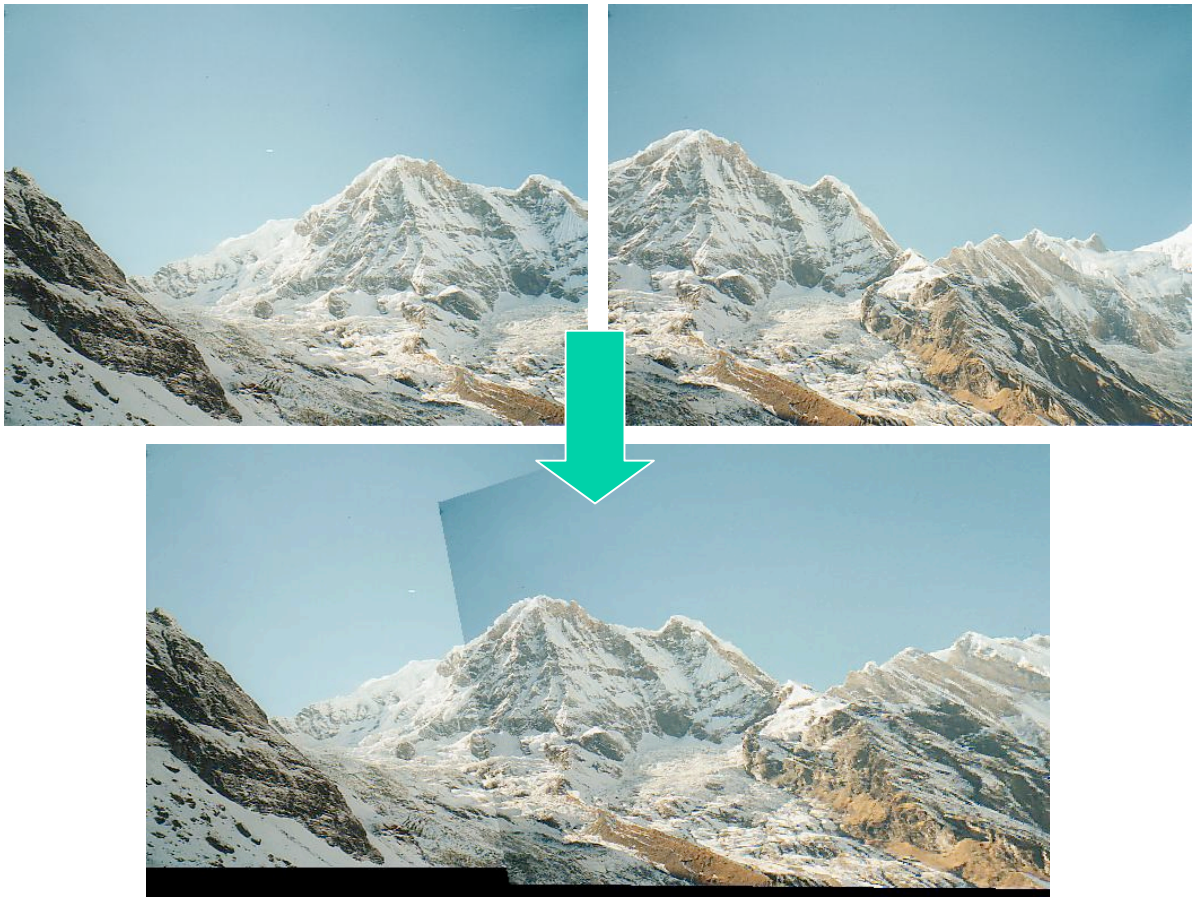
RANSAC motion model



RANSAC motion model



RANSAC motion model



Probabilistic model for verification

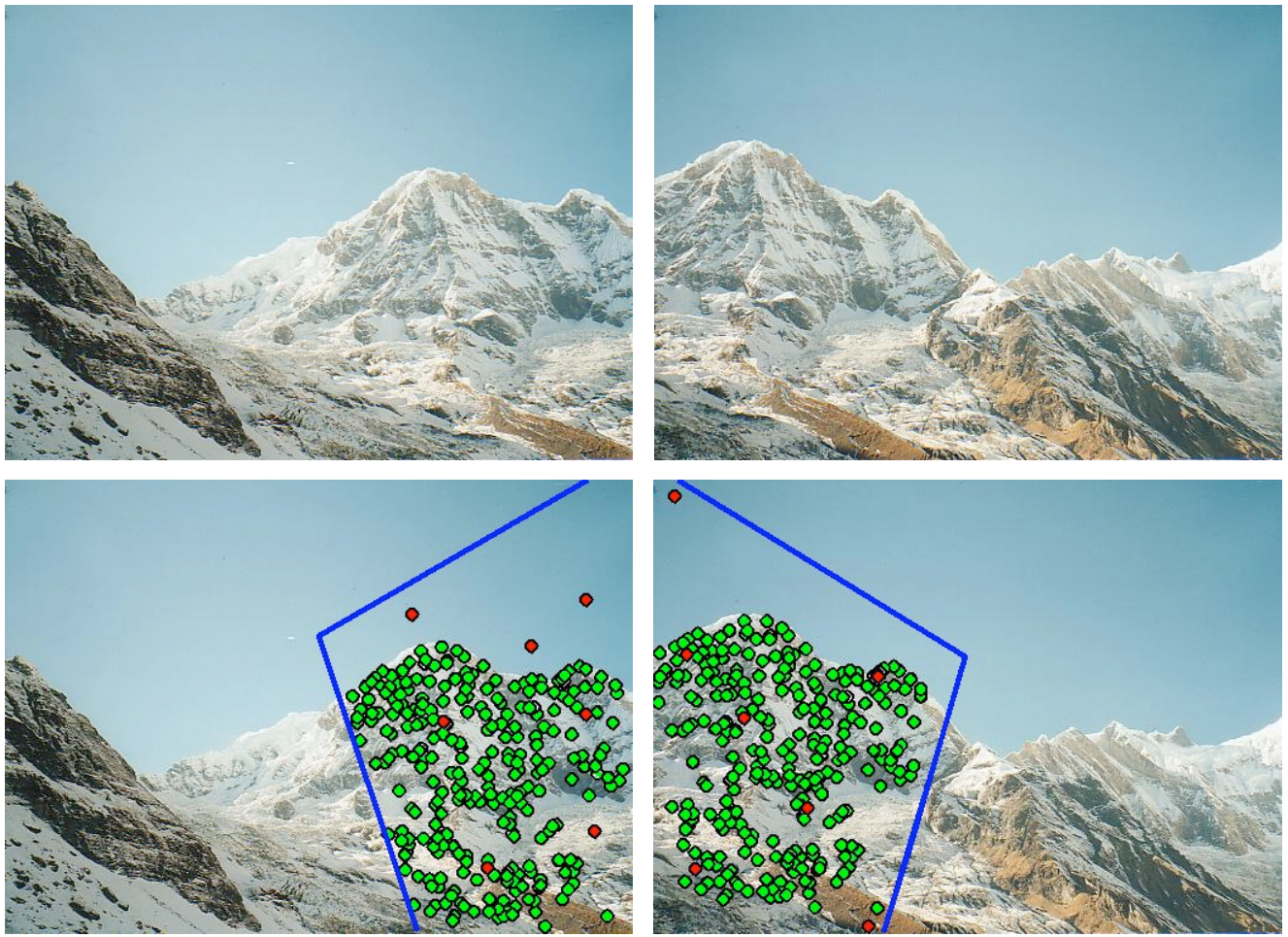


Image Mosaics

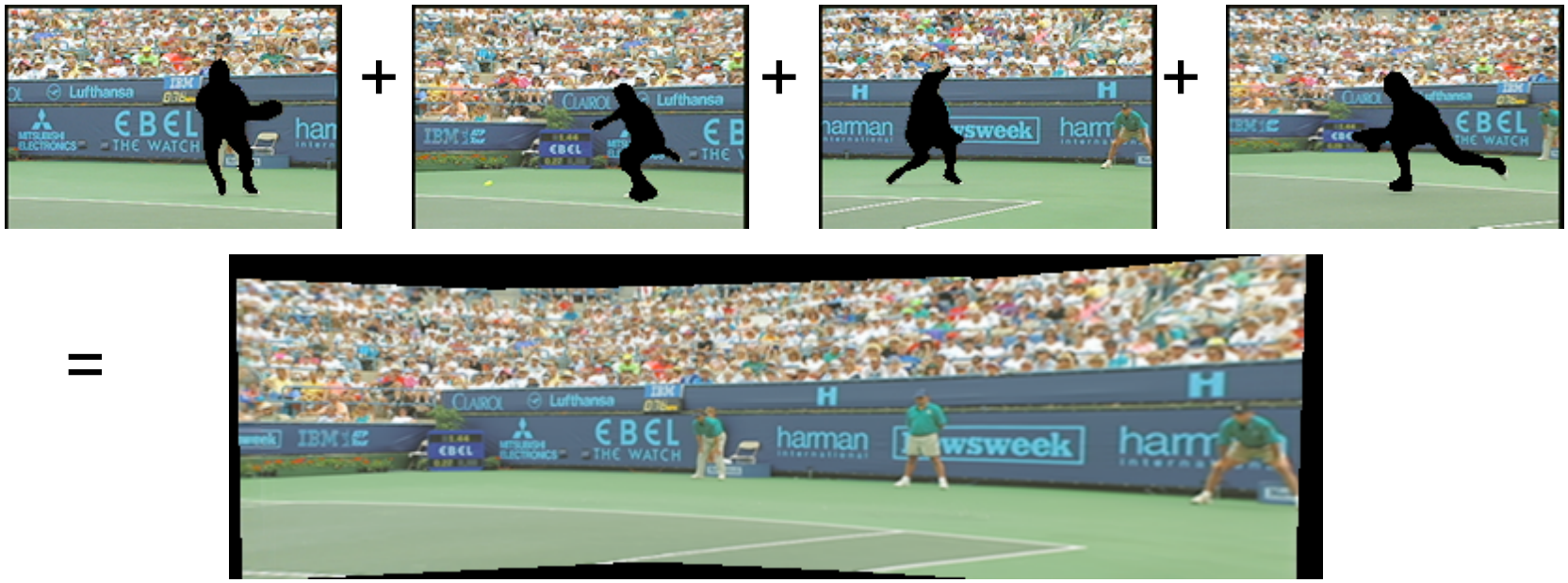
[Szeliski & Shum, SIGGRAPH'97]

Image Mosaics (stitching)

Blend together several overlapping images into one seamless *mosaic* (composite)

Mosaics for Video Coding

Convert masked images into a background sprite for content-based coding



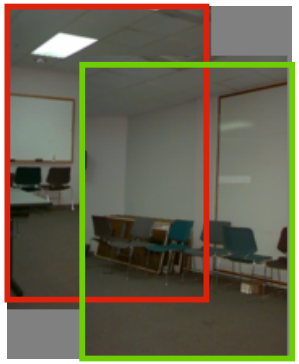
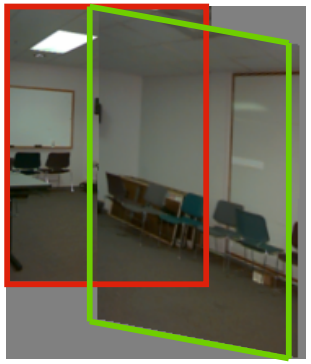
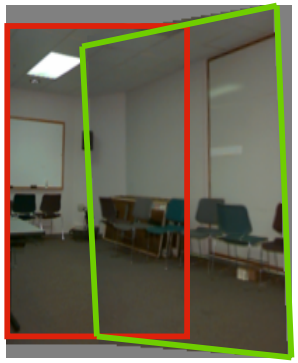
Rotation _ Homography

What happens when we rotate an image?

- translation?
- scale?
- affine?
- perspective?

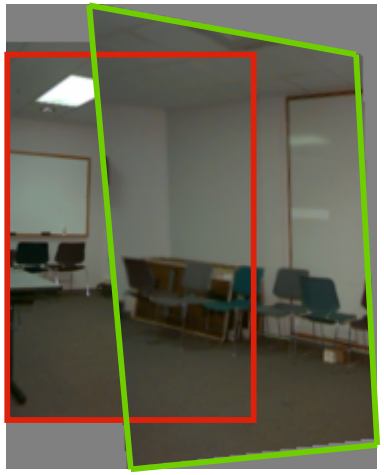
Motion models

- (See demo)

Translation	Affine	Perspective	3D rotation
			
2 unknowns	6 unknowns	8 unknowns	3 unknowns

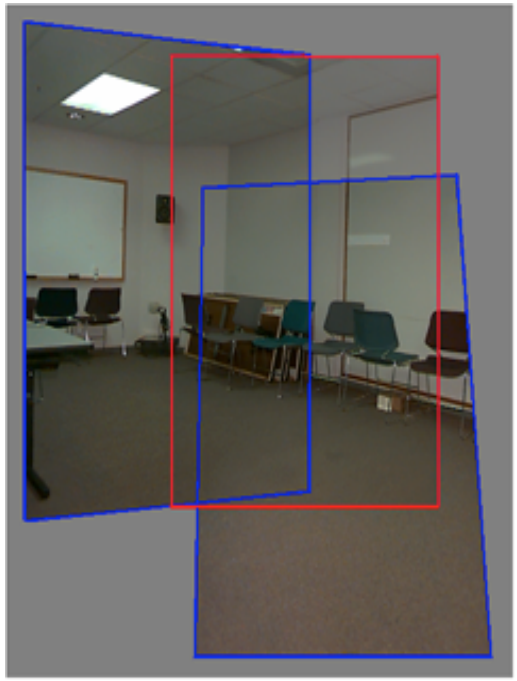
Plane perspective mosaics

- 8-parameter generalization of affine motion
  - works for pure rotation or planar surfaces
- Limitations:
  - local minima
  - slow convergence
  - difficult to control interactively

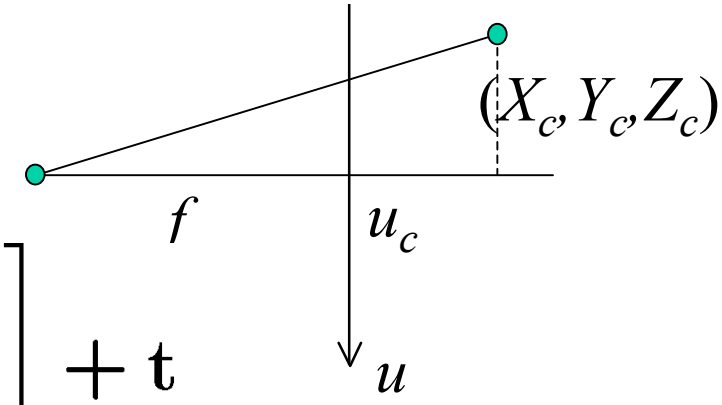


Rotational mosaics

- Directly optimize rotation and focal length
- Advantages:
  - ability to build full-view panoramas
  - easier to control interactively
  - more stable and accurate estimates



3D _ 2D Perspective Projection



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Rotational mosaic

Projection equations

1. Project from image to 3D ray

   $(x_0,y_0,z_0) \quad = (u_0\text{-}u_c, v_0\text{-}v_c, f)$

2. Rotate the ray by camera motion

   $(x_1,y_1,z_1) \quad = \boldsymbol{R}_{01}\,(x_0,y_0,z_0)$

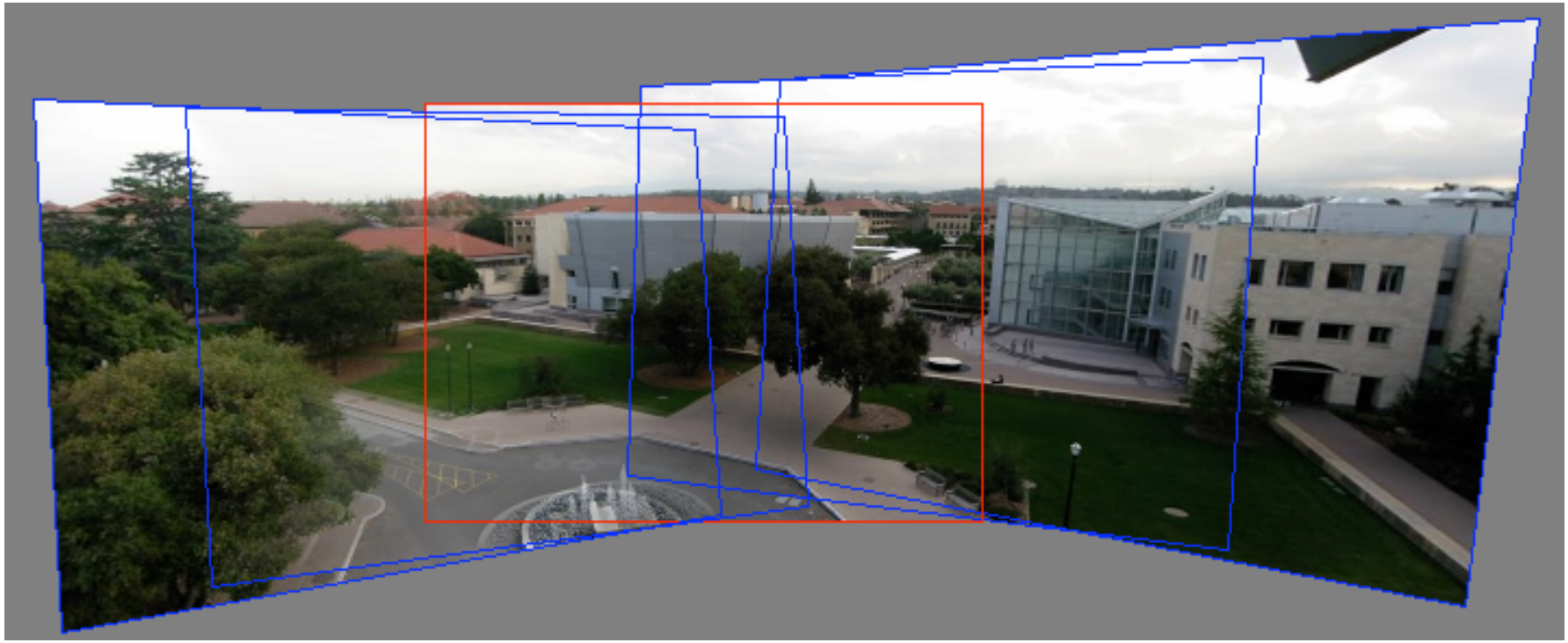
3. Project back into new (source) image

   $(u_1,v_1) \quad = (fx_1/z_1+u_c, fy_1/z_1+v_c)$

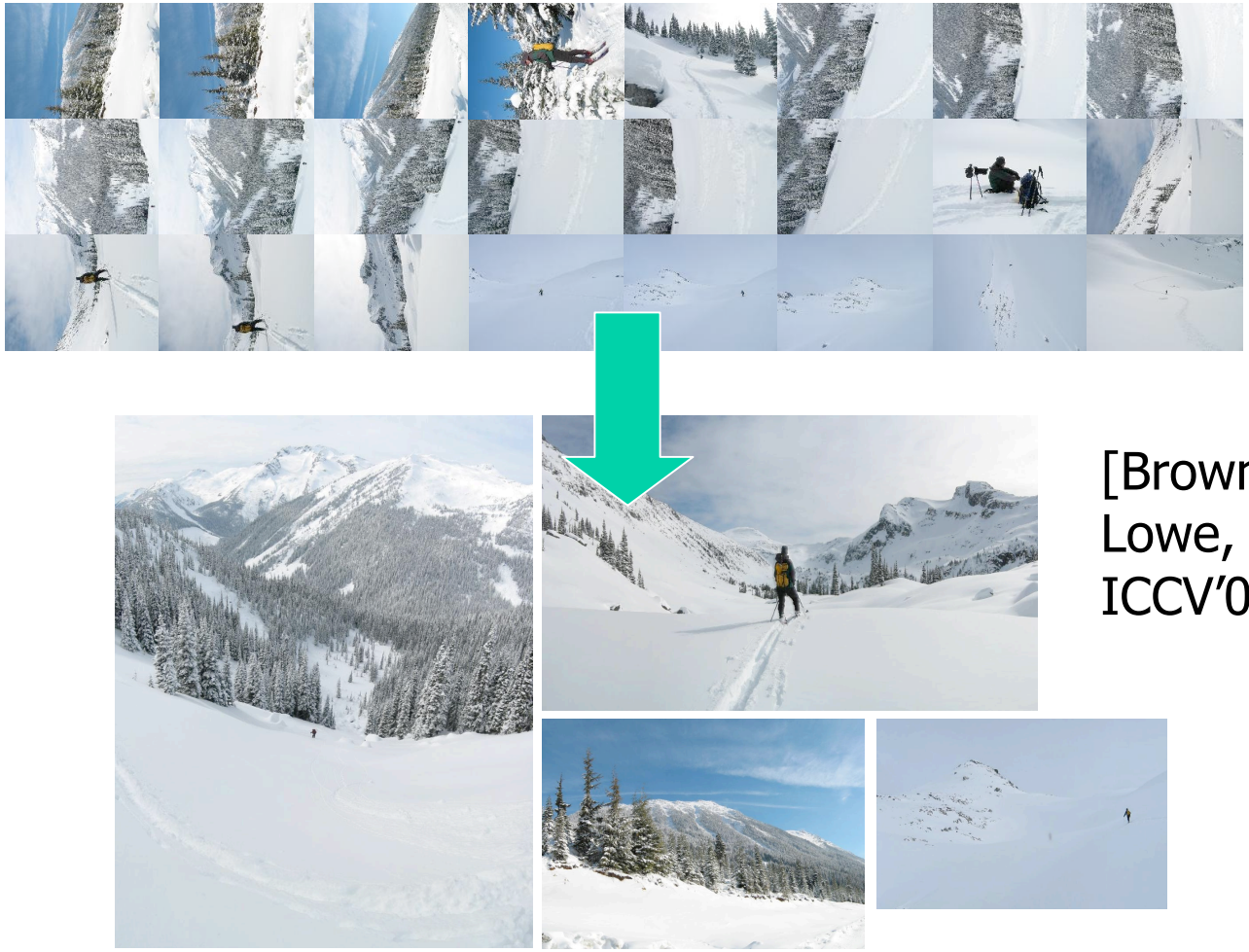
Establishing correspondences

1. Direct method:
   - Use generalization of affine motion model [Szeliski & Shum ’97]
2. Feature-based method
   - Use Shi-Tomasi tracker after initial rough alignment
     [Lowe ICCV’99; Schmid ICCV’98]
   - Compute $\boldsymbol{R}$ from correspondences (absolute orientation)

