Visual-Geometric 3-D Scene Reconstruction from Uncalibrated Image Sequences

Reinhard Koch and Jan-Michael Frahm

Tutorial at DAGM 2001, München

Multimedia Information Processing Group

Christian-Albrechts-University of Kiel

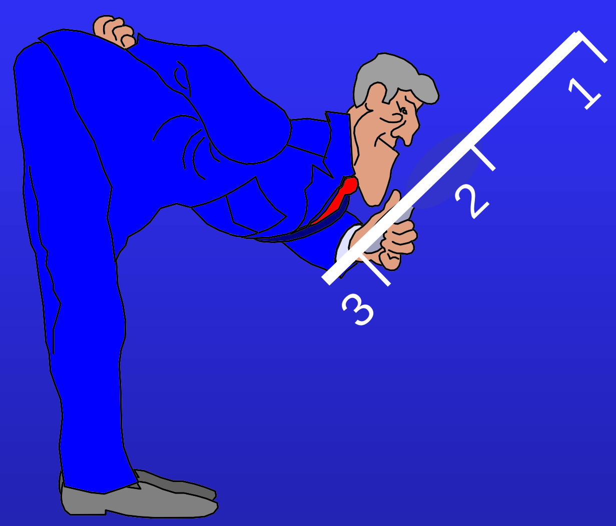
Germany