Stitching demo

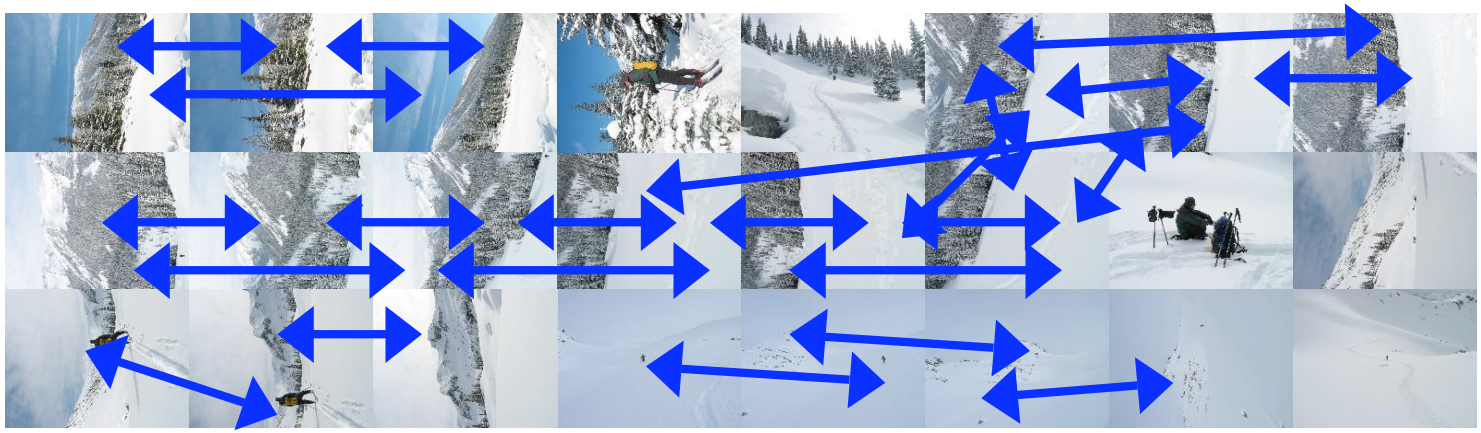


Recognizing Panoramas

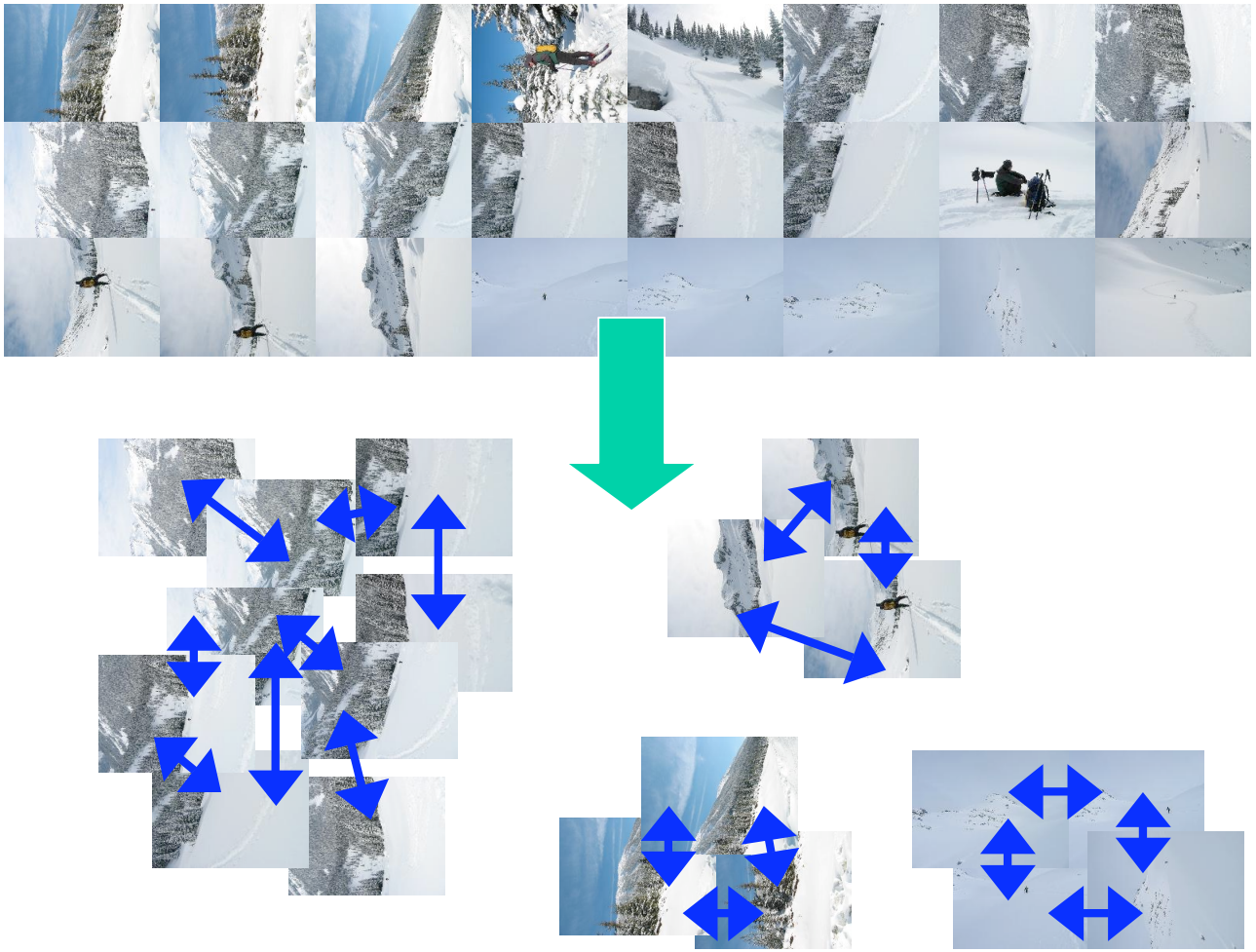


[Brown & Lowe, ICCV'03]

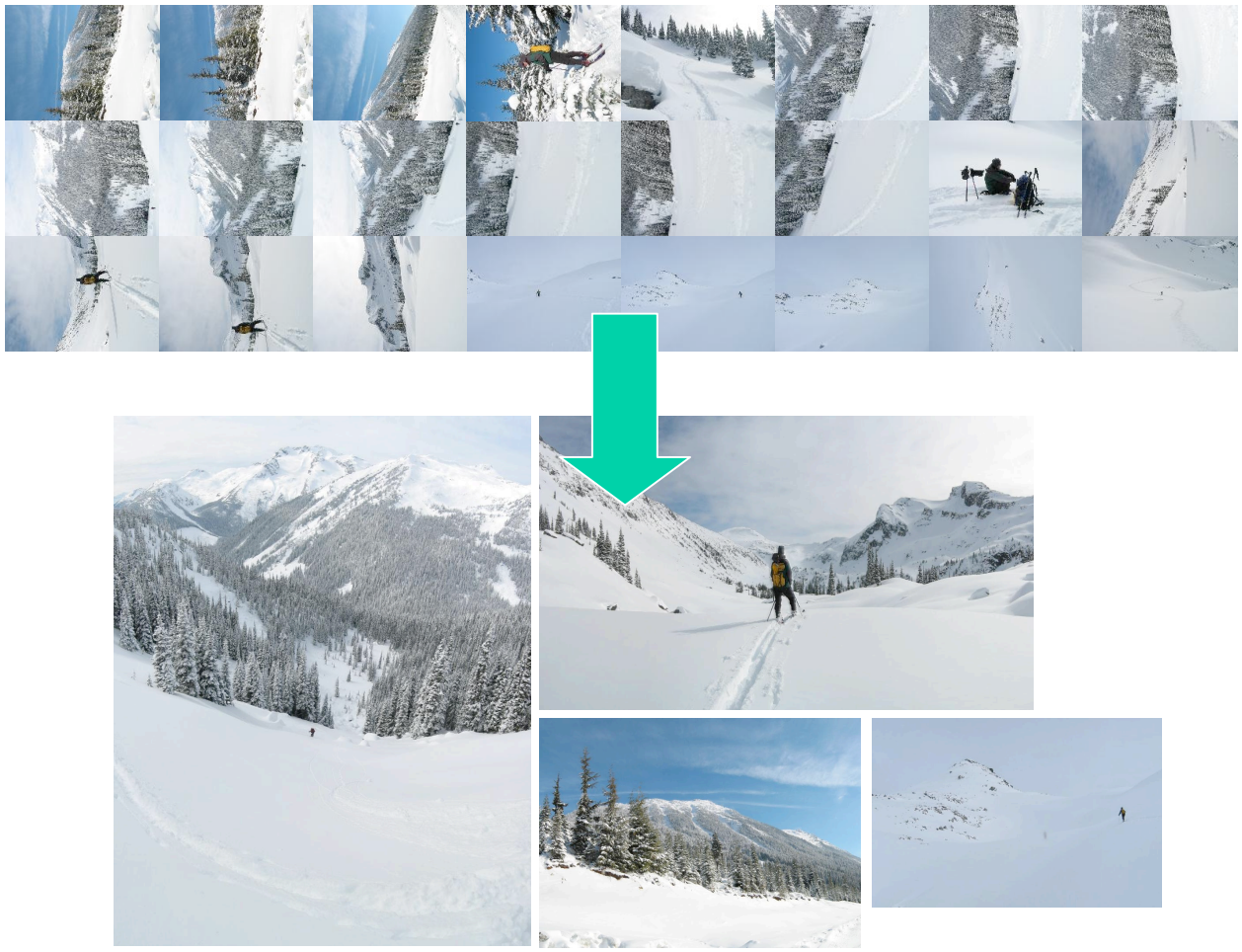
Finding the panoramas



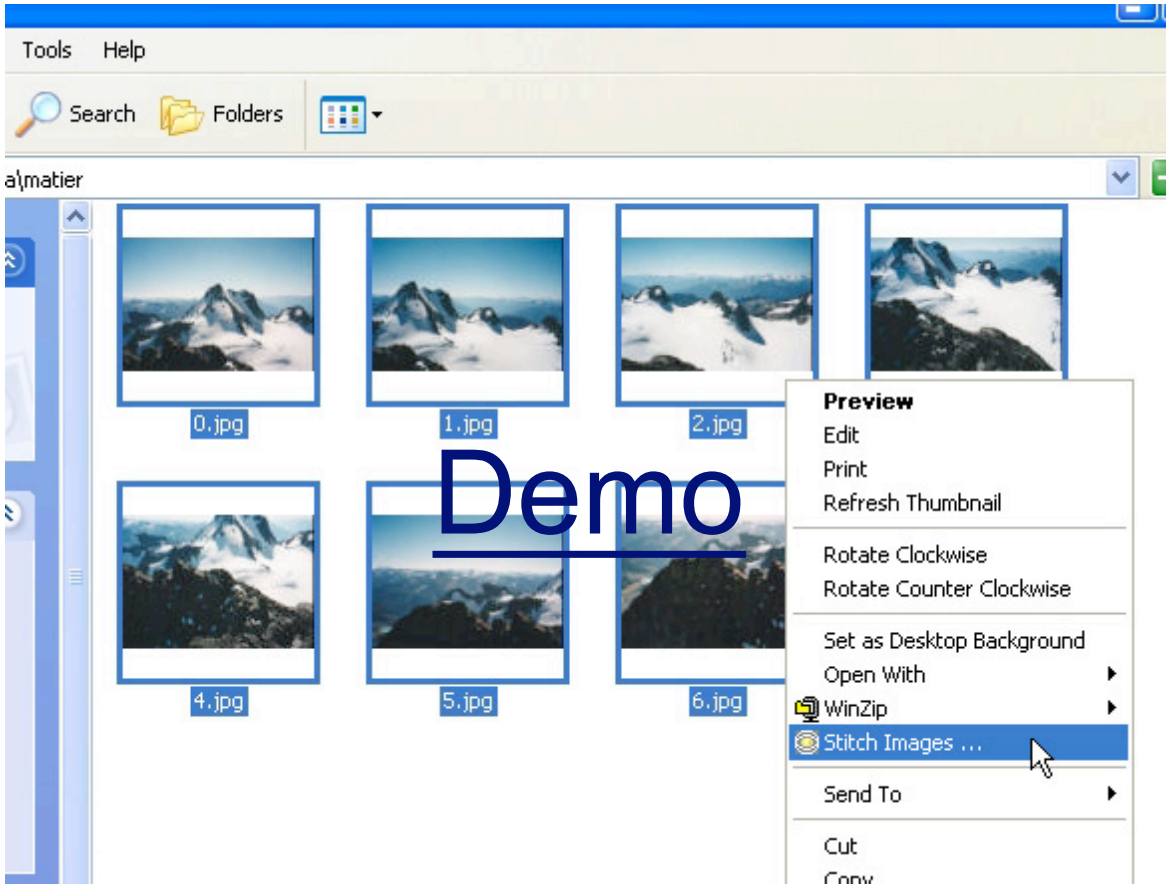
Finding the panoramas



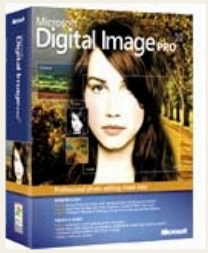
Finding the panoramas



Fully automated 2D stitching…



…in a real product!



Microsoft® Digital Image Pro 10

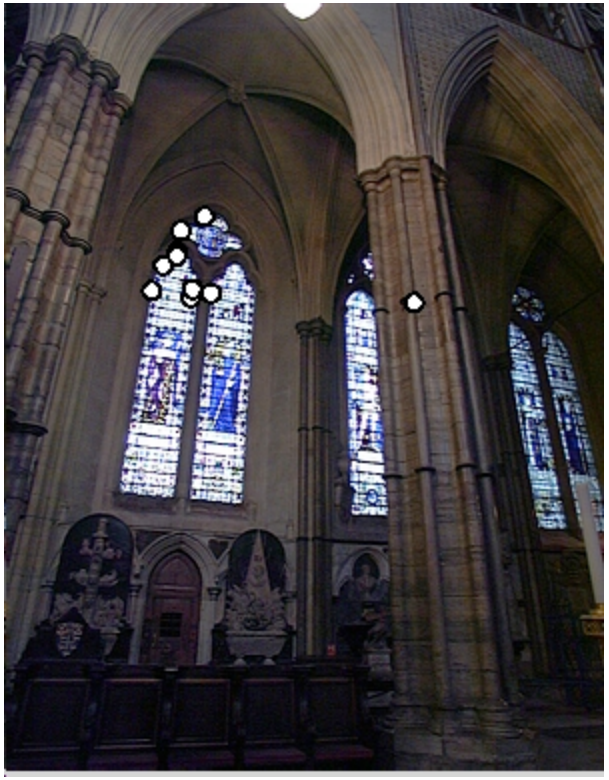
Overview | System Requirements

Microsoft Digital Image Pro 10 makes digital photos look more professional by giving you powerful photo editing tools that are easy enough for anyone to master. Advanced filters and professional techniques make artistic, creative results simple to achieve.

Features

- **New color management tools**
  Color management and exposure tools offer precise color enhancement and correction.
- **Improved Fix Red-Eye tool**
  Eliminate red eye from your photos in a snap.
- **New Panoramic Stitching**
  Create a panoramic view instantly by connecting multiple photos into one seamless image.
- **Share your photo creations**
  Share photos through e-mail, order prints online, or use Auto Collage to print multiple sizes and multiple photos on one page.
- **Create compelling photos**
  Get creative with photo filters, effects and designer templates for cards, calendars, and more.

Matching Mistakes



Matching Mistakes

Matching Mistakes

Moving objects: large areas of disagreement

Matching Mistakes

Accidental alignment

- repeated / similar regions

Failed alignments

- moving objects / parallax
- low overlap
- “feature-less” regions (more variety?)

No 100% reliable algorithm?

How to fix these?

Tune the feature detector

Tune the feature matcher (cost metric)

Tune the RANSAC stage (motion model)

Tune the verification stage

Use “higher-level” knowledge

- e.g., typical camera motions

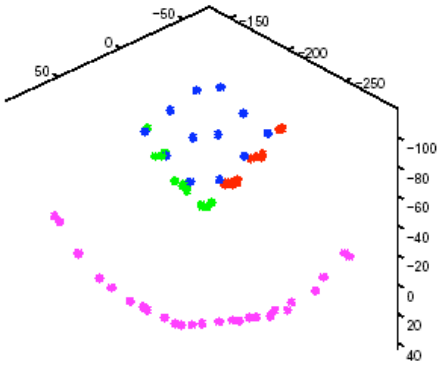
_ Sounds like a big “learning” problem

- Need a large training/test data set (panoramas)

Structure (from) Motion

Structure [from] Motion

Given a set of feature tracks,
estimate the 3D structure and
3D (camera) motion.

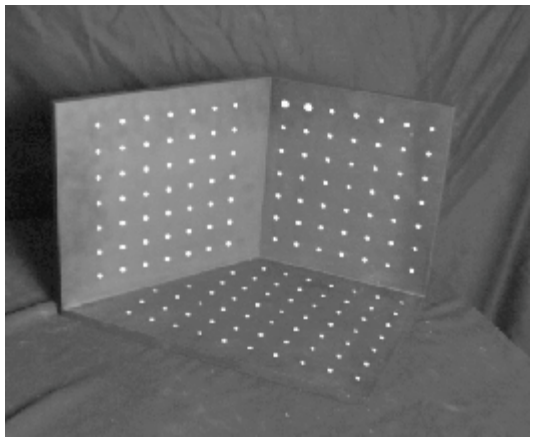


Camera calibration

Determine camera parameters from *known* 3D points or calibration object(s)

1. *internal* or *intrinsic* parameters such as focal length, optical center, aspect ratio:
   what kind of camera?
2. *external* or *extrinsic* (pose) parameters:
   where is the camera?

How can we do this?

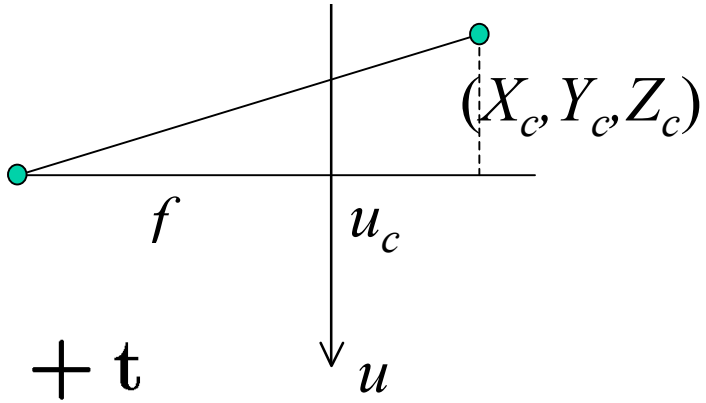


Camera calibration – approaches

Possible approaches:

1. linear regression (least squares)
2. non-linear optimization
3. vanishing points
4. multiple planar patterns
5. panoramas (rotational motion)

Image formation equations



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Calibration matrix

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{K}\ \mathbf{X}_c$$

Is this form of K good enough?

- non-square pixels (digital video)
- skew
- radial distortion

$$\mathbf{K} = \begin{bmatrix} fa & s & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix}$$

Camera matrix

Fold *intrinsic* calibration matrix **K** and *extrinsic* pose parameters (**R**,**t**) together into a *camera matrix*

$$\mathbf{M} = \mathbf{K}\,[\mathbf{R} \mid \mathbf{t}\,]$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

(put 1 in lower r.h. corner for 11 d.o.f.)

Camera matrix calibration

Directly estimate 11 unknowns in the **M** matrix using known 3D points (X_i, Y_i, Z_i) and measured feature positions (u_i, v_i)

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

Camera matrix calibration

Linear regression:

- Bring denominator over, solve set of (over-determined) linear equations. How?

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

- Least squares (pseudo-inverse)
- Is this good enough?

Optimal estimation

Feature measurement equations

$$u_i = f(\mathbf{M}, \mathbf{x}_i) + n_i = \hat{u}_i + n_i, \quad n_i \sim N(0, \sigma)$$
$$v_i = g(\mathbf{M}, \mathbf{x}_i) + m_i = \hat{v}_i + m_i, \quad m_i \sim N(0, \sigma)$$

Likelihood of $\boldsymbol{M}$ given $\{(u_i, v_i)\}$

$$L = \prod_i p(u_i|\hat{u}_i) p(v_i|\hat{v}_i)$$
$$= \prod_i e^{-(u_i - \hat{u}_i)^2/\sigma^2} e^{-(v_i - \hat{v}_i)^2/\sigma^2}$$

Optimal estimation

Log likelihood of ***M*** given $\{(u_i, v_i)\}$

$$C = -\log L = \sum_i (u_i - \hat{u}_i)^2/\sigma_i^2 + (v_i - \hat{v}_i)^2/\sigma_i^2$$

How do we minimize *C*?

Non-linear regression (least squares), because $\hat{u}_i$ and v_i are non-linear functions of ***M***

Levenberg-Marquardt

Iterative non-linear least squares [Press'92]

- Linearize measurement equations

$$\hat{u}_i = f(\mathbf{m}, \mathbf{x}_i) + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m}$$

$$\hat{v}_i = g(\mathbf{m}, \mathbf{x}_i) + \frac{\partial g}{\partial \mathbf{m}} \Delta \mathbf{m}$$

- Substitute into log-likelihood equation: quadratic cost function in **Δm**

$$\sum_i \sigma_i^{-2} (\hat{u}_i - u_i + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m})^2 + \cdots$$

Levenberg-Marquardt

Iterative non-linear least squares [Press'92]

- Solve for minimum $\frac{\partial C}{\partial \mathbf{m}} = 0$

$$\mathbf{A}\Delta\mathbf{m} = \mathbf{b}$$

Hessian $$\mathbf{A} = \left[\sum_i \sigma_i^{-2} \frac{\partial f}{\partial \mathbf{m}} \left(\frac{\partial f}{\partial \mathbf{m}}\right)^T + \cdots\right]$$

error: $$\mathbf{b} = \left[\sum_i \sigma_i^{-2} \frac{\partial f}{\partial \mathbf{m}} (u_i - \hat{u}_i) + \cdots\right]$$

Levenberg-Marquardt

What if it doesn't converge?

- Multiply diagonal by $(1 + \lambda)$, increase λ until it does
- Halve the step size $\Delta\mathbf{m}$
- Use line search
- Other ideas?

Uncertainty analysis: covariance $\Sigma = A^{-1}$

Is *maximum* likelihood the best idea?

How to start in vicinity of global minimum?

Camera matrix calibration

Advantages:

- very simple to formulate and solve
- can recover **K** [**R** | **t**] from **M** using QR decomposition [Golub & VanLoan 96]

Disadvantages:

- doesn't compute internal parameters
- more unknowns than true degrees of freedom
- need a separate camera matrix for each new view

Separate intrinsics / extrinsics

New feature measurement equations

$$\begin{aligned} \hat{u}_{ij} &= f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \\ \hat{v}_{ij} &= g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \end{aligned}$$

Use non-linear minimization

Standard technique in photogrammetry, computer vision, computer graphics

- [Tsai 87] – also estimates κ_1 (freeware @ CMU)
  http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/v-source.html
- [Bogart 91] – *View Correlation*

Separate intrinsics / extrinsics

How do we parameterize **R** and **_R**?

- Euler angles: bad idea
- quaternions: 4-vectors on unit sphere
- use incremental rotation **R**(**I** + Δ**R**)

$$\Delta\mathbf{R} = [\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

- update with Rodriguez formula

$$\mathbf{R}(\hat{\mathbf{n}}, \theta) = \mathbf{I} + \sin\theta\,[\hat{\mathbf{n}}]_\times + (1 - \cos\theta)\,[\hat{\mathbf{n}}]_\times^2\,, \qquad \boldsymbol{\omega} = \theta\hat{\mathbf{n}}$$

Intrinsic/extrinsic calibration

Advantages:

- can solve for more than one camera pose at a time
- potentially fewer degrees of freedom

Disadvantages:

- more complex update rules
- need a good initialization (recover **K** [**R** | **t**] from **M**)

Pose estimation

Once the internal camera parameters are known, can compute camera pose

$$\begin{aligned} \hat{u}_{ij} &= f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \\ \hat{v}_{ij} &= g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \end{aligned}$$

[Tsai87] [Bogart91]

Application: superimpose 3D graphics onto video

How do we initialize (***R***,***t***)?

Pose estimation

Initialization techniques:

- vanishing points [Caprile 90]
- planar pattern [Zhang 99]
- *Through-the-Lens Camera Control* [Gleicher92]: differential update
- 3+ point “linear methods”:

  [DeMenthon 95][Quan 99][Ameller 00]

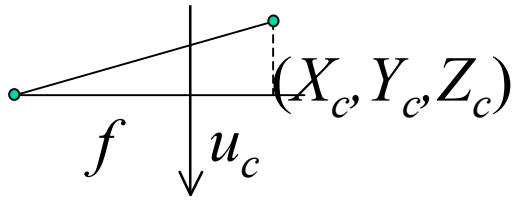
Pose estimation

Solve orthographic problem, iterate

[DeMenthon 95]

Use inter-point distance constraints

[Quan 99][Ameller 00]



$$\mathbf{u}_i = \begin{bmatrix} u_i - u_c \\ v_i - v_c \\ f \end{bmatrix}, \quad x_i = \|\mathbf{X}_i\|$$

$$d_{ij}^2 = \|\mathbf{X}_i - \mathbf{X}_j\| = x_i^2 + x_j^2 - 2x_i x_j cos\theta_{ij}$$

Solve set of polynomial equations in x_i^{2p}

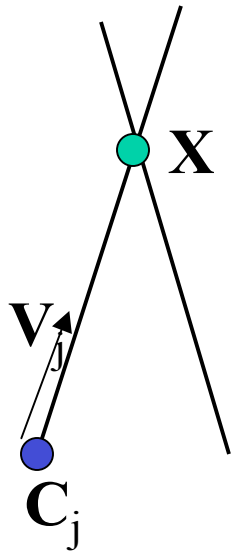
Triangulation

Problem: Given some points in *correspondence* across two or more images (taken from calibrated cameras), $\{(u_j,v_j)\}$, compute the 3D location $\mathbf{X}$

Triangulation

Method I: intersect viewing rays in 3D, minimize:

$$\arg\min_{\mathbf{X}} \sum_j \|\mathbf{C}_j + s\mathbf{V}_j - \mathbf{X}\|$$



- **X** is the unknown 3D point
- $\mathbf{C}_j$ is the optical center of camera j
- $\mathbf{V}_j$ is the *viewing ray* for pixel (u_j, v_j)
- s_j is unknown distance along $\mathbf{V}_j$

Advantage: geometrically intuitive

Triangulation

Method II: solve linear equations in **X**

- advantage: very simple

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

Method III: non-linear minimization

- advantage: most accurate (image plane error)

Structure from Motion

Structure from motion

Given many points in *correspondence* across several images, $\{(u_{ij}, v_{ij})\}$, simultaneously compute the 3D location $\mathbf{x}_i$ and camera (or *motion*) parameters ($\mathbf{K}$, $\mathbf{R}_j$, $\mathbf{t}_j$)

$$\begin{aligned} \hat{u}_{ij} &= f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \\ \hat{v}_{ij} &= g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \end{aligned}$$

Two main variants: calibrated, and uncalibrated (sometimes associated with Euclidean and projective reconstructions)

Structure from motion

$$\hat{u}_{ij} = f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)$$
$$\hat{v}_{ij} = g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)$$

How many points do we need to match?

- 2 frames:

  ($\boldsymbol{R}$,$\boldsymbol{t}$): 5 dof + $3n$ point locations ≤

  $4n$ point measurements ⇒

  $n \geq 5$

- k frames:

  $6(k–1)\text{-}1 + 3n \leq 2kn$

- always want to use many more

Two-frame methods

Two main variants:

1. Calibrated: "Essential matrix" $\boldsymbol{E}$
   use ray directions $(\boldsymbol{x}_i, \boldsymbol{x}_i')$
2. Uncalibrated: "Fundamental matrix" $\boldsymbol{F}$

[Hartley & Zisserman 2000]

Essential matrix

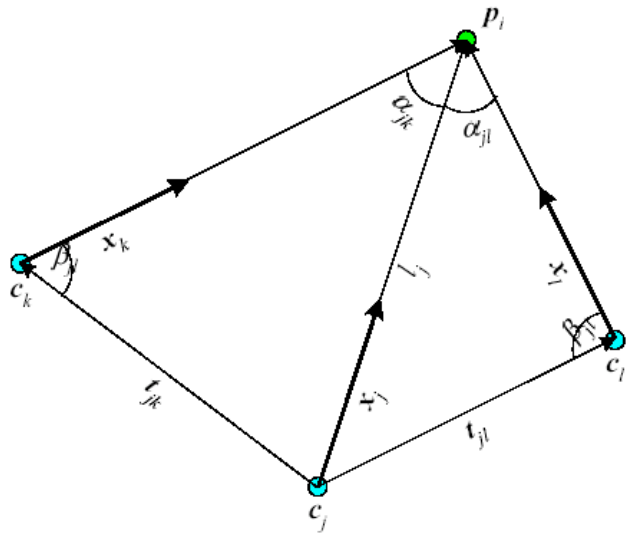
Co-planarity constraint:

$$\boldsymbol{x'} \approx \boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$$

$$[\boldsymbol{t}]_\times\,\boldsymbol{x'} \approx [\boldsymbol{t}]_\times\,\boldsymbol{R}\,\boldsymbol{x}$$

$$\boldsymbol{x'}^{T}\,[\boldsymbol{t}]_\times\,\boldsymbol{x'} \approx \boldsymbol{x'}^{T}\,[\boldsymbol{t}]_\times\,\boldsymbol{R}\,\boldsymbol{x}$$

$$\boldsymbol{x'}^{T}\,\boldsymbol{E}\,\boldsymbol{x} = 0 \quad \text{with } \boldsymbol{E} = [\boldsymbol{t}]_\times\,\boldsymbol{R}$$



- Solve for $\boldsymbol{E}$ using least squares (SVD)
- $\boldsymbol{t}$ is the least singular vector of $\boldsymbol{E}$
- $\boldsymbol{R}$ obtained from the other two s.v.s

Fundamental matrix

Camera calibrations are unknown

$\boldsymbol{x}' \boldsymbol{F} \boldsymbol{x} = 0$ with $\boldsymbol{F} = [\boldsymbol{e}]_\times \boldsymbol{H} = \boldsymbol{K}'[\boldsymbol{t}]_\times \boldsymbol{R}\, \boldsymbol{K}^{-1}$

- Solve for $\boldsymbol{F}$ using least squares (SVD)
  - re-scale $(\boldsymbol{x}_i, \boldsymbol{x}_i')$ so that $|\boldsymbol{x}_i| \approx 1/2$ [Hartley]
- $\boldsymbol{e}$ (epipole) is *still* the least singular vector of $\boldsymbol{F}$
- $\boldsymbol{H}$ obtained from the other two s.v.s
- "plane + parallax" (projective) reconstruction
- use self-calibration to determine $\boldsymbol{K}$ [Pollefeys]

Multi-frame Structure from Motion

Factorization

SVD: $\boldsymbol{W} = \boldsymbol{R}_{2F\times 3}\,\boldsymbol{S}_{3\times P} = \boldsymbol{U}\_\,\boldsymbol{V} = \boldsymbol{U}'\,\boldsymbol{V}'$

Make $\boldsymbol{R}$ orthogonal

$$\boldsymbol{R} = \boldsymbol{Q}\boldsymbol{U}', \quad \boldsymbol{S} = \boldsymbol{Q}^{-1}\boldsymbol{V}'$$

$$\boldsymbol{i}_f^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{i}_f = 1, \boldsymbol{j}_f^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{j}_f = 1, \boldsymbol{i}_f^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{j}_f = 0$$

Two step "linear" algorithm:

1. estimate $\boldsymbol{A}=\boldsymbol{Q}^T\boldsymbol{Q}$ using least squares
2. estimate $\boldsymbol{Q}$ using SVD: $\boldsymbol{W} = \boldsymbol{U}\_\,\boldsymbol{U}^T$

Better than Newton iteration [Lorenzo]

Bundle Adjustment

$$\begin{aligned} \hat{u}_{ij} &= f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \\ \hat{v}_{ij} &= g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \end{aligned}$$

What makes this non-linear minimization hard?

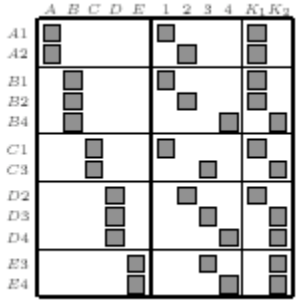
- many more parameters: potentially slow
- poorer conditioning (high correlation)
- potentially lots of outliers
- gauge (coordinate) freedom

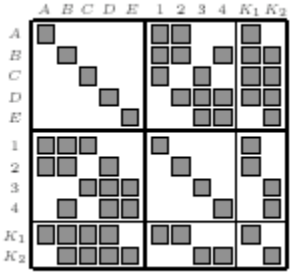
Lots of parameters: sparsity

$$\begin{aligned}\hat{u}_{ij} &= f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \\ \hat{v}_{ij} &= g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)\end{aligned}$$

Only a few entries in Jacobian are non-zero

$$\frac{\partial \hat{u}_{ij}}{\partial \mathbf{K}}, \quad \frac{\partial \hat{u}_{ij}}{\partial \mathbf{R}_j}, \quad \frac{\partial \hat{u}_{ij}}{\partial \mathbf{t}_j}, \quad \frac{\partial \hat{u}_{ij}}{\partial \mathbf{x}_i},$$

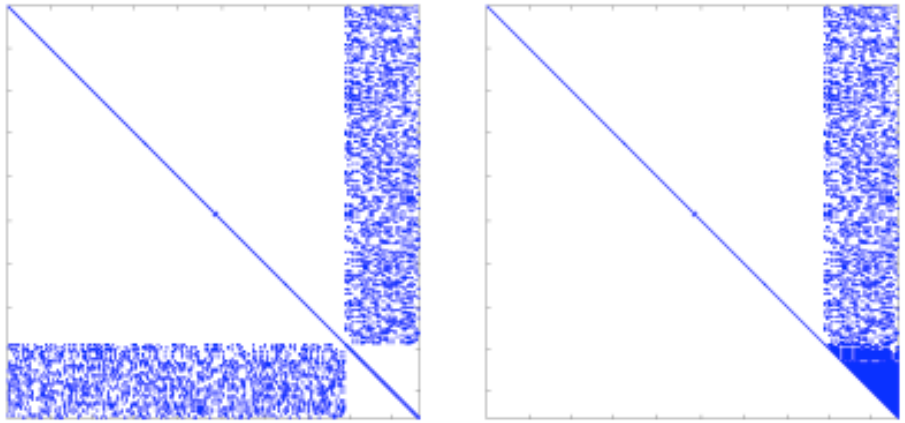
J = 

H = 

Sparse Cholesky (skyline)

First used in finite element analysis

Applied to SfM by [Szeliski & Kang 1994]

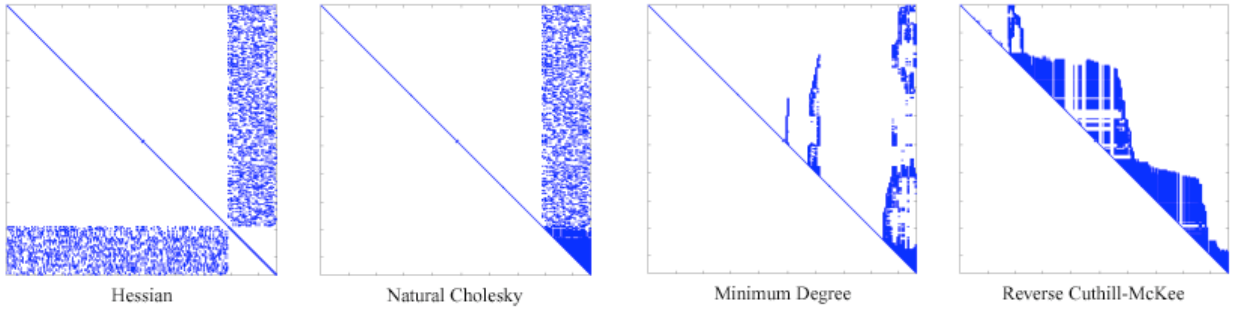


structure | motion fill-in

Conditioning and gauge freedom

Poor conditioning:

- use 2nd order method
- use Cholesky decomposition



Gauge freedom

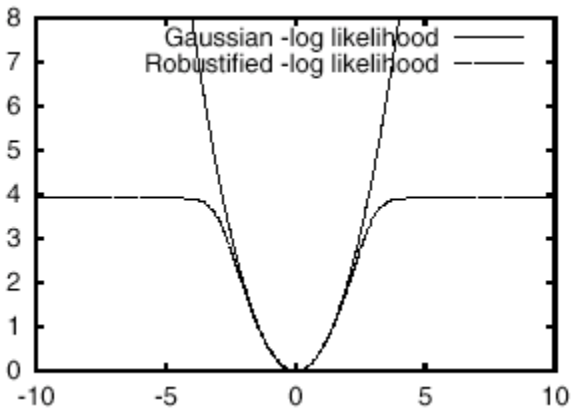
- fix certain parameters (orientation) *or*
- zero out last few rows in Cholesky decomposition

Robust error models

Outlier rejection

- use robust penalty applied to each set of joint measurements

$$\sum_i \sigma_i^{-2} \rho\left(\sqrt{(u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2}\right)$$



- for extremely bad data, use random sampling
  [RANSAC, Fischler & Bolles, CACM'81]
  [Least Median Sq., Rousseeuw 1984]

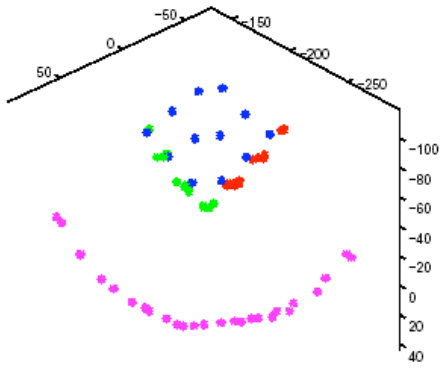
Structure from motion: limitations

Very difficult to reliably estimate *metric* structure and motion unless:

- large (*x* or *y*) rotation *or*
- large field of view and depth variation

Camera calibration important for Euclidean reconstructions

Need good feature tracker



Model reduction

Eliminate variables (frames, points)

- linear case: Kalman filter (optimal)
- non-linear case: iterated/extended KF
- Drew Steedly thesis: partition data into (quasi-)independent subsequences

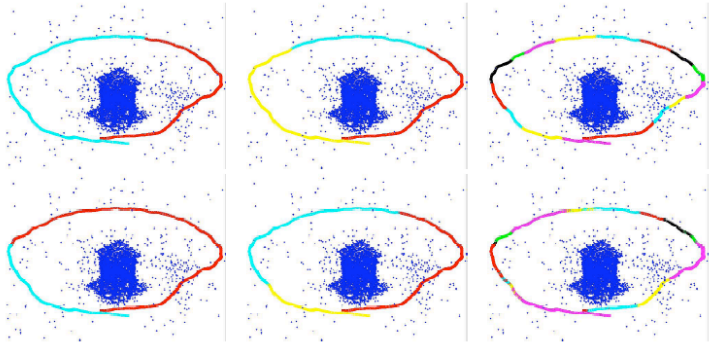


Figure 5: *Color coded 2, 3 and 16-way (left middle, and right) partitions of the 1106 image pillar sequence. The top shows the partitions generated using the Hessian, and the bottom shows the result of partitioning with the occupancy of the Hessian. The occupancy contains enough information to group correctly at the gross level but resorts to more random assignments at lower levels where cameras see the same points. The Hessian-based partitions maintain sharp boundaries throughout.*

Data association

Solve correspondence and reconstruction simultaneously using EM & MCMC [Frank Dellaert *et al*., ML'2003]

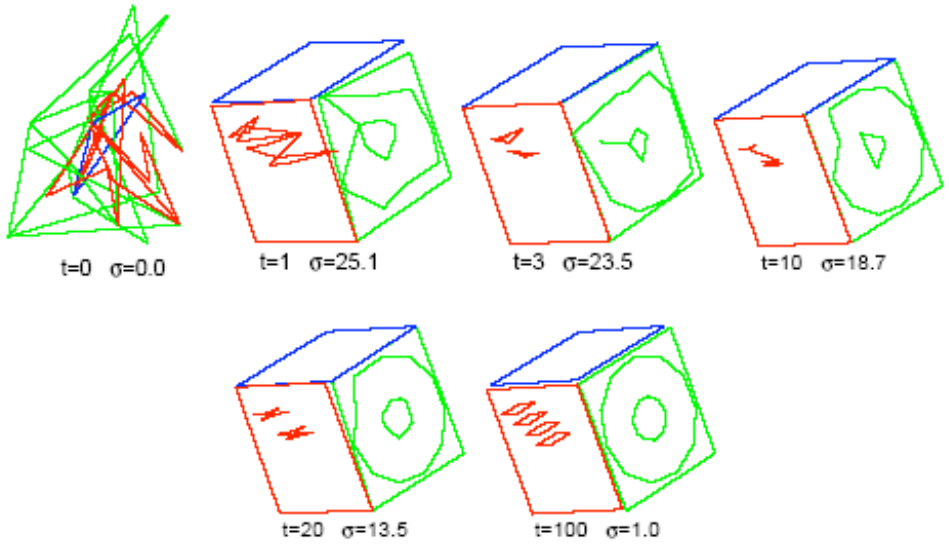


Figure 3. The structure estimate as initialized and at successive iterations t of the algorithm.

Stereo Reconstruction

Stereo Matching

Given two or more images of the same scene or object, compute a representation of its shape

What are some possible representations?

- depth maps
- volumetric models
- 3D surface models
- planar (or offset) layers

Stereo Matching

What are some possible algorithms?

- match “features” and interpolate
- match edges and interpolate
- match all pixels with windows (coarse-fine)
- use optimization:
  - iterative updating
  - dynamic programming
  - energy minimization (regularization, stochastic)
  - graph algorithms
  - loopy Belief Propagation

Stereo section outline

Image rectification

Matching criteria

Local algorithms (aggregation)

- iterative updating

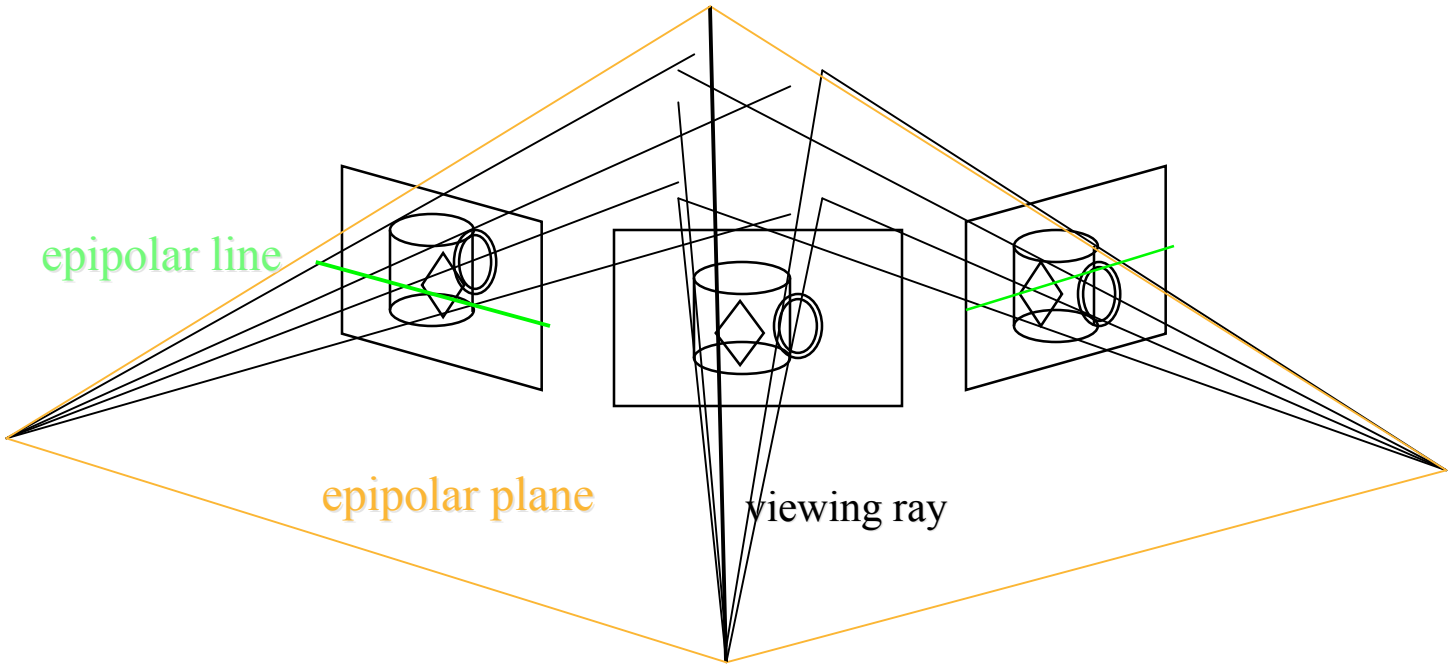
Optimization algorithms

- energy (cost) formulation & Markov Random Fields
- mean-field, stochastic, and graph algorithms

Evaluation

Stereo: epipolar geometry

Match features along epipolar lines



Stereo: epipolar geometry

for *two* images (or images with collinear camera centers), can find epipolar lines

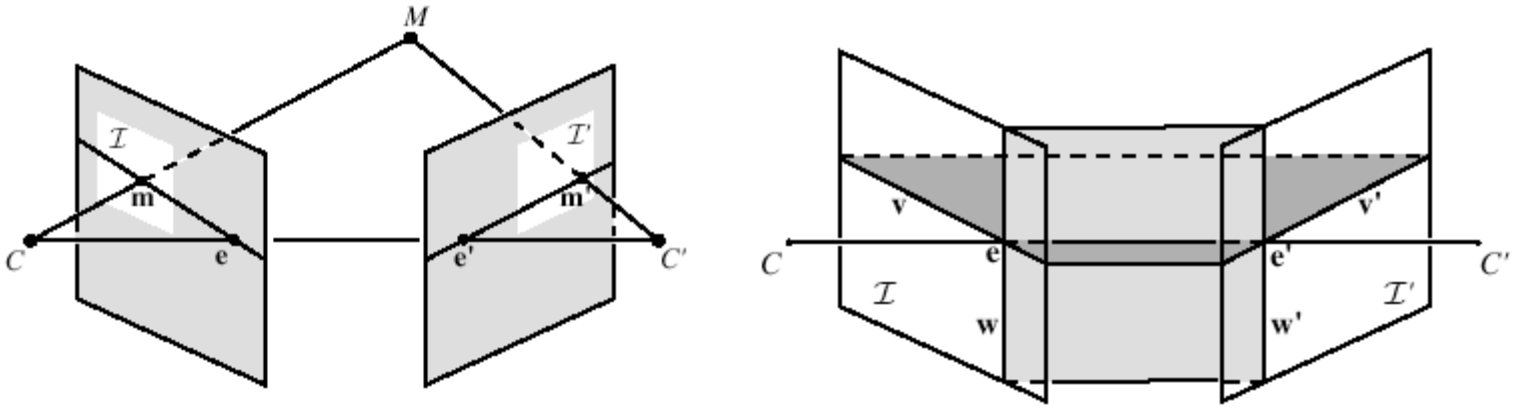
epipolar lines are the projection of the *pencil* of planes passing through the centers

Rectification: warping the input images (perspective transformation) so that epipolar lines are horizontal

Rectification

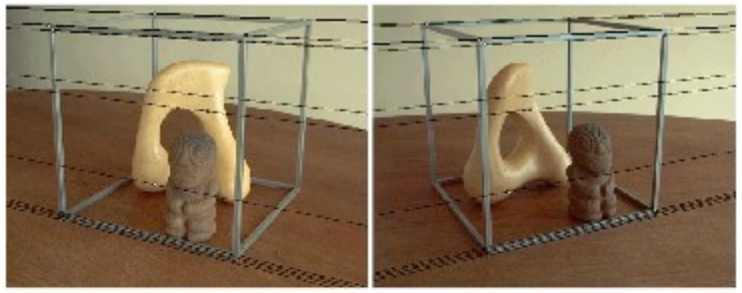
Project each image onto same plane, which is parallel to the epipole

Resample lines (and shear/stretch) to place lines in correspondence, and minimize distortion

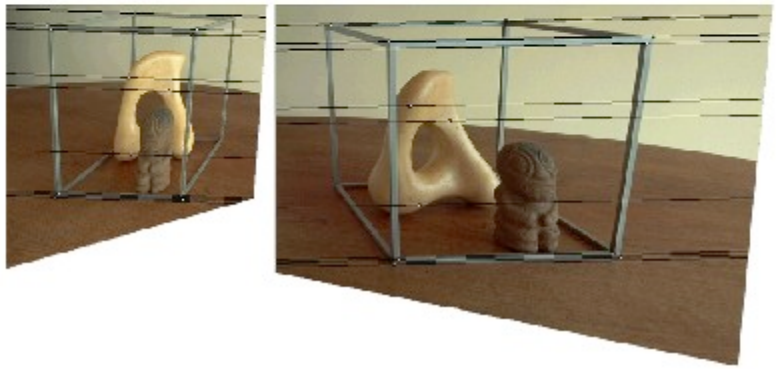


[Loop and Zhang, CVPR'99; … lots more]

Rectification



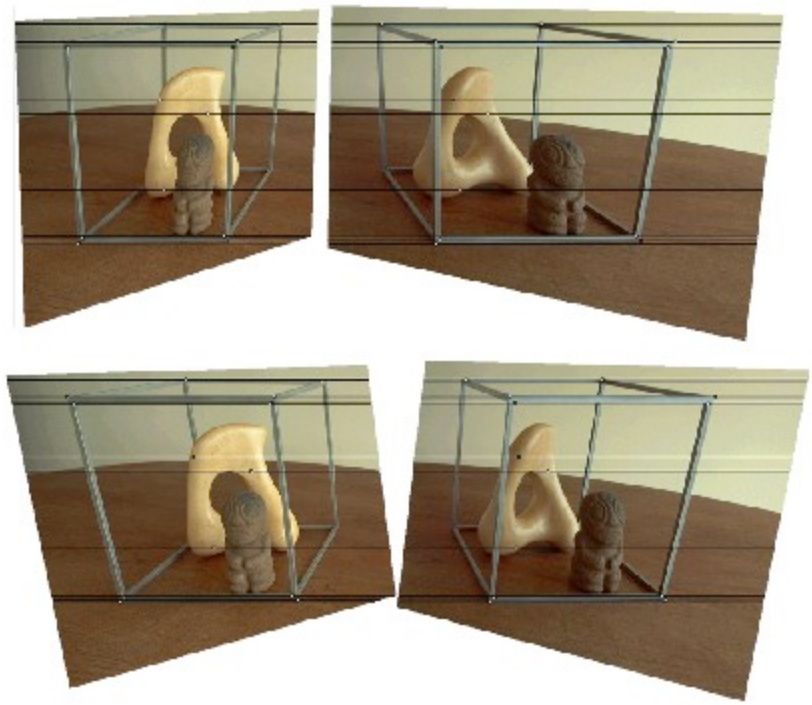
(a) Original image pair overlayed with several epipolar lines.



(b) Image pair transformed by the specialized projective mapping $\mathbf{H}_p$ and $\mathbf{H}'_p$. Note that the epipolar lines are now parallel to each other in each image.

BAD!

Rectification



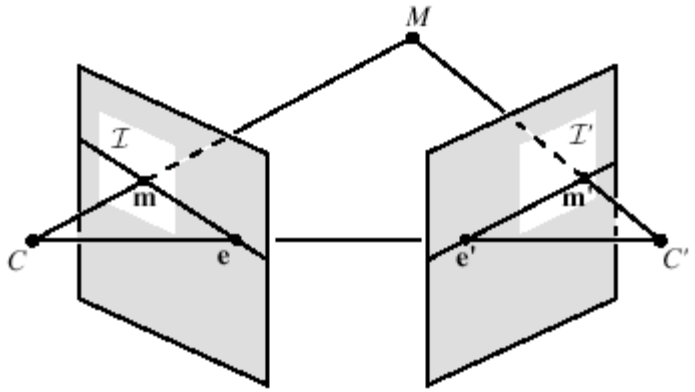
(c) Image pair transformed by the similarity $\mathbf{H}_r$ and $\mathbf{H}'_r$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).

(d) Final image rectification after shearing transform $\mathbf{H}_s$ and $\mathbf{H}'_s$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

GOOD!

Triangulation

Project a set (2 or more) image points into 3D and find the optimum position

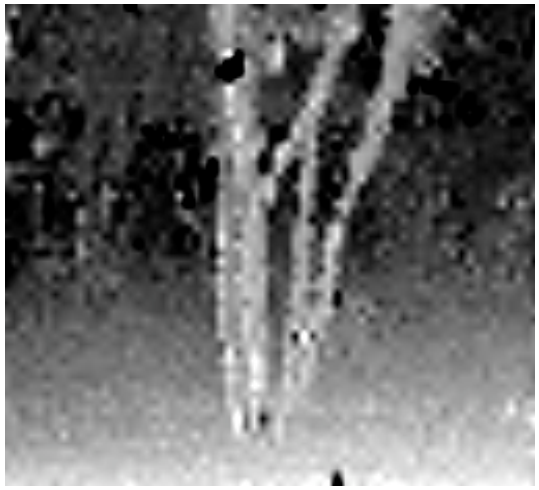


- Direct linear transform (DLT)
- Cubic equation solving [Hartley & Sturm 97]

Finding correspondences

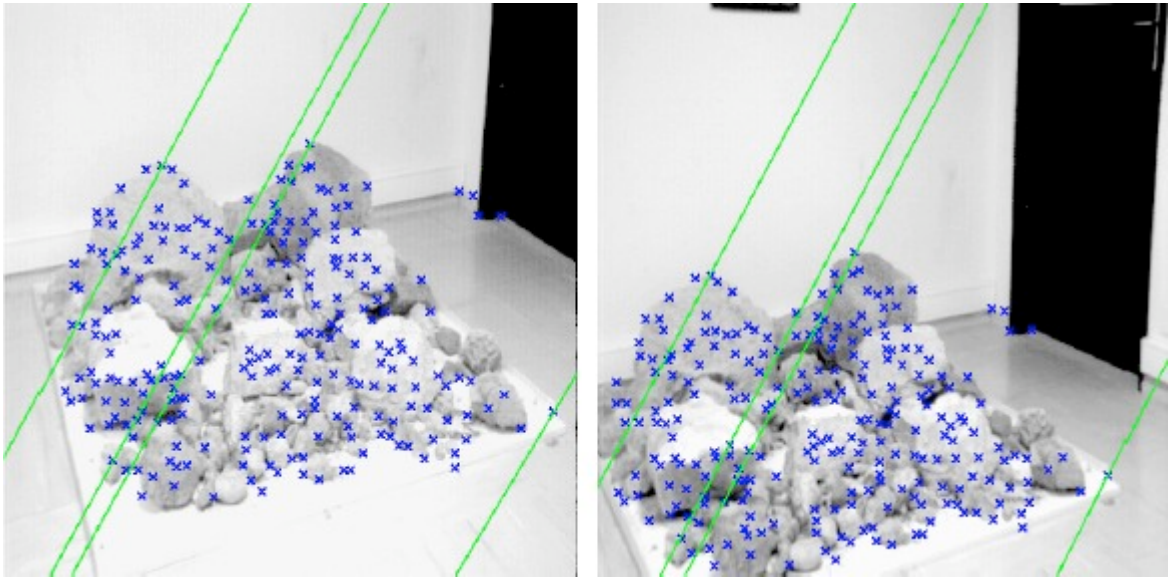
apply feature matching criterion (e.g., correlation or Lucas-Kanade) at *all* pixels simultaneously

search only over epipolar lines (many fewer candidate positions)



Feature-based stereo

Match “corner” (interest) points



Interpolate complete solution

Data interpolation

Given a sparse set of 3D points, how do we *interpolate* to a full 3D surface?

Scattered data interpolation [Nielson93]

- triangulate
- put onto a grid and fill (use pyramid?)
- place a *kernel function* over each data point
- minimize an energy function

Energy minimization

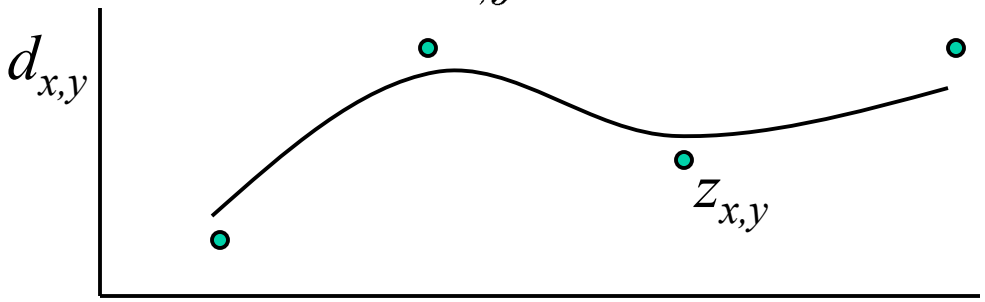
1-D example: approximating splines

$$E_{\text{total}}(\mathbf{d}) = E_{\text{data}}(\mathbf{d}) + \lambda E_{\text{smoothness}}(\mathbf{d})$$

$$E_{\text{data}}(\mathbf{d}) = \sum_{x,y} (d_{x,y} - z_{x,y})^2$$

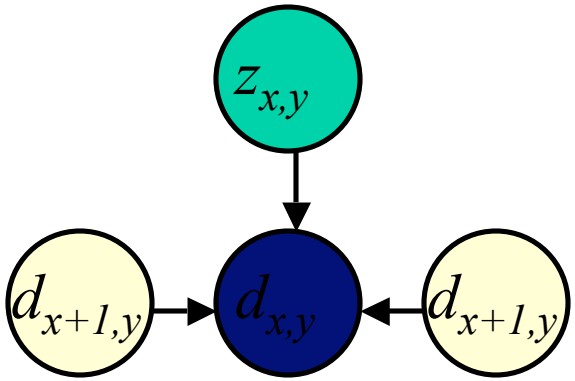
$$E_{\text{membrane}}(\mathbf{d}) = \sum_{x,y} (d_{x,y} - d_{x-1,y})^2$$

$$E_{\text{thin plate}}(\mathbf{d}) = \sum_{x,y} (2d_{x,y} - d_{x-1,y} - d_{x+1,y})^2$$



Relaxation

Iteratively improve a solution by locally minimizing the energy: *relax* to solution



Earliest application: WWII numerical simulations

Relaxation

How can we get the best solution?

Differentiate energy function, set to 0

$$\begin{aligned} \frac{\partial E}{\partial d_{x,y}} &= 2(d_{x,y} - z_{x,y}) + \\ & \quad 2\lambda(2d_{x,y} - d_{x-1,y} - d_{x+1,y}) = 0 \\ d_{x,y} &\leftarrow \frac{1}{1+2\lambda}(z_{x,y} + d_{x-1,y} + d_{x+1,y}) \end{aligned}$$

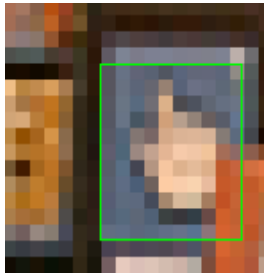
Image registration

How do we determine correspondences?

- *block matching* or *SSD* (sum squared differences)

$$E(x, y; d) = \sum_{(x',y') \in N(x,y)} [I_L(x'+d, y') - I_R(x', y')]^2$$

d is the *disparity* (horizontal motion)



Matching criteria

Raw pixel values (correlation - *photoconsistency*)

Band-pass filtered images [Jones & Malik 92]

“Corner” like features [Zhang, …]

Edges [many people…]

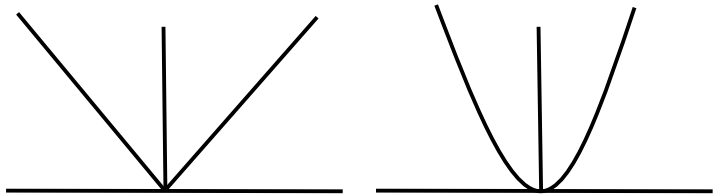
Gradients [Seitz 89; Scharstein 94]

Rank statistics [Zabih & Woodfill 94]

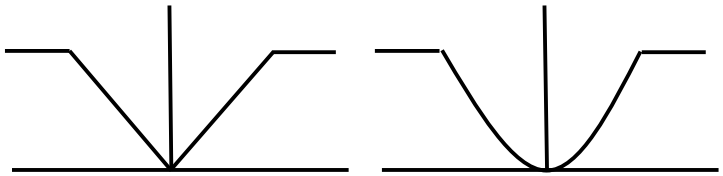
Mutual information [Kim *et al.* 03]

Matching costs

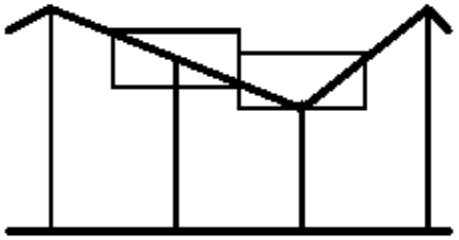
Abs. Diff., Squared Diff.



Truncated versions



Birchfield / Tomasi



Fractional disparity steps (1, _, _, …)

Aggregation

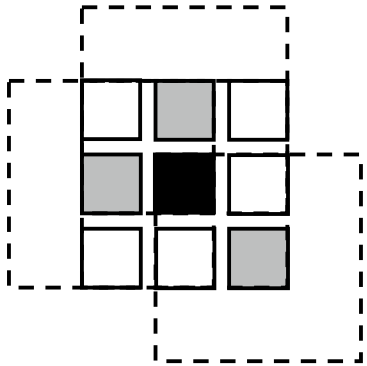
Square windows (SSD)

Binomial filters (~ Gaussian)

Shiftable windows (min filter)

Diffusion (iterative aggregation)

Bayesian diffusion

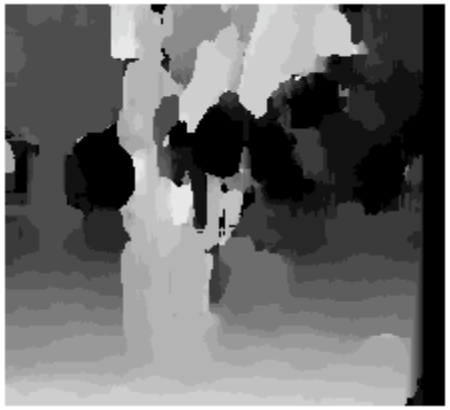


Neighborhood size

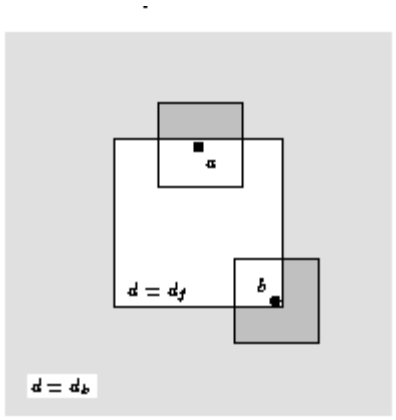
Smaller neighborhood: more details

Larger neighborhood: fewer isolated mistakes

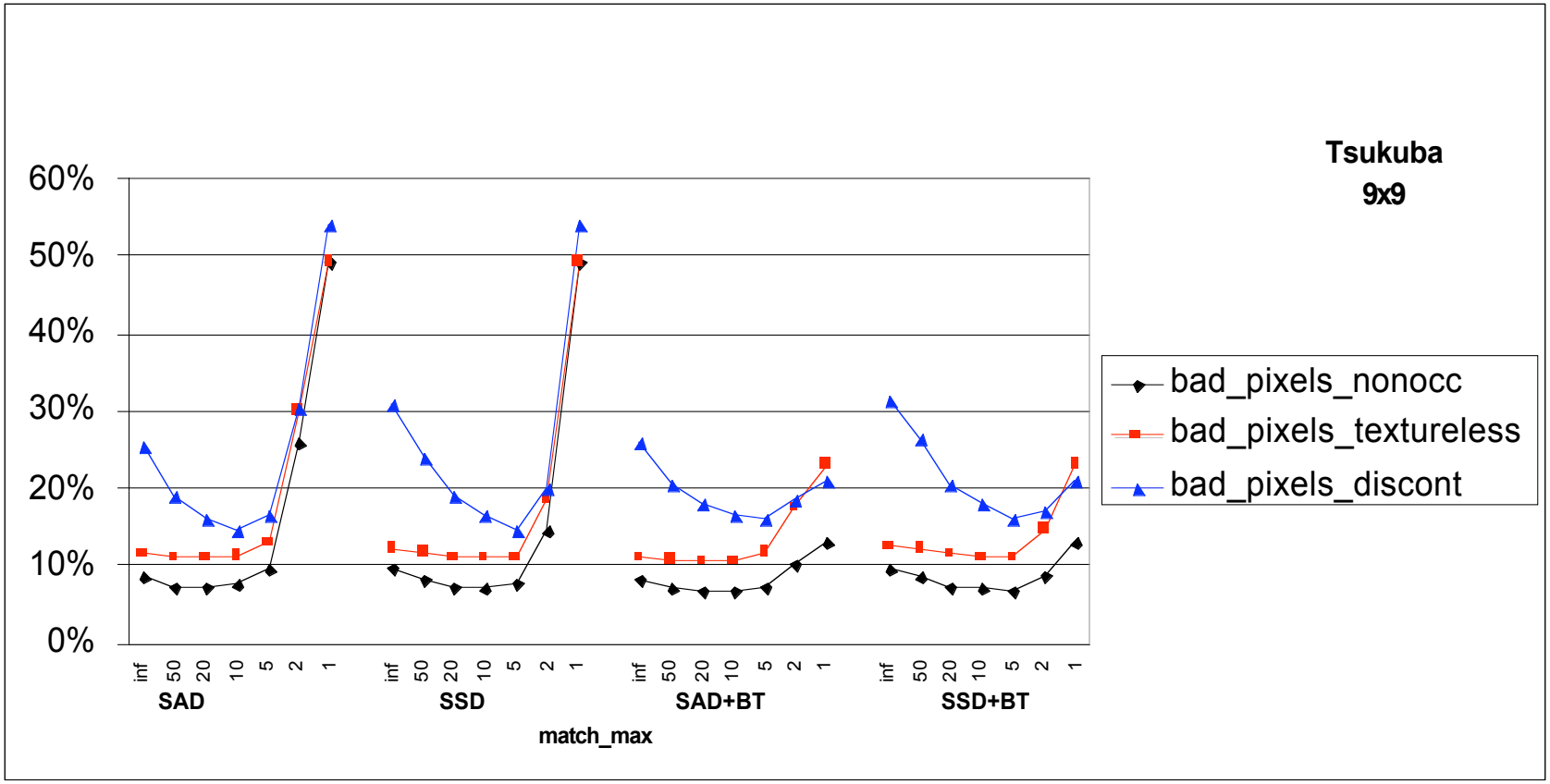
w = 3



w = 20

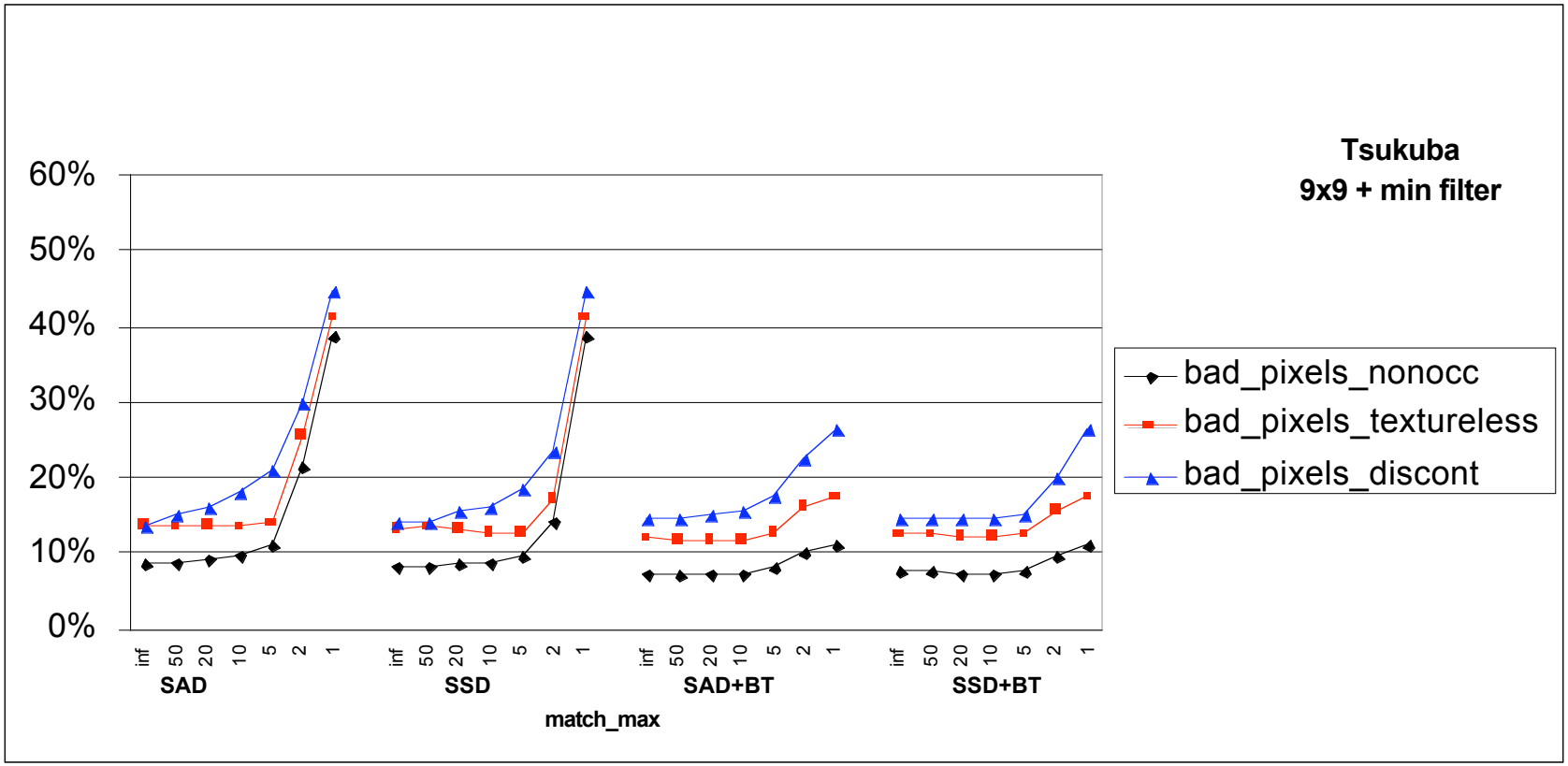


Evaluation of matching costs: 9x9 window



➔ Truncation helps

Evaluation of matching costs: 9x9 shiftable window (min filter)

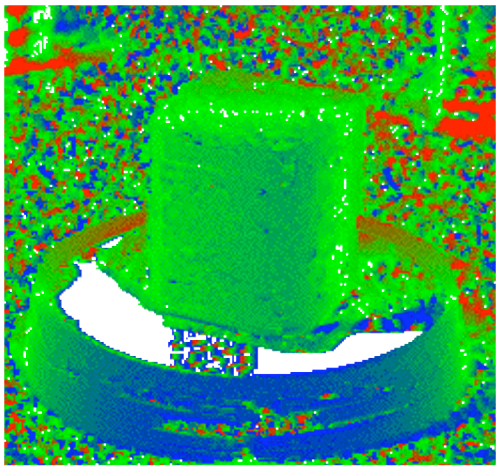


➔Truncation not necessary with min filter

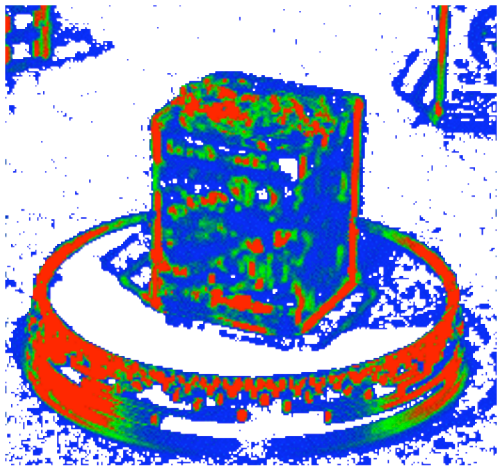
Stereo: certainty modeling

Compute certainty map from correlations

input



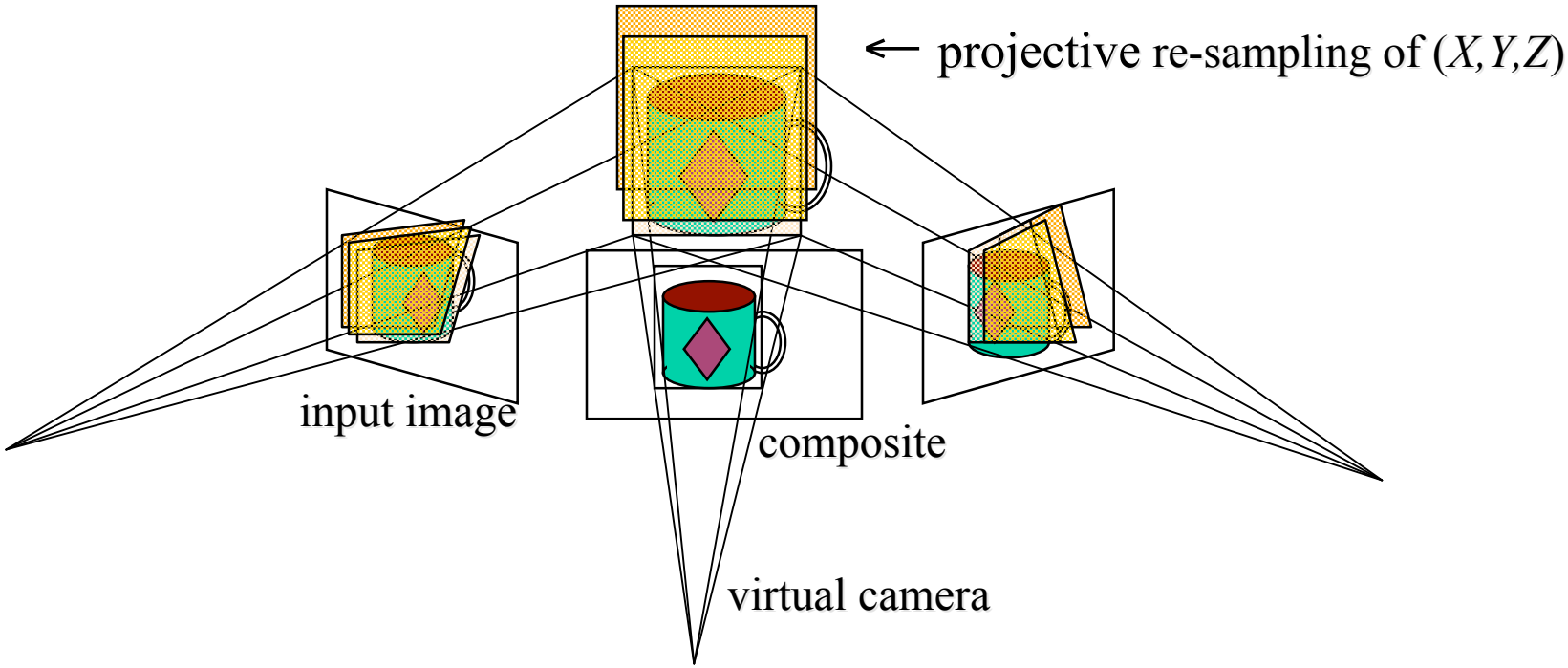
depth map



certainty map

Plane Sweep Stereo

Sweep family of planes through volume

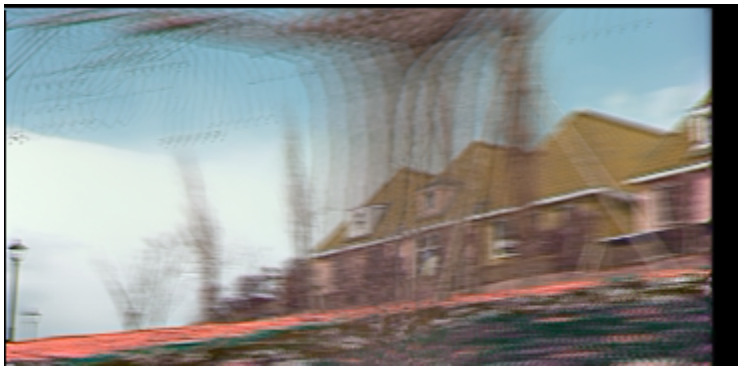


- each plane defines an image ⇒ composite homography

Plane Sweep Stereo

For each depth plane

- compute composite (mosaic) image — *mean*

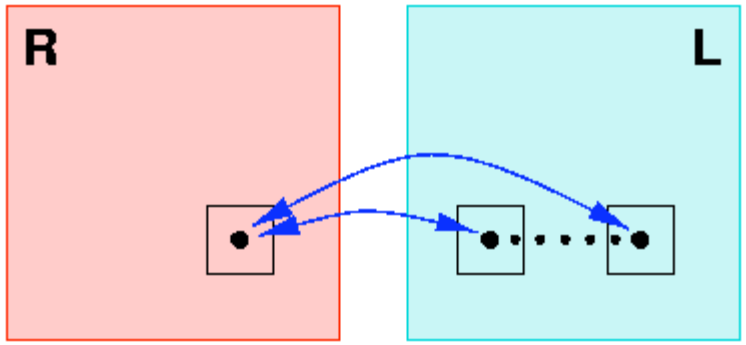


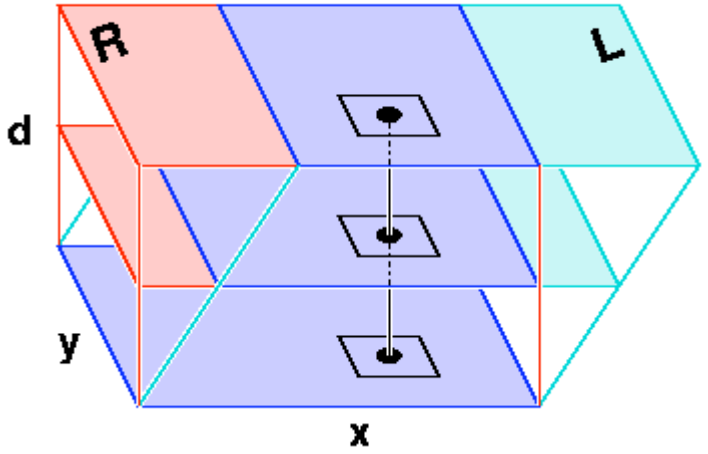
- compute error image — *variance*
- convert to confidence and aggregate spatially

Select winning depth at each pixel

Plane sweep stereo

Re-order (pixel / disparity) evaluation loops





for every pixel,
for every disparity
compute cost

for every disparity
for every pixel
compute cost

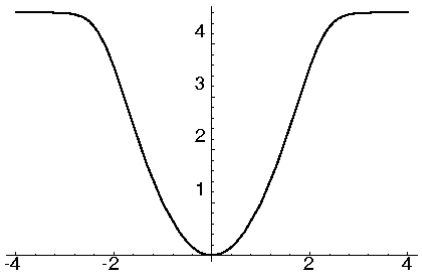
Stereo matching framework

1. For every disparity, compute *raw* matching costs

$$E_0(x, y; d) = \rho(I_L(x' + d, y') - I_R(x', y'))$$

   Why use a robust function?

   - occlusions, other outliers

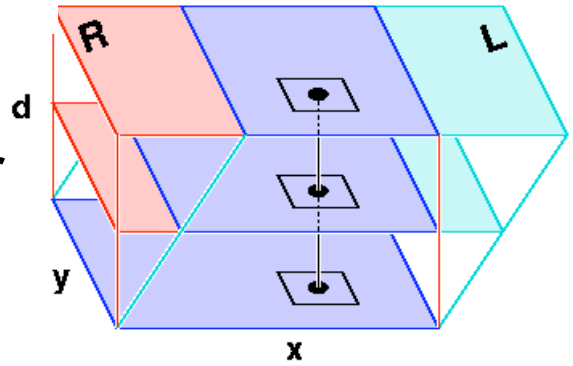


Can also use alternative match criteria

Stereo matching framework

2. Aggregate costs spatially

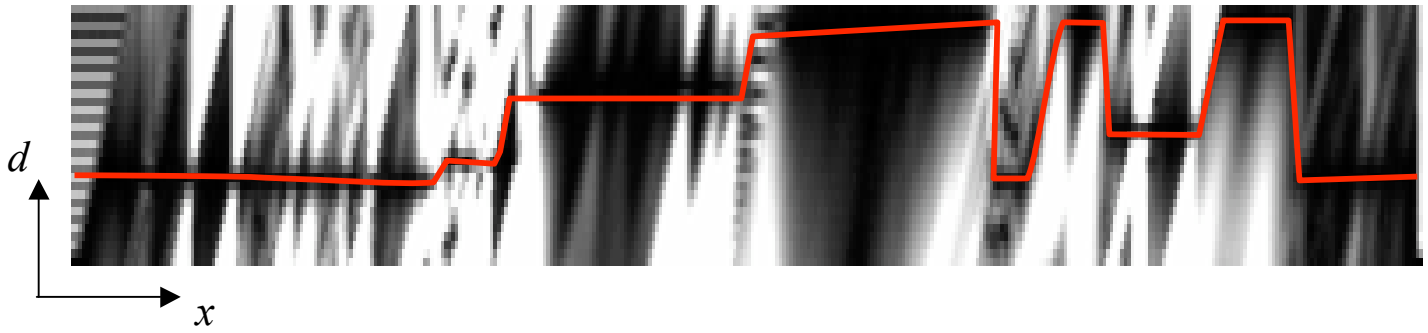
$$E(x, y; d) = \sum_{(x',y') \in N(x,y)} E_0(x', y', d)$$



- Here, we are using a *box filter* (efficient moving average implementation)
- Can also use weighted average, [non-linear] diffusion…

Disparity space image

$$\mathrm{DSI}(x, y, d) = \rho_I(I_{\mathrm{L}}(x, y) - I_{\mathrm{R}}(x - d, y))$$

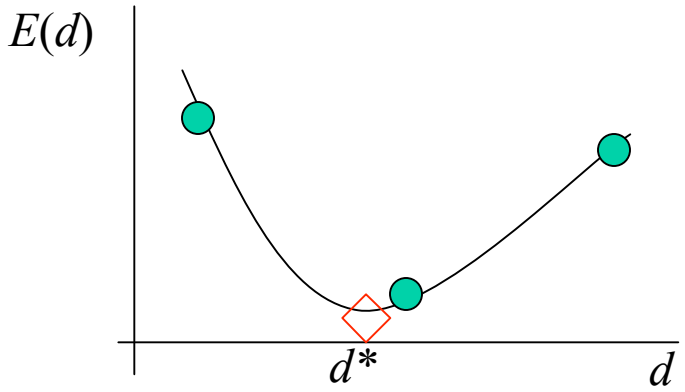


Stereo matching framework

3. Choose winning disparity at each pixel

$$d(x, y) = \arg\min_{d} E(x, y; d)$$

4. Interpolate to *sub-pixel* accuracy



Stereo with Non-Linear Diffusion

Problem with window-based approach:

- gets confused near discontinuities

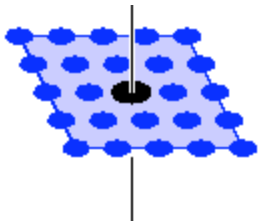
Alternative approach:

- use iterative (non-linear) aggregation to obtain better estimate
- provably equivalent to mean-field estimate of Markov Random Field

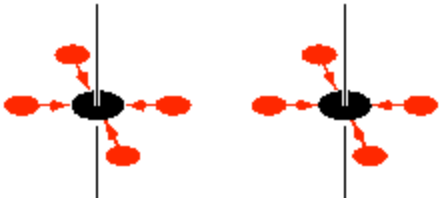
Linear diffusion

Average energy with neighbors

$$E(x,y,d) \leftarrow (1-4\lambda)E(x,y,d) + \lambda \sum_{(k,l)\in\mathcal{N}_4} E(x+k, y+l, d)$$



window

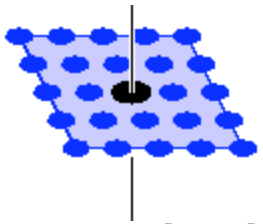


diffusion

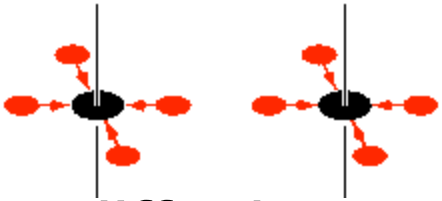
Linear diffusion

Average energy with neighbors + starting value

$$E(x,y,d) \leftarrow (1-4\lambda)E(x,y,d) + \lambda \sum_{(k,l)\in\mathcal{N}_4} E(x+k,y+l,d)$$
$$+\beta(E_0(x,y,d) - E(x,y,d))$$



window

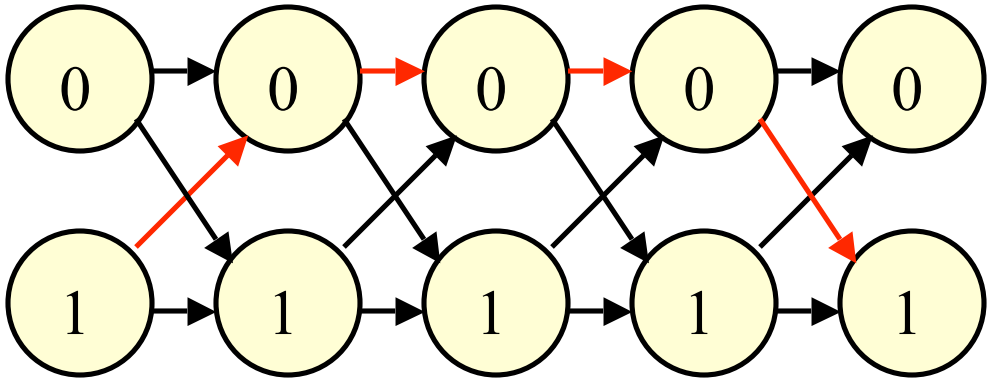


diffusion

Dynamic programming

Evaluate best cumulative cost at each pixel

$$
\begin{aligned}
E_{\text{total}}(\mathbf{d}) &= E_{\text{data}}(\mathbf{d}) + \lambda E_{\text{smoothness}}(\mathbf{d}) \\
E_{\text{data}}(\mathbf{d}) &= \sum_{x,y} (d_{x,y} - z_{x,y})^2 \\
E_{\text{smoothness}}(\mathbf{d}) &= \sum_{x,y} |d_{x,y} - d_{x-1,y}|
\end{aligned}
$$

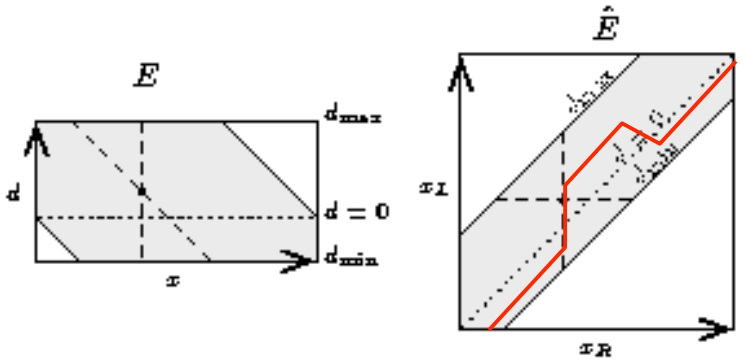


Dynamic programming

1-D cost function

$$E(\mathbf{d}) = \sum_{x,y} \rho_P(d_{x+1,y} - d_{x,y}) + \sum_{x,y} E_0(x, y; d)$$

$$\tilde{E}(x, y, d) = E_0(x, y; d) + \min_{d'} \left(\tilde{E}(x-1, y, d') + \rho_P(d_{x,y} - d'_{x-1,y}) \right)$$



Dynamic programming

Disparity space image and min. cost path

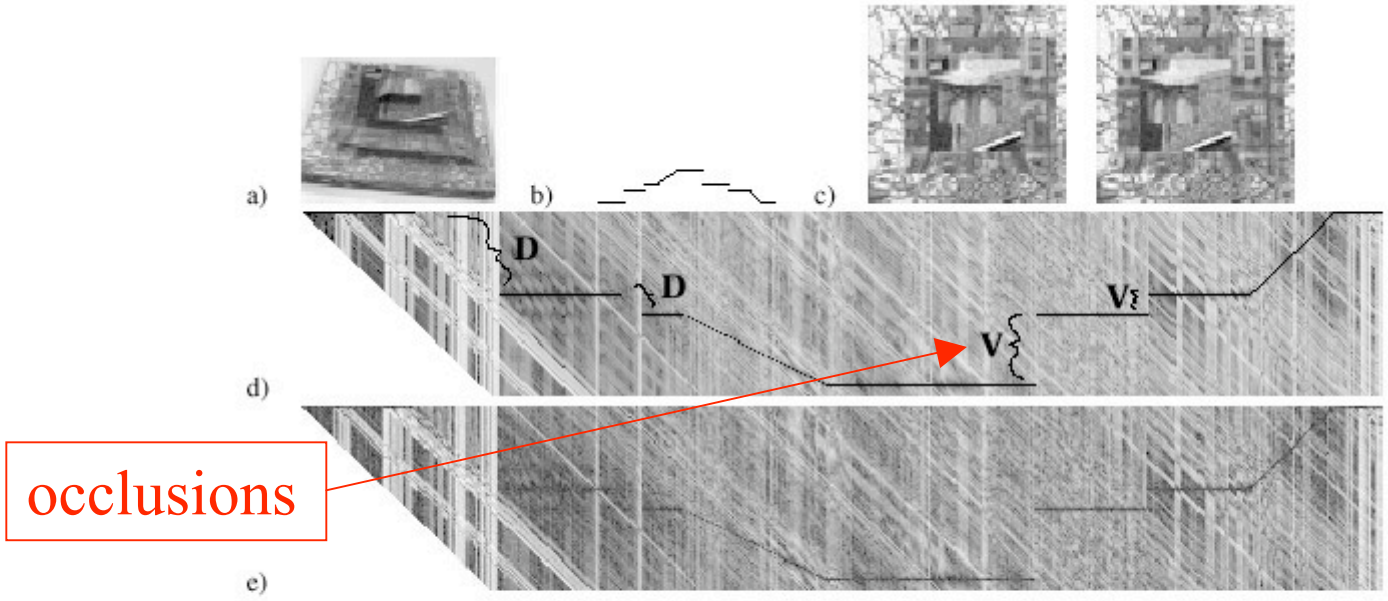


Fig. 4. This figure shows (a) a model of the stereo sloping wedding cake that we will use as a test example, (b) a depth profile through the center of the sloping wedding cake, (c) a simulated, noise-free image pair of the cake, (d) the enhanced, cropped, correlation $DSI=$ representation for the image pair in (c), and (e) the enhanced, cropped, correlation DSI for a noisy sloping wedding cake (SNR = 18 dB). In (d), the regions labeled "D" mark diagonal gaps in the matching path caused by regions occluded in the left image. The regions labeled "V" mark vertical jumps in the path caused by regions occluded in the right image.

Dynamic programming

Sample result
(note horizontal streaks)

[Intille & Bobick]

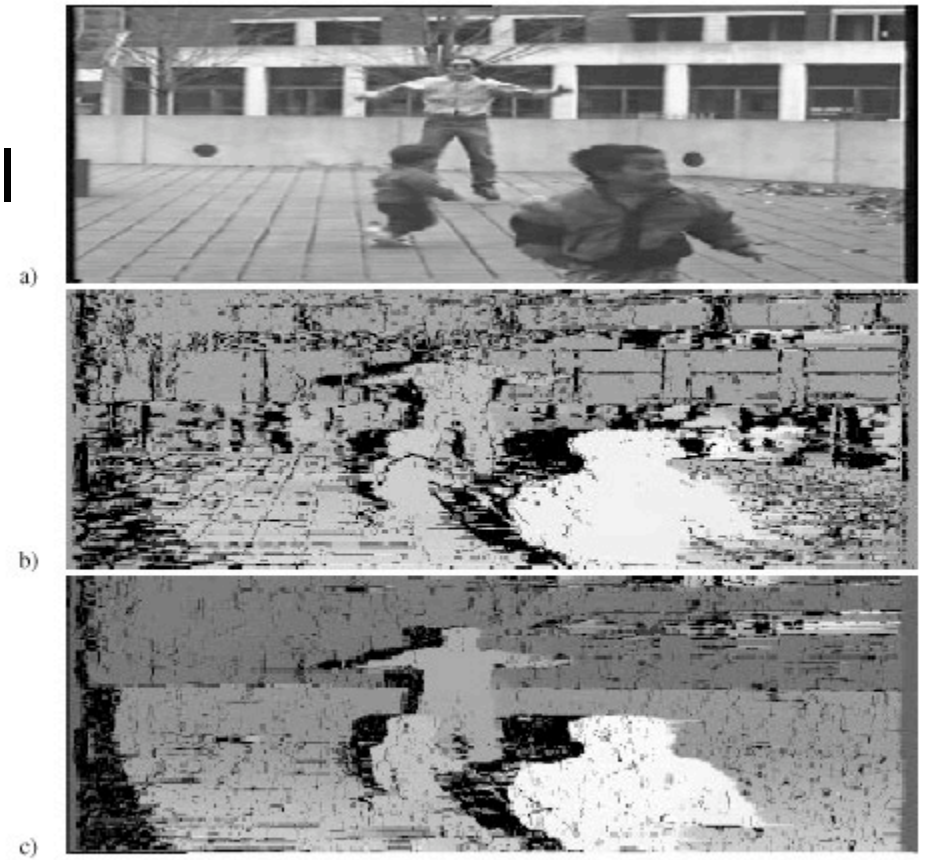
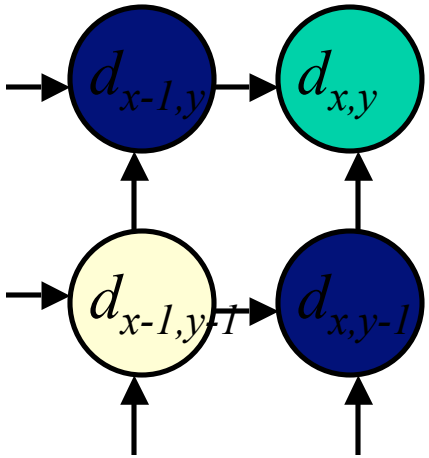


Fig. 12. Results of two stereo algorithms on Figure 1. (a) Original left image. (b) Cox *et al.* algorithm[14], and (c) the algorithm described in this paper.

Dynamic programming

Can we apply this trick in 2D as well?



No: $d_{x,y-1}$ and $d_{x-1,y}$ may depend on different values of $d_{x-1,y-1}$

General 2D optimization

General 2D energy function

$$
\begin{aligned}
E_{\text{total}}(\mathbf{d}) &= E_{\text{data}}(\mathbf{d}) + \lambda E_{\text{smoothness}}(\mathbf{d}) \\
E_{\text{data}}(\mathbf{d}) &= \sum_{x,y} f_{x,y}(d_{x,y}) \\
E_{\text{smoothness}}(\mathbf{d}) &= \sum_{x,y} \rho(d_{x,y} - d_{x-1,y}) \\
&\quad + \sum_{x,y} \rho(d_{x,y} - d_{x,y-1})
\end{aligned}
$$

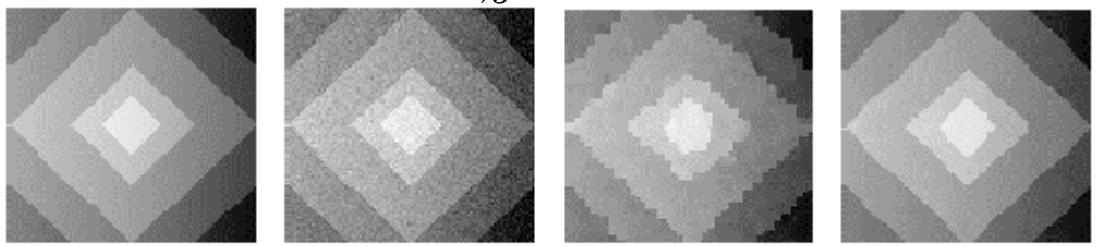
Advantages:

- well defined cost, minimum, search

Graph cuts

Solution technique for general 2D problem

$$
\begin{aligned}
E_{\text{total}}(\mathbf{d}) &= E_{\text{data}}(\mathbf{d}) + \lambda E_{\text{smoothness}}(\mathbf{d}) \\
E_{\text{data}}(\mathbf{d}) &= \sum_{x,y} f_{x,y}(d_{x,y}) \\
E_{\text{smoothness}}(\mathbf{d}) &= \sum_{x,y} \rho(d_{x,y} - d_{x-1,y}) \\
&\quad + \sum_{x,y} \rho(d_{x,y} - d_{x,y-1})
\end{aligned}
$$



(a) original image (b) observed image (c) local min w.r.t. standard moves (d) local min w.r.t. α-expansion moves

Graph cuts

[Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(11)*, 1222-1239]

α-β swap

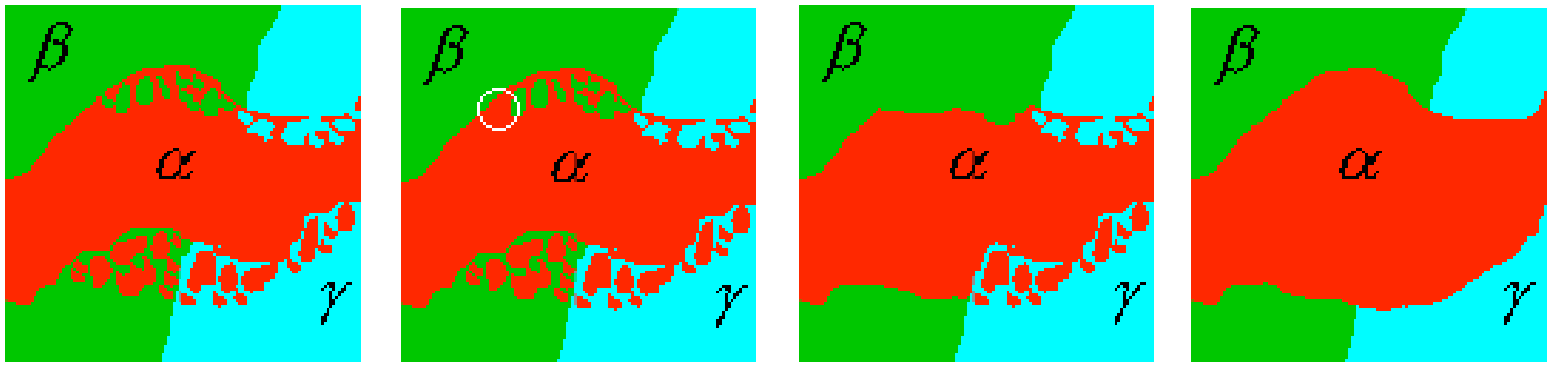
α expansion

modify smoothness penalty based on edges

compute best possible match within integer disparity

Graph cuts

Two different kinds of moves:



(a) initial labeling (b) standard move (c) α-β-swap (d) α-expansion

Bayesian inference

Formulate as statistical inference problem

Prior model $p_P(\boldsymbol{d})$

Measurement model $p_M(I_L, I_R | \boldsymbol{d})$

Posterior model

$$p_M(\boldsymbol{d} \mid I_L, I_R) \propto p_P(\boldsymbol{d})\, p_M(I_L, I_R | \boldsymbol{d})$$

Maximum a Posteriori (MAP estimate):

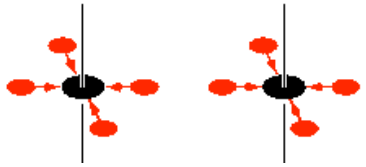
maximize $p_M(\boldsymbol{d} \mid I_L, I_R)$

Markov Random Field

Probability distribution on disparity field *d*(*x*,*y*)

$$p_P(d_{x,y}|\mathbf{d}) = p_P(d_{x,y}|\{d_{x',y'}, (x',y') \in \mathcal{N}(x,y)\})$$

$$p_P(\mathbf{d}) = \frac{1}{Z_P} e^{-E_P(\mathbf{d})}$$



$$E_P(\mathbf{d}) = \sum_{x,y} \rho_P(d_{x+1,y} - d_{x,y}) + \rho_P(d_{x,y+1} - d_{x,y})$$

Enforces *smoothness* or *coherence* on field

Measurement model

Likelihood of intensity correspondence

$$p_M(I_L, I_R|\mathbf{d}) = \frac{1}{Z_M} e^{-E_0(x,y;d)}$$

$$E_0(x, y; d) = \rho(I_L(x' + d, y') - I_R(x', y'))$$

Corresponds to Gaussian noise for quadratic ρ

MAP estimate

Maximize posterior likelihood

$$
\begin{aligned}
E(\mathbf{d}) &= -\log p(\mathbf{d}|I_L, I_R) \\
&= \sum_{x,y} \rho_P(d_{x+1,y} - d_{x,y}) + \rho_P(d_{x,y+1} - d_{x,y}) \\
&\quad + \sum_{x,y} \rho_M(I_L(x + d_{x,y}, y) - I_R(x, y))
\end{aligned}
$$

Equivalent to *regularization* (energy minimization with smoothness constraints)

Why Bayesian estimation?

Principled way of determining cost function

Explicit model of noise and prior knowledge

Admits a wider variety of optimization algorithms:

- gradient descent (local minimization)
- stochastic optimization (Gibbs Sampler)
- mean-field optimization
- graph theoretic (actually deterministic) [Zabih]
- [loopy] belief propagation

Mean-field interpretation

Bayesian non-linear diffusion rule:

- update your probability distribution assuming your neighbors' distributions are *independent* (valid for Markov chain)

Equivalent to finding best *factored* approximation

$$P(\mathbf{d}|I_L,I_R) \sim Q(\mathbf{d}) = \prod_i Q_i(d_i)$$

Mean-field interpretation

log MAP estimate

$$-\log P(\mathbf{d}|I_L,I_R) = \sum_{ij} E_{ij}(d_i,d_j) + \sum_i E_i(d_i)$$
$$= \sum_{ij} \mathbf{s}_i A_{ij} \mathbf{s}_i + \sum_i b_i \mathbf{s}_i$$

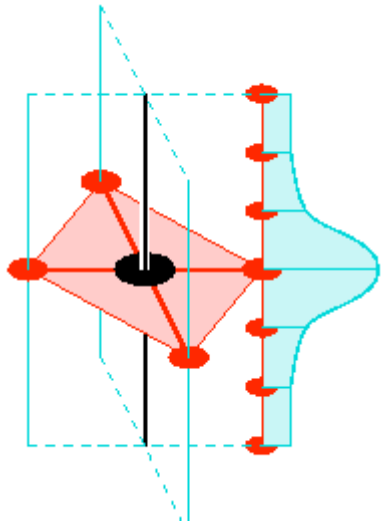
Kullback-Leibler divergence

$$D_{KL} = H(Q) - \sum_{\mathbf{d}} Q(\mathbf{d}) \log P(\mathbf{d})$$
$$= \sum_{ik} q_{ik} \log q_{ik} + \sum_{ij} \mathbf{s}_i A_{ij} \mathbf{s}_i + \sum_i b_i \mathbf{s}_i$$

Mean-field interpretation

minimize K-L divergence with

$$\sum_k q_{ik} = 1$$



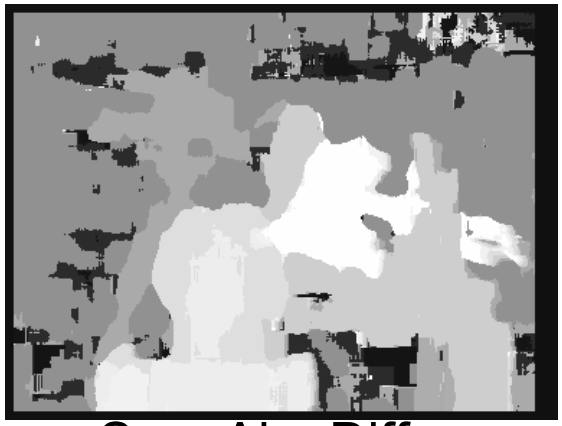
update rule:

$$q_{ik} \propto \exp[- (\sum_j \mathbf{a}_{ij}^k \mathbf{q}_j + b_{ik})]$$

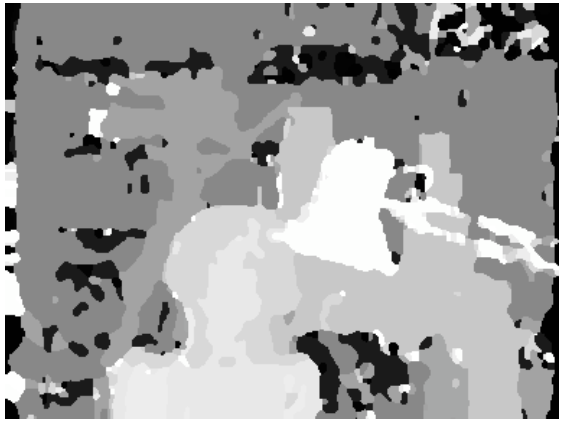
$$= \exp[- (\sum_{jl} E_{ij}(d_i=k,d_j=l)p(d_j=l) + E_i(d_i=k))]$$

Depth Map Results

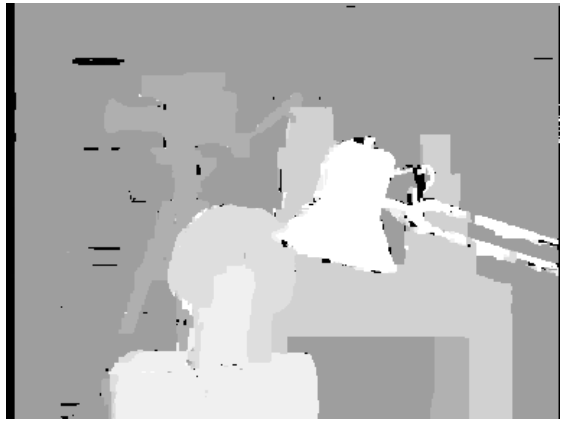
Input image



Sum Abs Diff



Mean field



Graph cuts

A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms

Daniel Scharstein, Middlebury College
Richard Szeliski, Microsoft Research
(Ramin Zabih, Cornell)

Evaluation of Stereo Algorithms

Little work on quantitative comparison

Last stereo surveys written in early 90s

Need way to assess individual algorithm components and design decisions

No standard data set with ground truth

[Scharstein & Szeliski,
Intl. J. Comp. Vision, 47(1):7-42]
http://www.middlebury.edu/stereo

Overall comparison

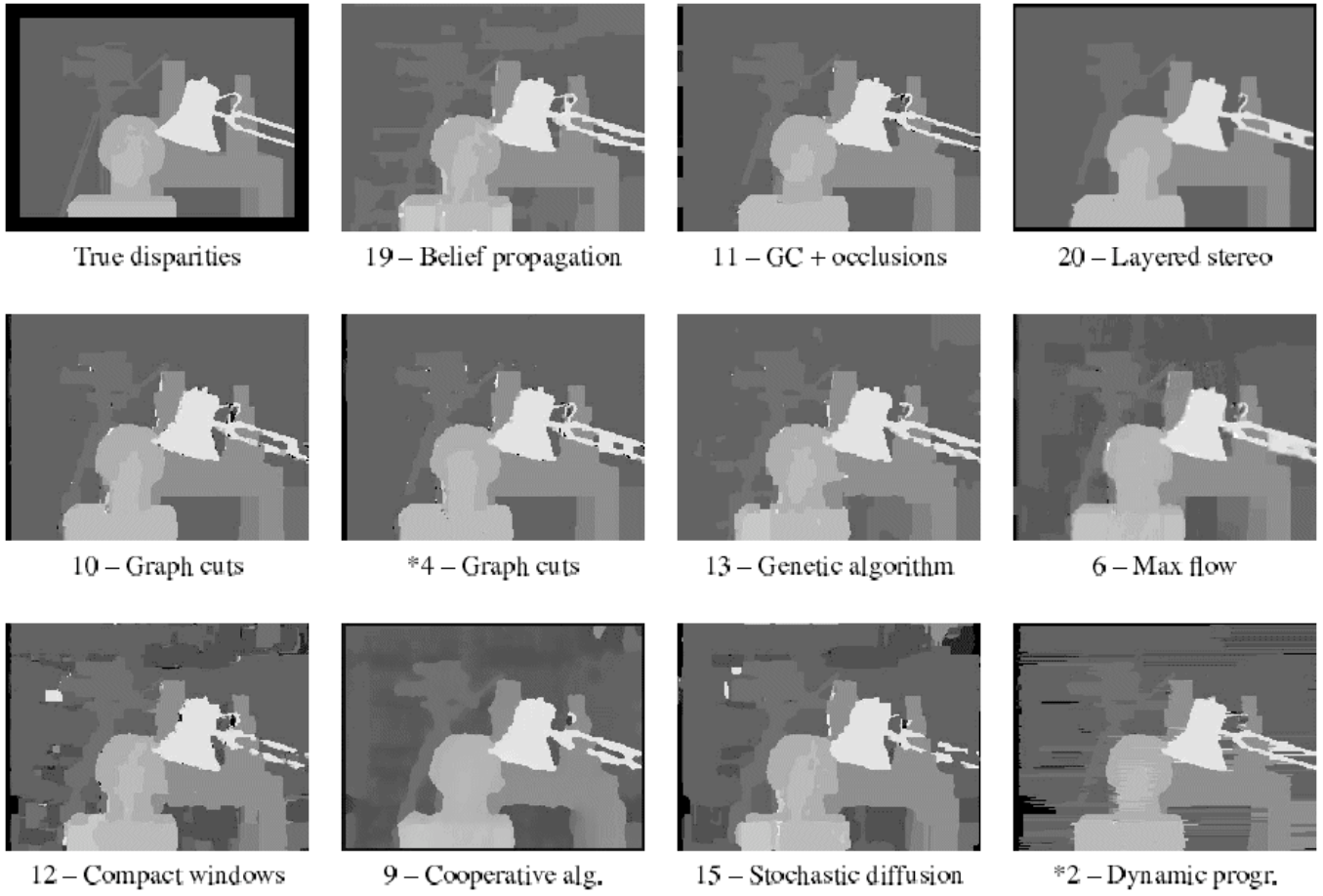
Compared

- 5 methods implemented by us
  (SSD+MF, DP, SO, GC, Bay)
- 15 methods provided by their authors
  (6 in this workshop)

See http://www.middlebury.edu/stereo

Sample data set [Tsukuba]

Sample depth map results



Results

- Global optimization outperforms local methods (GC is winner)
- Parameter λ highly image-specific
- Intensity-dependent smoothness cost helps
- Fractional disparity step necessary for good sub-pixel accuracy

Stereo: opportunities

Applications of machine learning / inference:

- Better prior models for reconstruction
- Better noise models for occlusions, specularities, …
- Better inference (difficult optimization problems)
  - Swendsen-Wang cuts?
- Learning of parameters, data-dependent self-tuning

Model-Based Reconstruction

Image-Based Modeling

Modeling and Rendering Architecture from Photographs

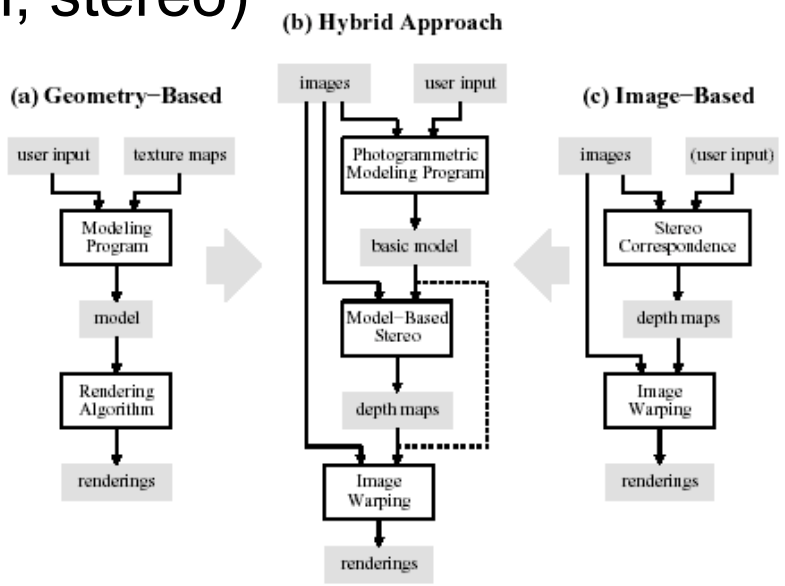
Debevec, Taylor, and Malik 1996

Original photograph with marked edges | Recovered model | Model edges projected onto photograph | Synthetic rendering

Image-Based Modeling

Create 3D model (and texture maps) from images

- automated
  - (structure from motion, stereo)
- interactive
  - Façade system



Façade

1. Select building blocks
2. Align them in each image
3. Solve for camera pose and block parameters (using constraints)

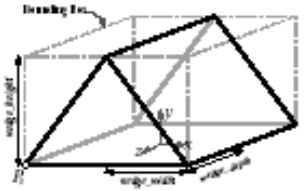


Figure 3: *A wedge block with its parameters and bounding box.*

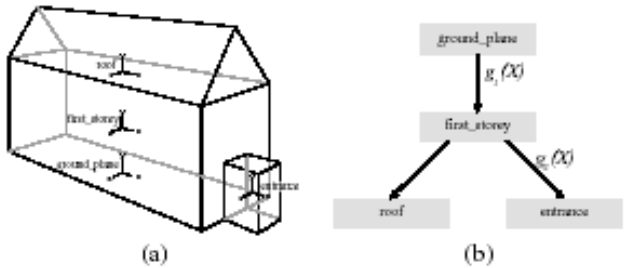
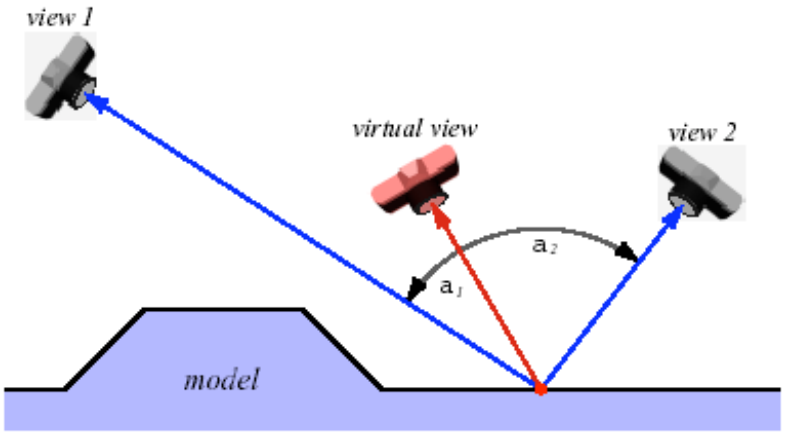
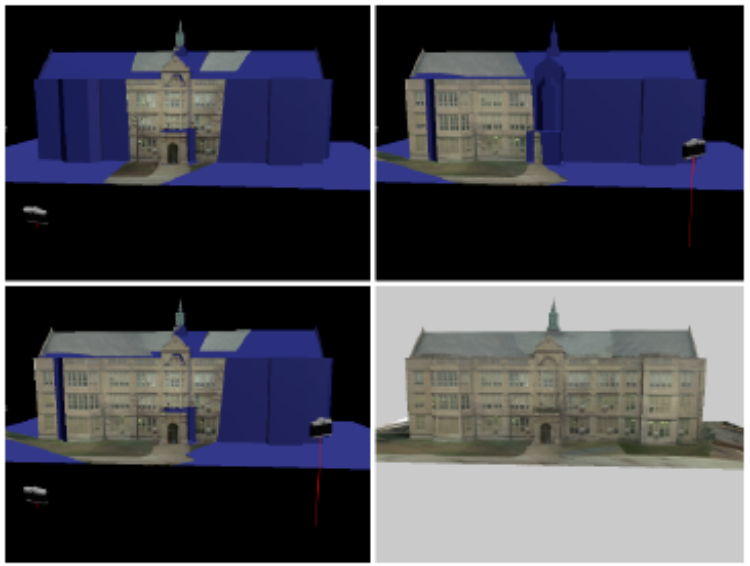


Figure 4: **(a)** *A geometric model of a simple building.* **(b)** *The model's hierarchical representation. The nodes in the tree represent parametric primitives (called* blocks*) while the links contain the spatial relationships between the blocks.*

View-dependent texture mapping

1. Determine visible cameras for each surface element
2. Blend textures (images) depending on distance between original camera and novel viewpoint





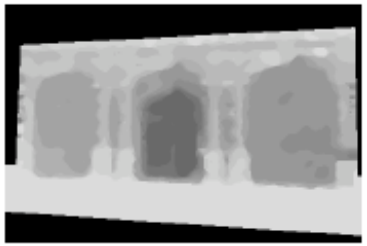
Model-based stereo

Compute offset from block model

(a) *Key Image*

(b) *Warped Offset Image*

(c) *Offset Image*



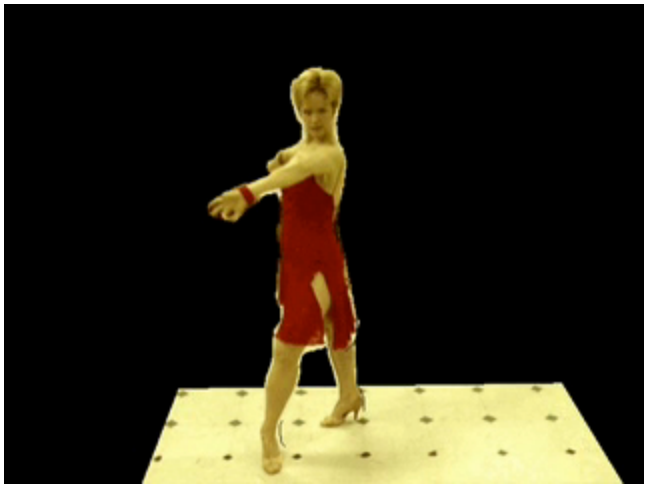
(d) *Computed Disparity Map*

Some more results:

Virtualized Reality™

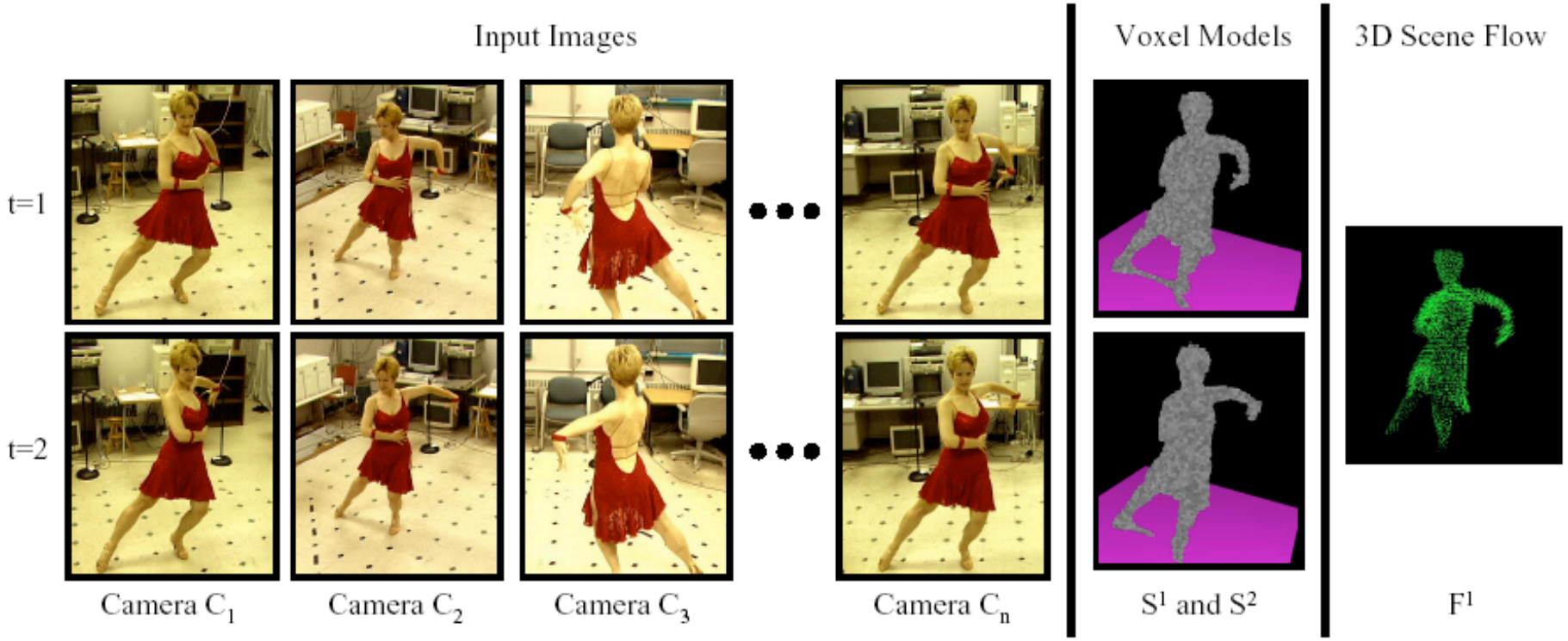
- Capture synchronized video from a full hemisphere of views.

- Perform new view generation



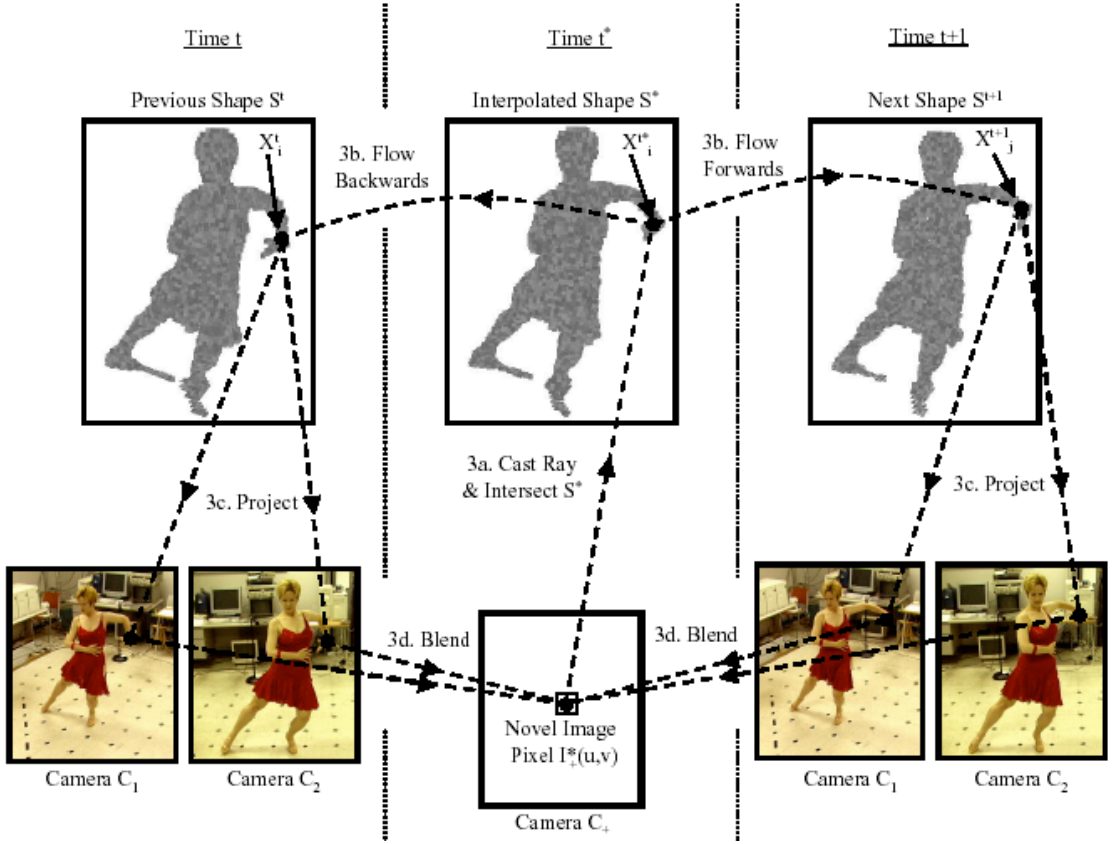
Virtualized Reality™

Spatio-Temporal View Interpolation
S. Vedula, S. Baker, and T. Kanade
Eurographics Workshop on Rendering, June, 2002.



Virtualized Reality™

Build 3D model and compute 3D scene flow, interpolate view and time.



Modeling faces from images

[Frédéric Pighin *et al.,* SIGGRAPH'98]

Goals:

- Generate photorealistic facial animation.
- Track face position & expression in video.
- Generate novel animation from tracked data

Editing faces in video:

- Lighting, camera angle, facial alterations

Performance-driven animation:

- Virtual actors & user-interface agents

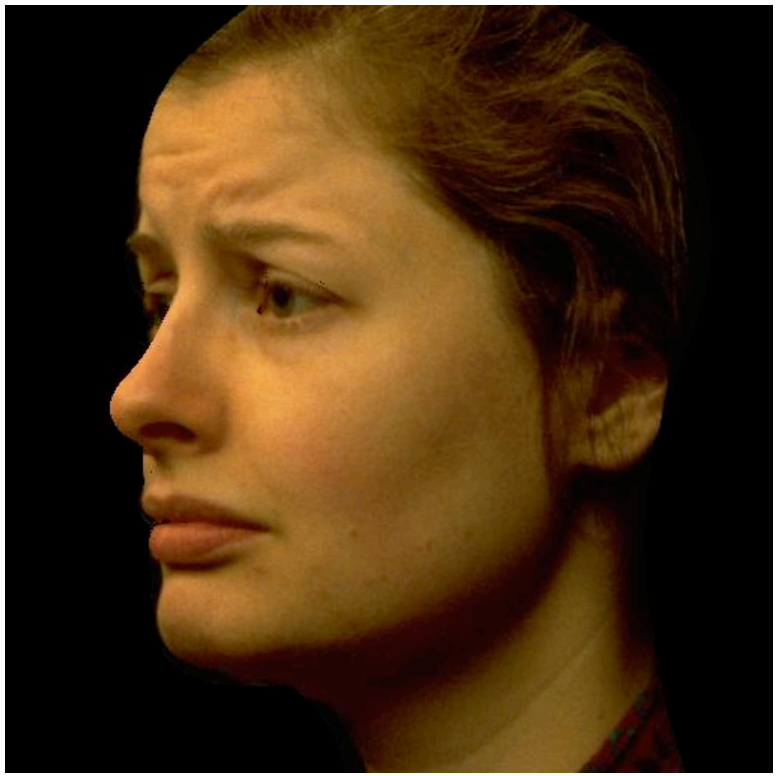
Approach

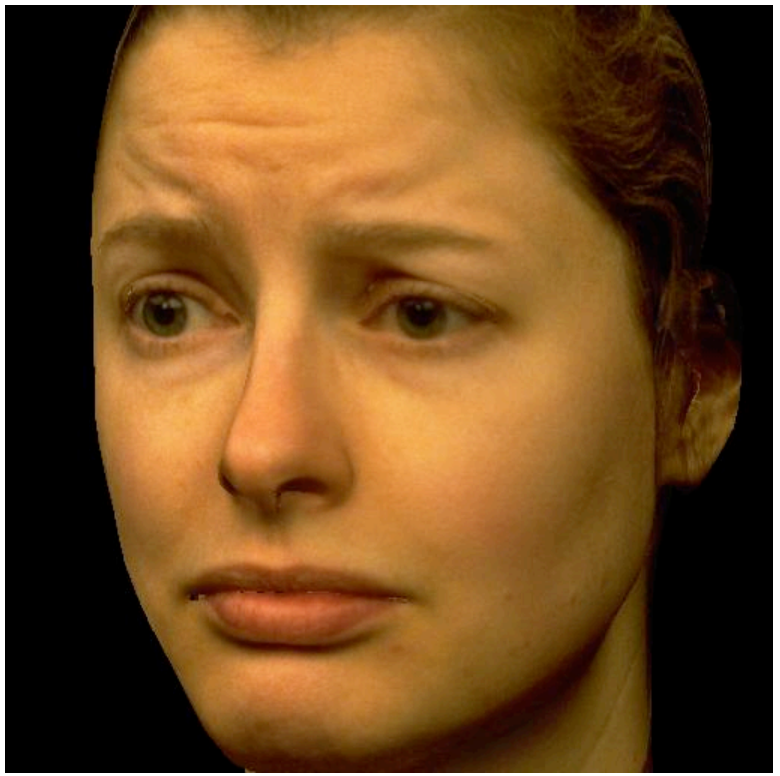
Use images to adapt a generic face model.

Face modeling example

Input images

Face modeling example, cont.

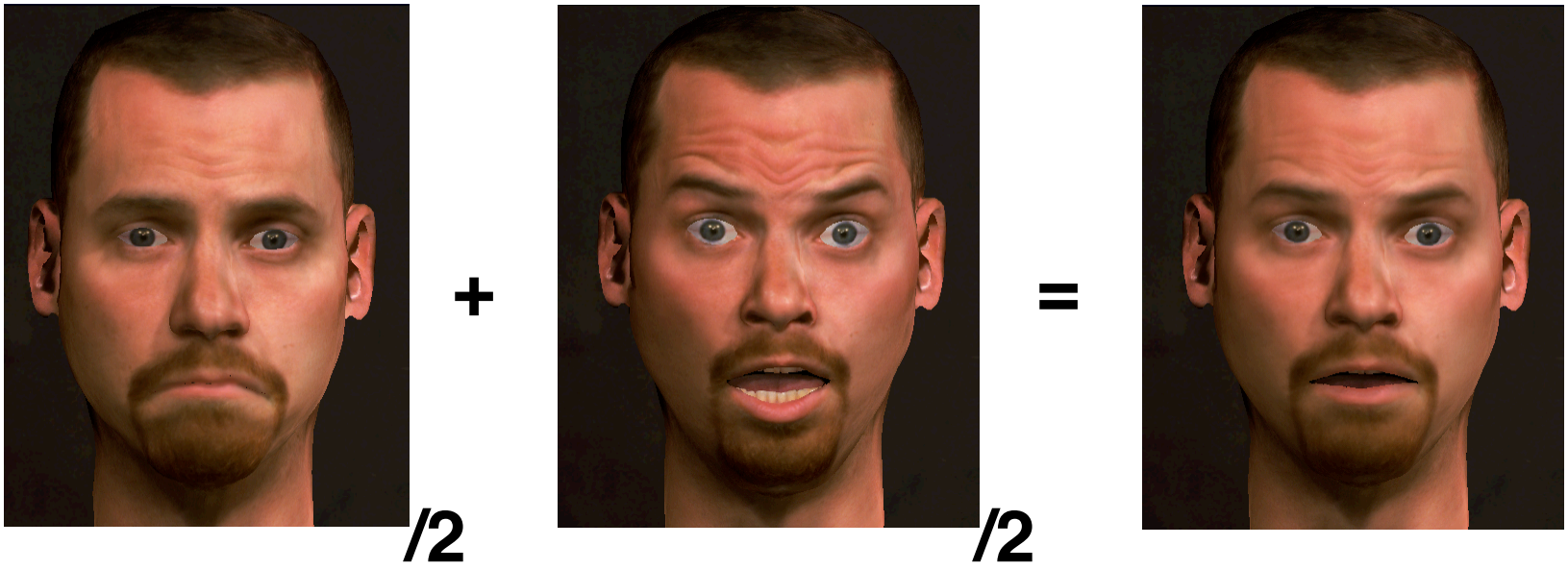




Modeling results

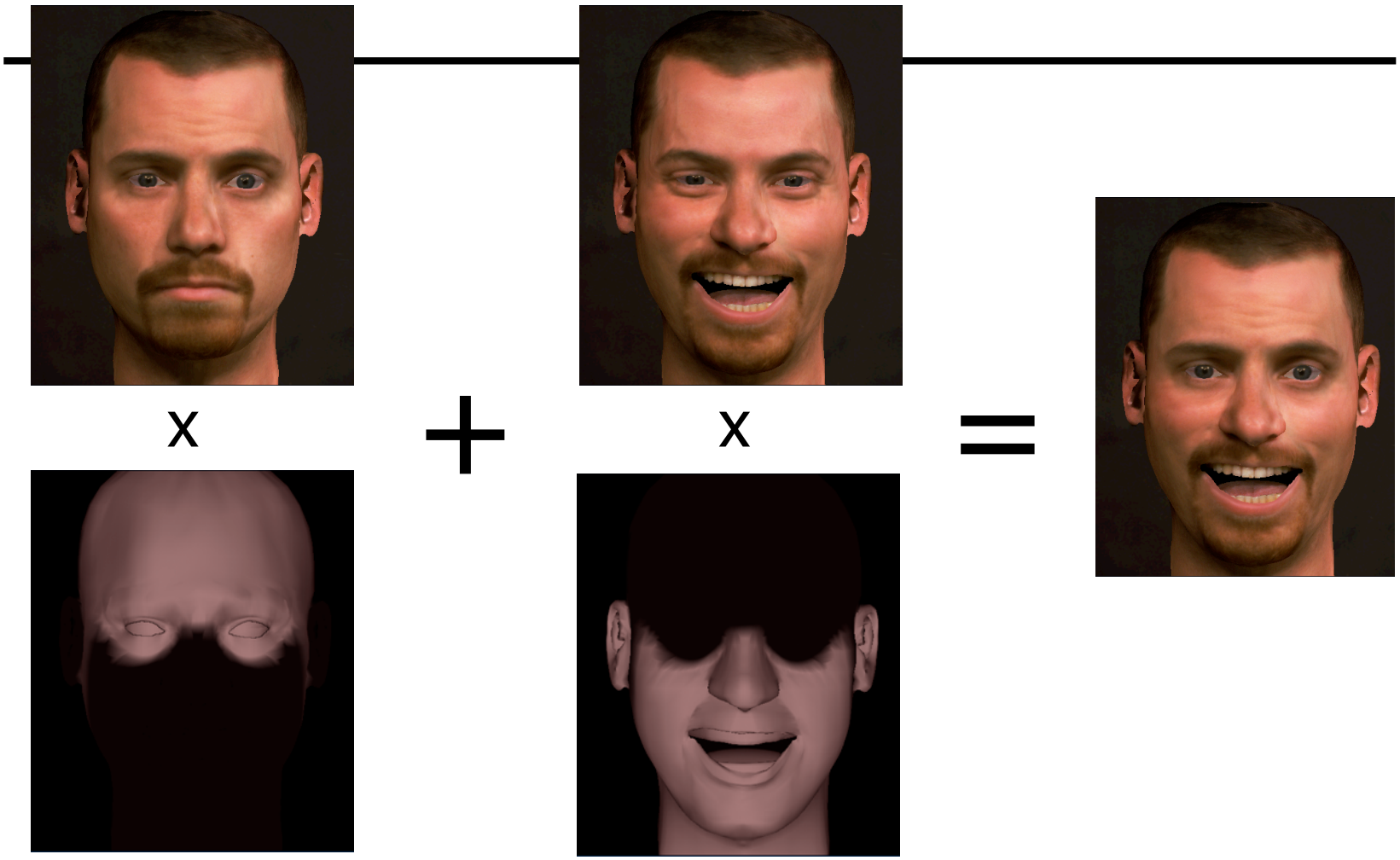
Creating new expressions

New expressions are created with 3D morphing:



Applying a global blend

Creating new expressions, cont.



Applying a region-based blend

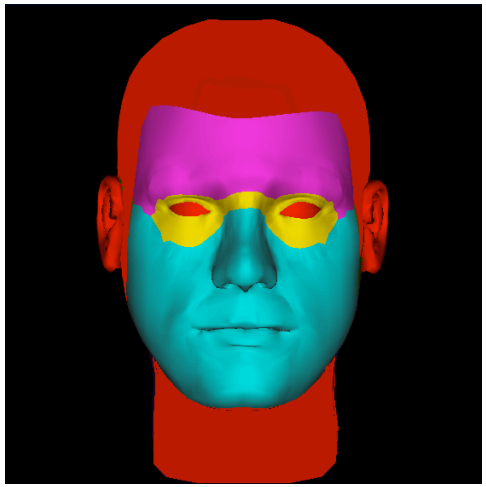
Animating between expressions

Morphing over time creates animation:

“neutral” → **“joy”**

Tracking the face

Use the 3D texture-mapped models as a basis for fitting the face.

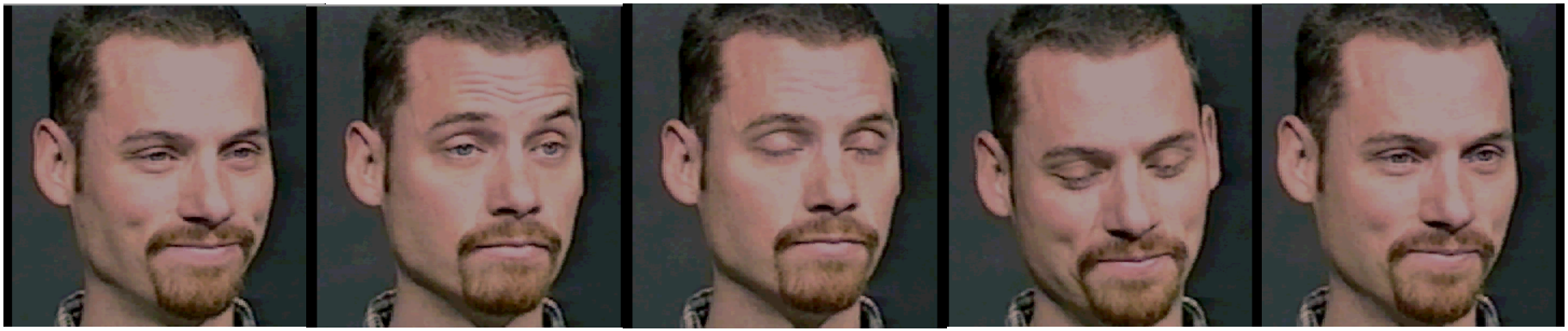


The face is split into 3 regions:

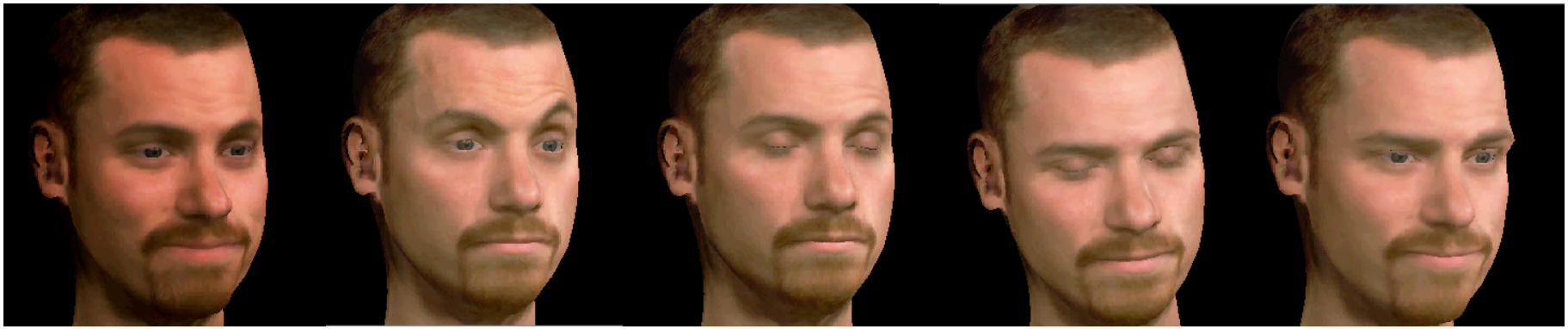
For each frame, we track:

- Position & orientation
- Expression for each region

Tracking results

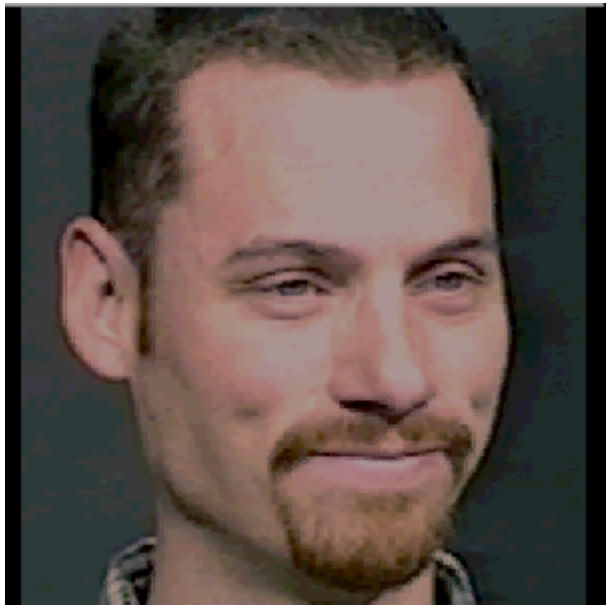


Original video frames



Tracked faces

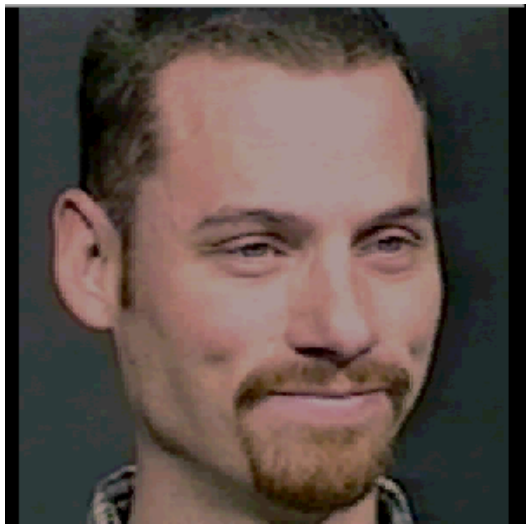
Editing: Change of viewpoint



Original

New viewpoint

Editing: Exaggerating expression

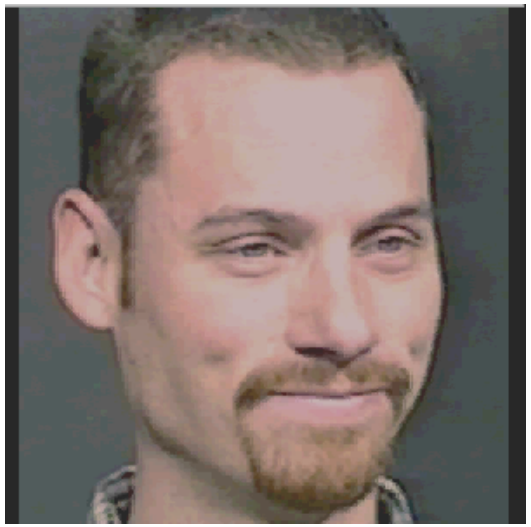


Original

Fitted model

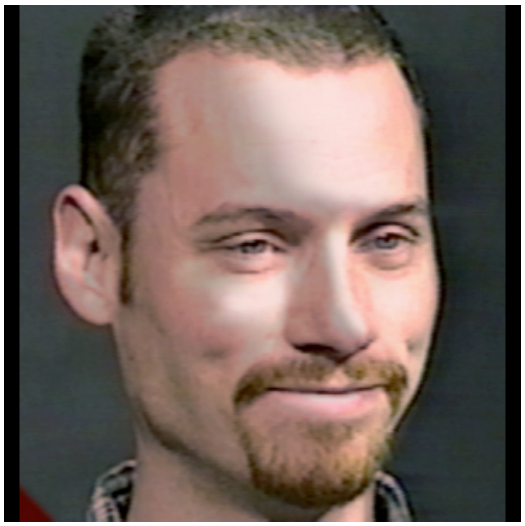
Exaggeration

Editing: Changing the lighting



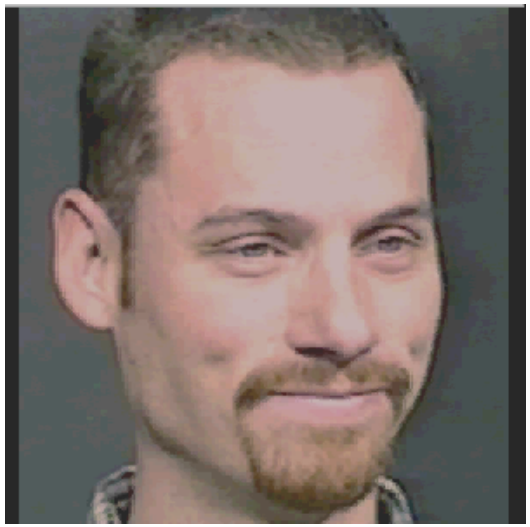
Original

Lit model

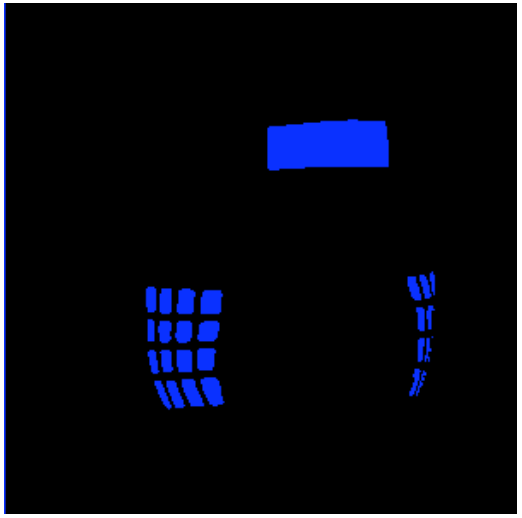


Relit

Editing: Adding tattoos



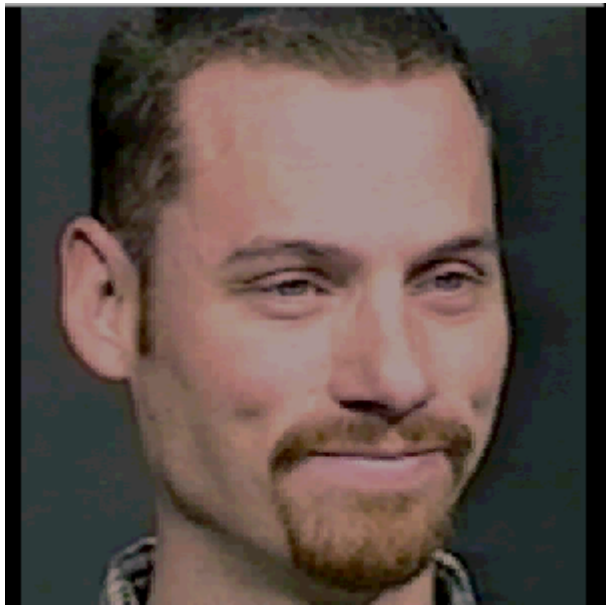
Original



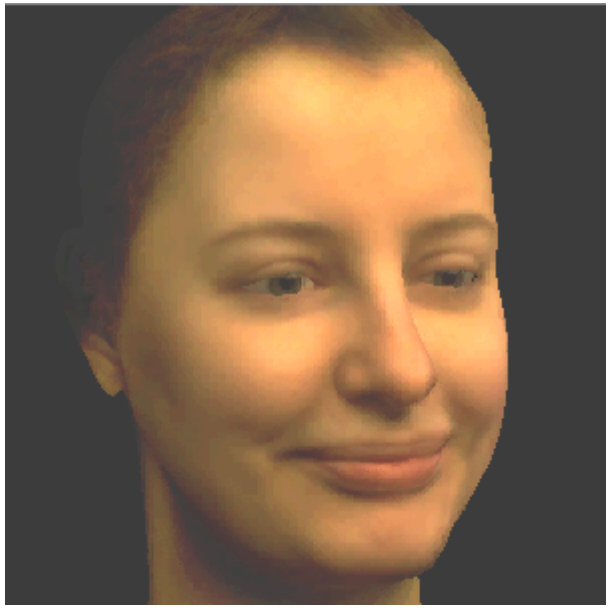
Rendered tattoo

Synthetic tattoo

Performance-driven animation



Original



Same expression on new face

Conclusion

Photorealistic facial animation via:

- Image-based modeling & rendering
- 3D morphing

Pose & expression tracking via:

- Analysis by synthesis

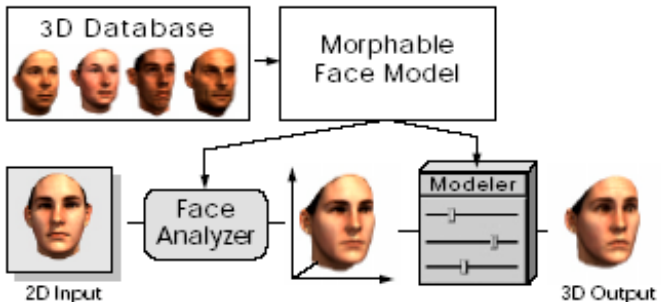
Applications include:

- Editing (change of viewpoint, lighting, etc.)
- Performance-driven animation

Morphable model for face synth.

[Blanz & Vetter, SIGGRAPH'98]

- Build a *statistical* model of 3D face shape and appearance
  - Database of color range scans, PCA
- Fit to a *single* image of a new person



Morphable model for face synth.

[Blanz & Vetter, SIGGRAPH'98]

- Statistical variation
- Single image matching

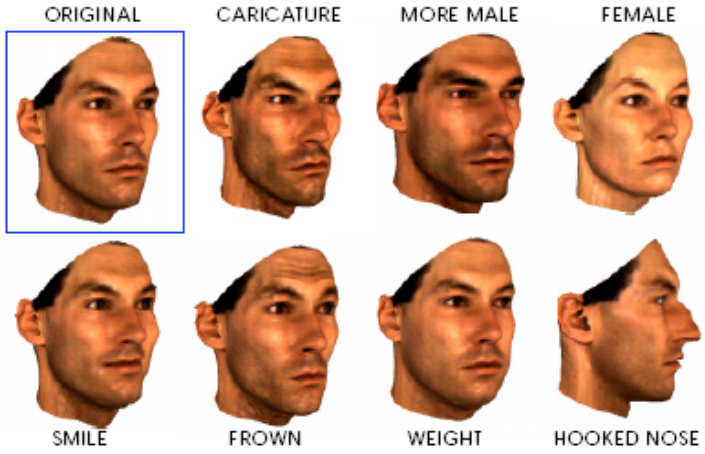


Figure 3: Variation of facial attributes of a single face. The appearance of an original face can be changed by adding or subtracting shape and texture vectors specific to the attribute.

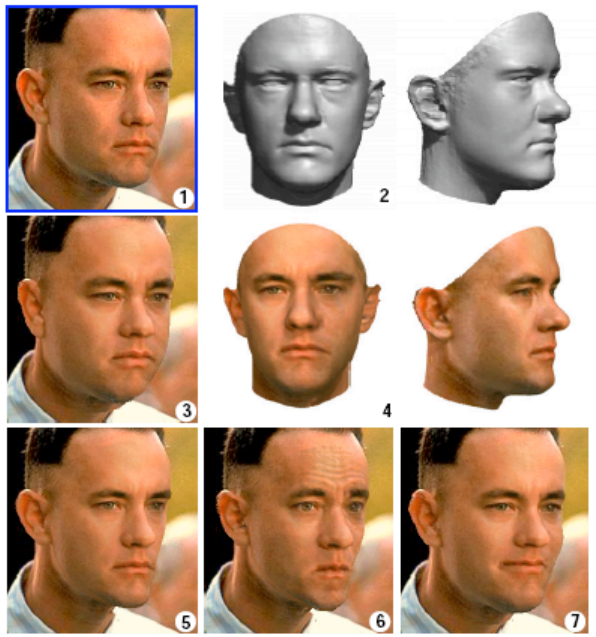


Figure 6: Matching a morphable model to a single image (1) of a face results in a 3D shape (2) and a texture map estimate. The texture estimate can be improved by additional texture extraction (4). The 3D model is rendered back into the image after changing facial attributes, such as gaining (3) and loosing weight (5), frowning (6), or being forced to smile (7).

Image-Based Faces

Estimate shape from images

Match metrics to shape

Project video onto shape

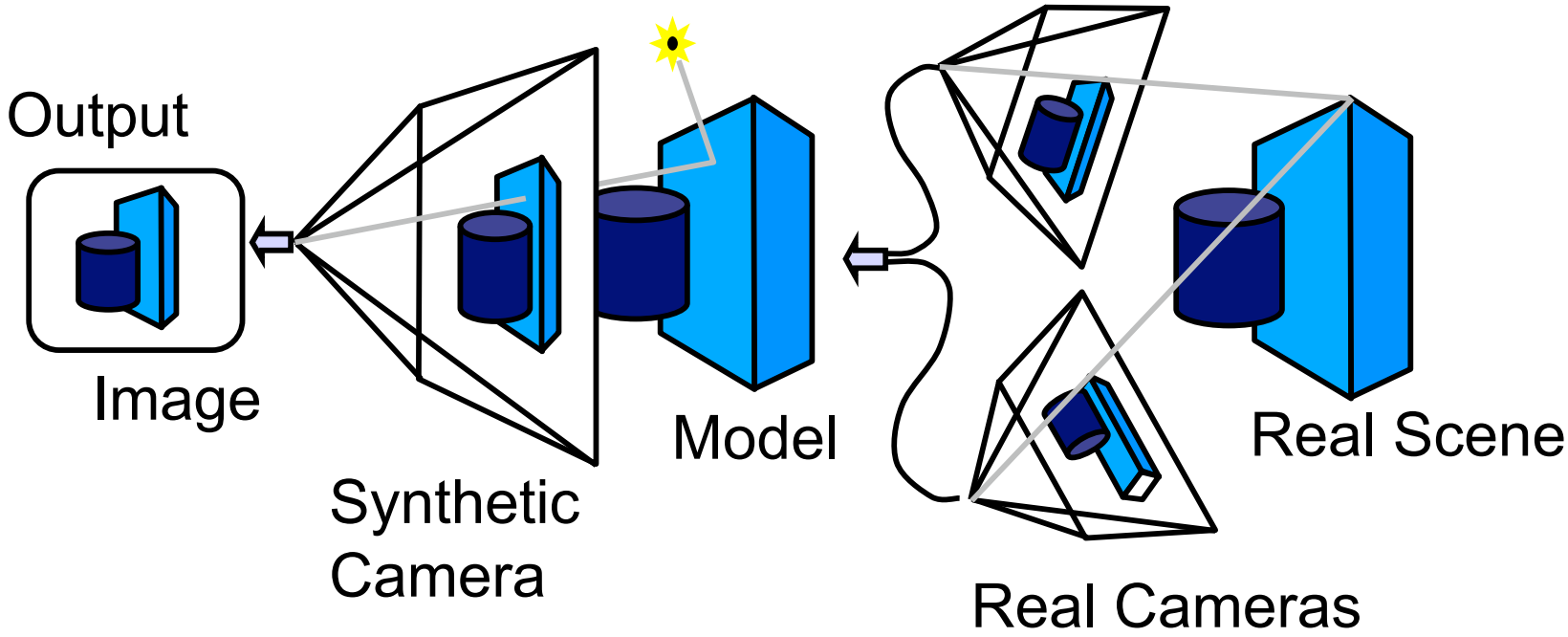
- Texture map

Animate

[Z. Liu *et al.*, MSR-TR-2000-11]

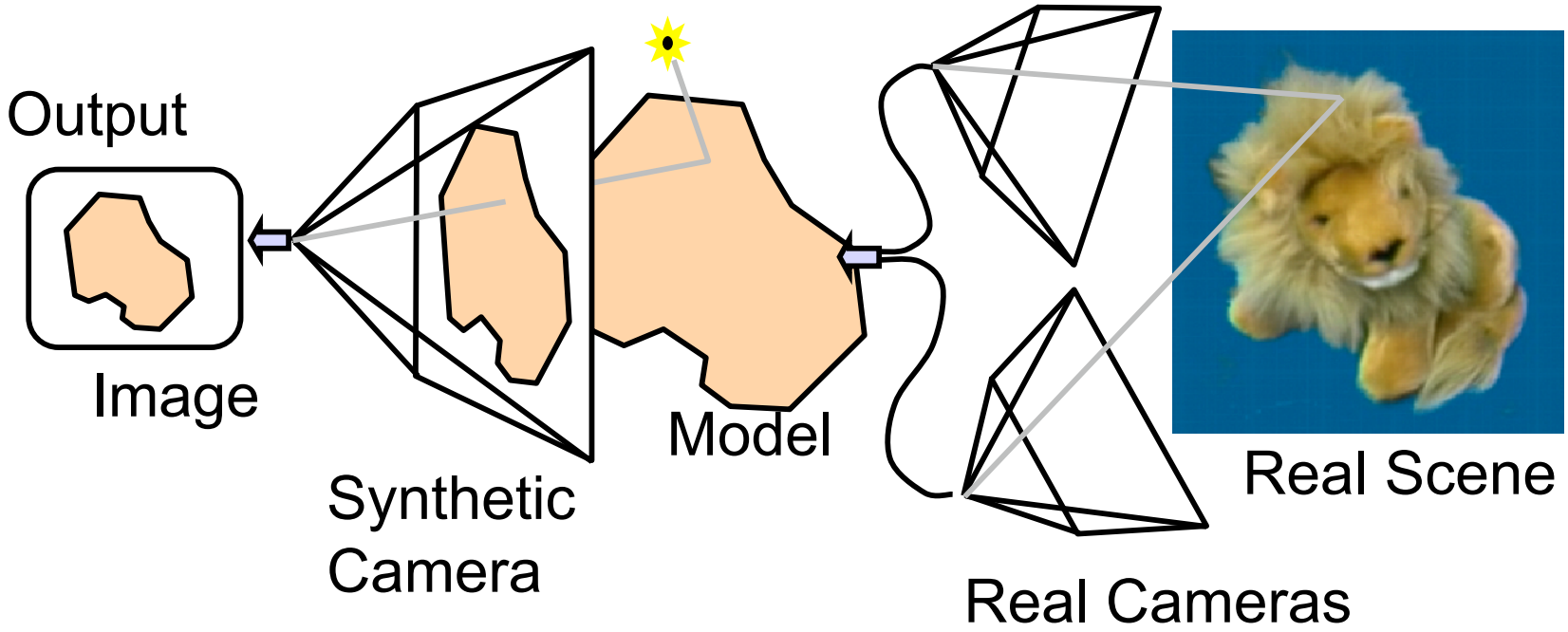
Image and Video-Based Rendering

Vision & graphics (revisited)



(slides courtesy of Michael Cohen) ***ideal model***

But, vision technology falls short



… and so does graphics.

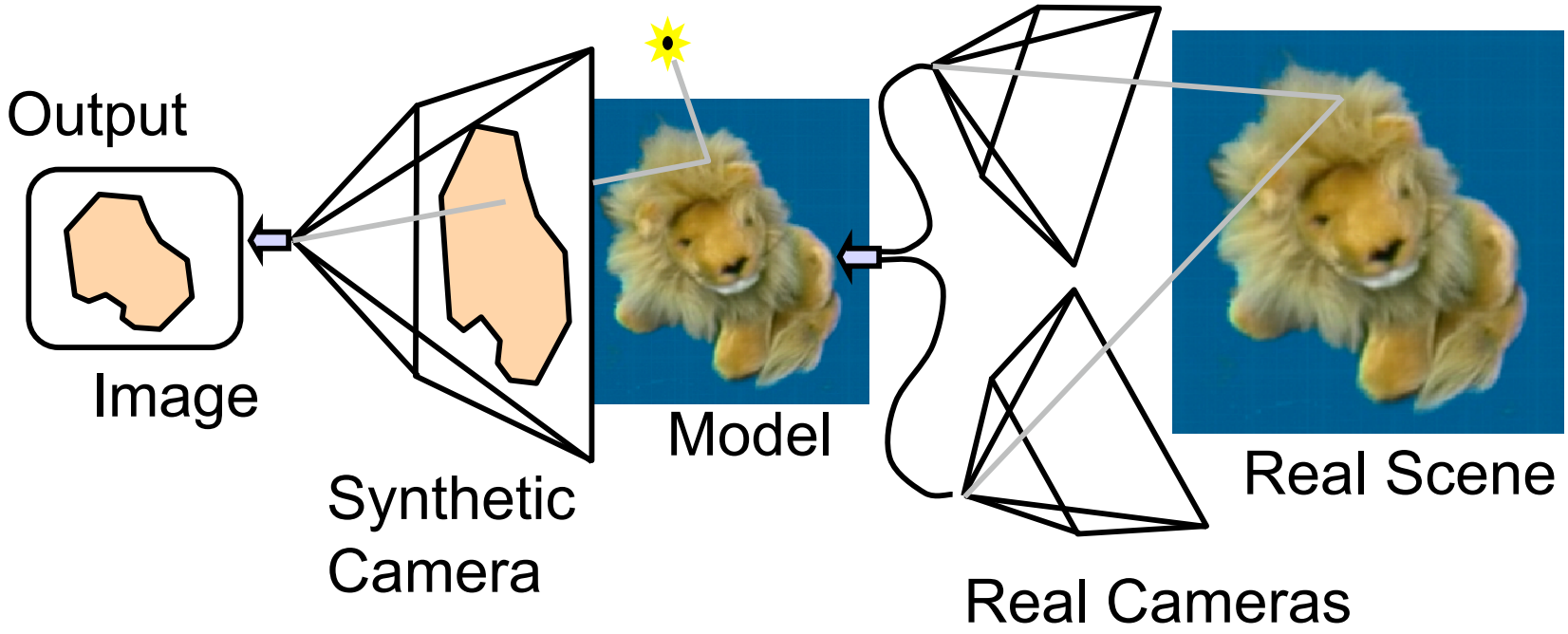


Image Based Rendering

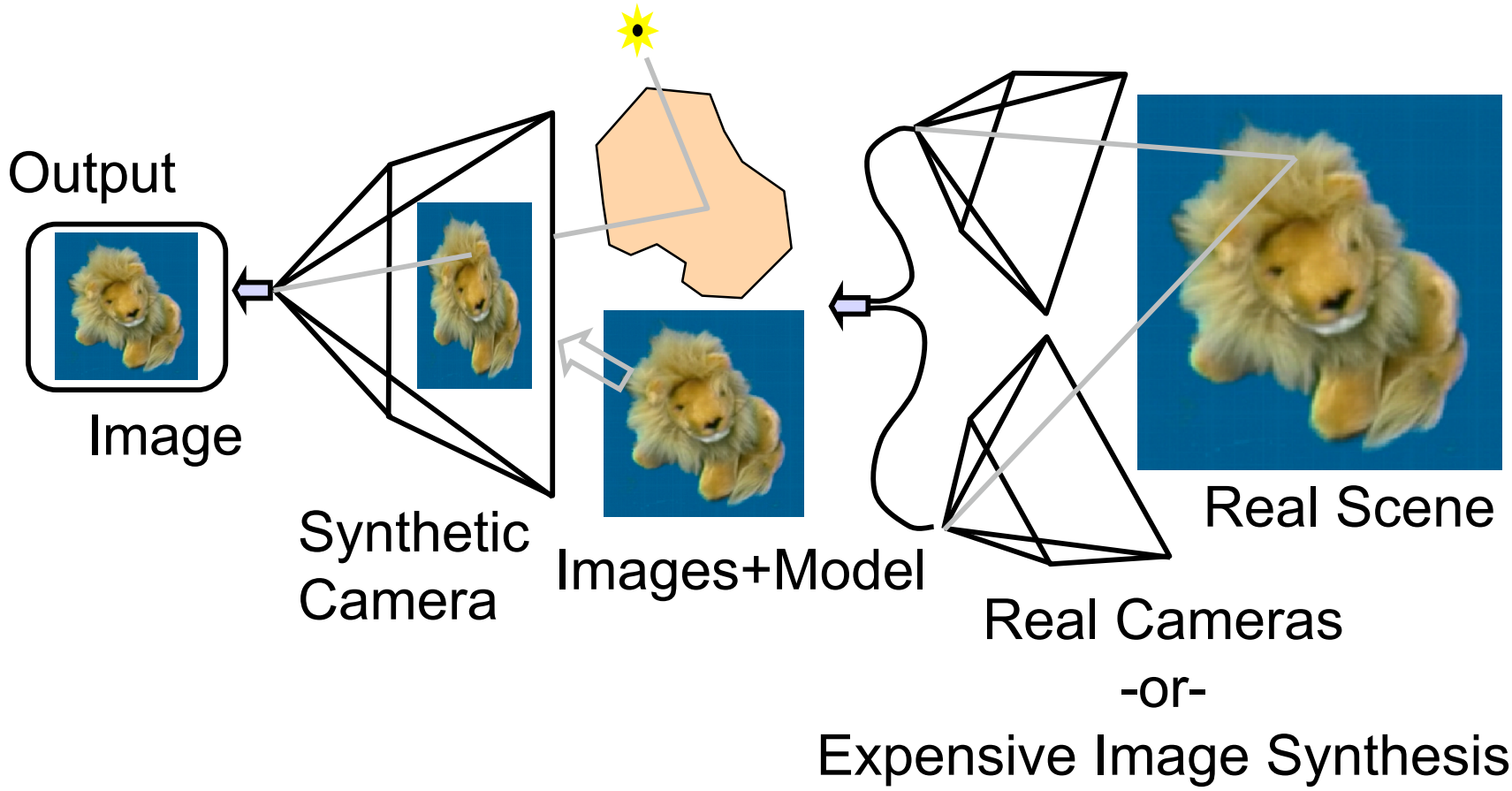


Image-Based Rendering

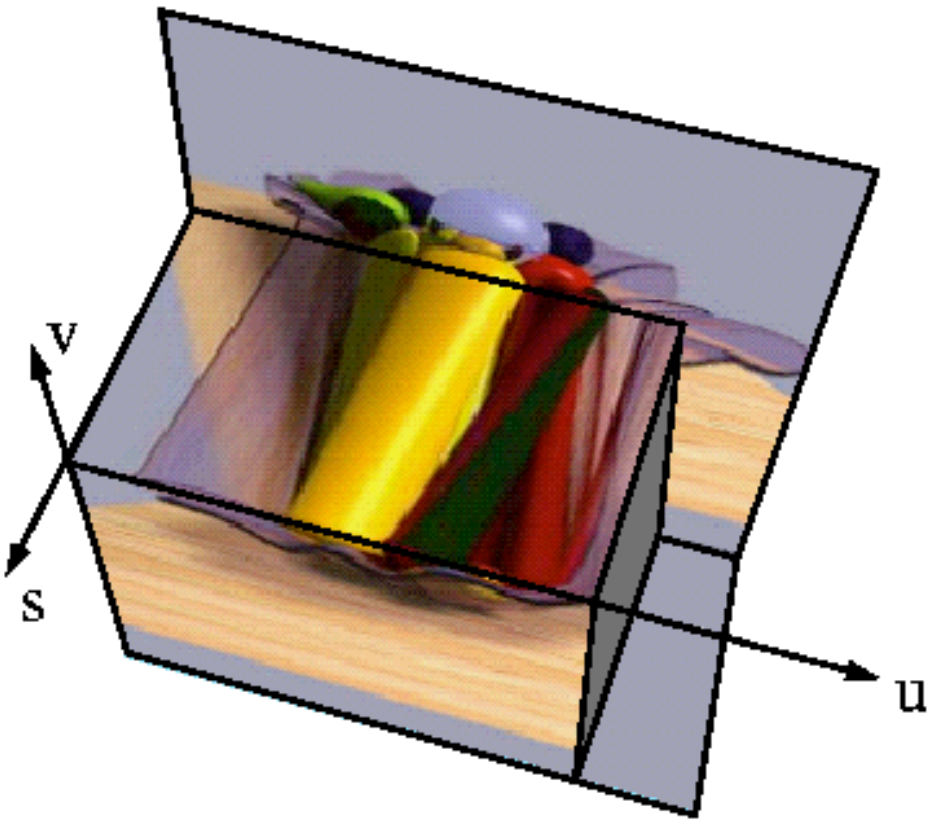
Combine different kinds of images:

- High dynamic range [SG'97…SG'03]
- Flash + Non-Flash [SG'04]
- Photomontage [SG'04]
- Interactive segmentation [Intelligent Scissors, SG'95…Grab Cut & Lazy Snapping, SG'04]
- Natural image matting [CVPR'00…SG'04]
- Image Analogies [SG'00]

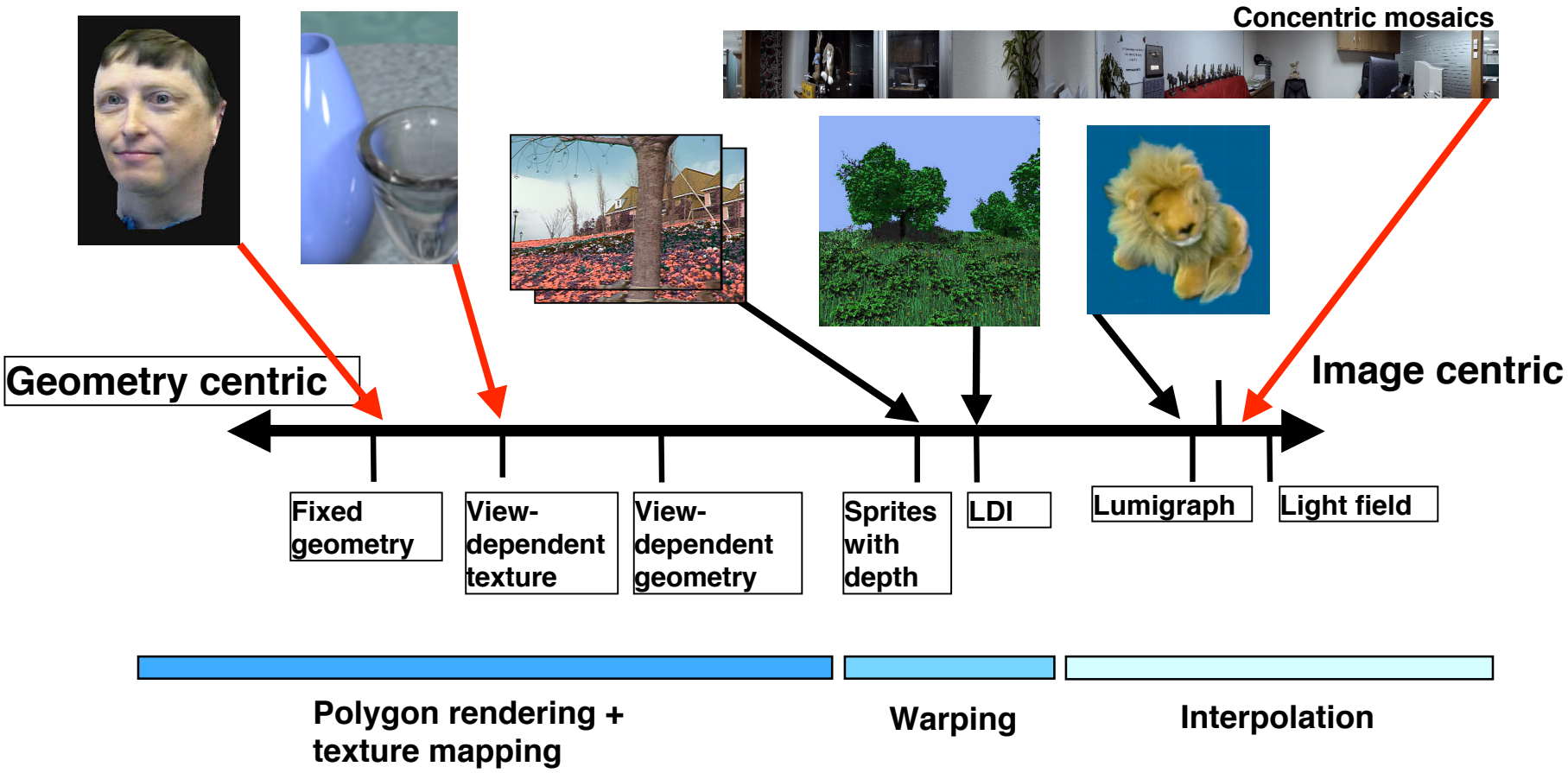
Lumigraph - Ray Space

Image effects:

- parallax
- occlusion
- transparency
- highlights



Graphics/Imaging Continuum



Video-Based Rendering

Image-Based Rendering:

- render from (real-world) images for efficiency, quality, and photo-realism

Video-Based Rendering

- use *video* instead of still images for dynamic elements and source footage
- generate *computer video* instead of *computer graphics*

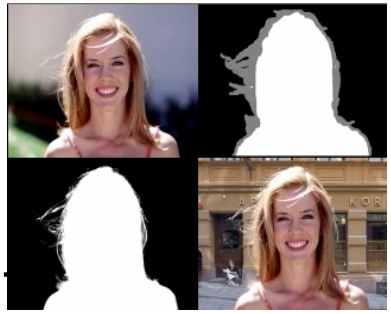
VBR Examples

Facial animation

- Video Rewrite, …

Layer/matte extraction

- Video Matting, …



Dynamic (stochastic) element

- Video Textures, …

3-D world navigation

- Image-Based Realities, …

Video Input Driven Animation

Facial animation

Modeling from still images

- [Pighin *et al.*, SG'98]

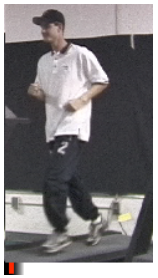
Lip-synching from video

- Video Rewrite
  [Bregler *et al.*, SG'97]
- [Ezzat *et al.*, SG'02]

Video Textures

How can we turn a short *video clip*
into an ∞ amount of continuous video?

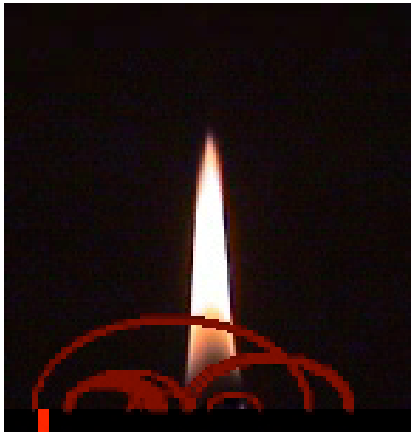
- dynamic elements in 3D games and presentations
- alternative to 3D graphics animation?

[Schödl, Szeliski, Salesin, Essa, SG'2000]

Video Textures

Find cyclic structure in the video



(Optional) region-based analysis

Play frames with random shuffle

Smooth over discontinuities (morph)

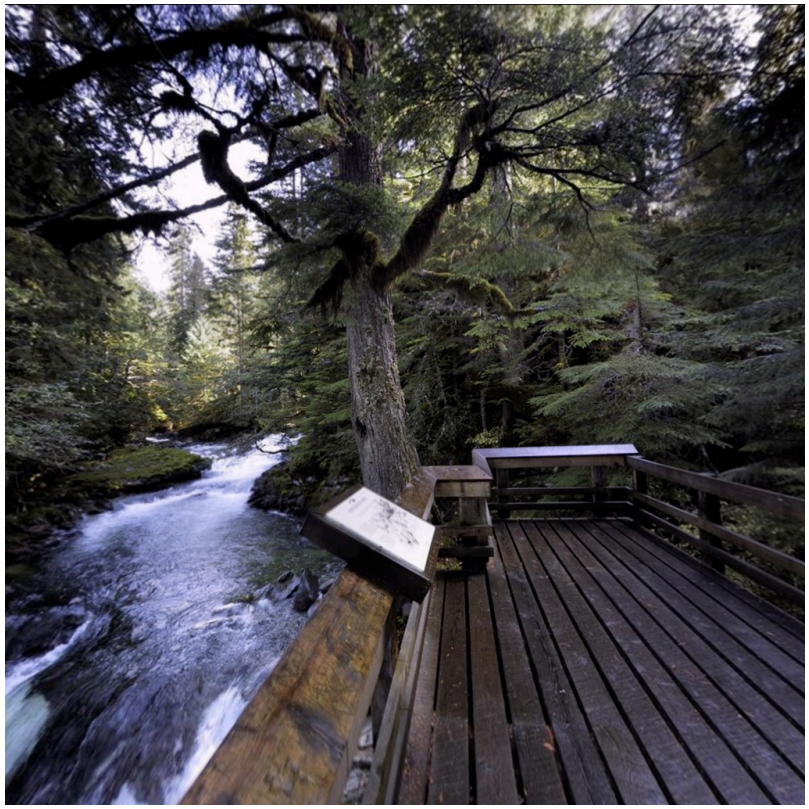
Crossfading and morphing

Jump Cut Crossfade Morph

Video portrait

Dynamic scene element

Live waterfall in static panorama



Interactive fish

A complete animation

Video-Based Rendering

Use *video* instead of still images for dynamic elements and source footage

Generate *computer video* instead of *computer graphics*

Exciting new area for 3D and dynamic *interactivity*

Vision, graphics, and learning

Rather than explicitly *modeling* the dynamic 3D appearance/behavior of the world, *learn* it from lots of sample data

- (lessons learned from speech, AI, …)

The closer you stick to the original data, the greater the *realism*

The more you (automatically) model/abstract, the greater the *control*

Vision, graphics, and learning

Other recent examples:

- motion capture (soup): splice together captured sequences under control [SIGGRAPH'2001]
- image analogies: high-level texture synthesis from sample imagery [Hertzmann *et al.*, SG'00]
- sample-based super-resolution [Freeman]
- face recognition (& hallucination)
- Aaron Herztmann's UW course [2001]

Vision, graphics, and learning

Open problems:

- level of analysis (pixel, object, frame, …)
- automatic segmentation, tracking
- mathematical tools (LDS, Markov Chains, generative models,…)
- perceptual metrics, measures of success
- control: expressiveness, behaviors, intent
- how close can we get to *reality*?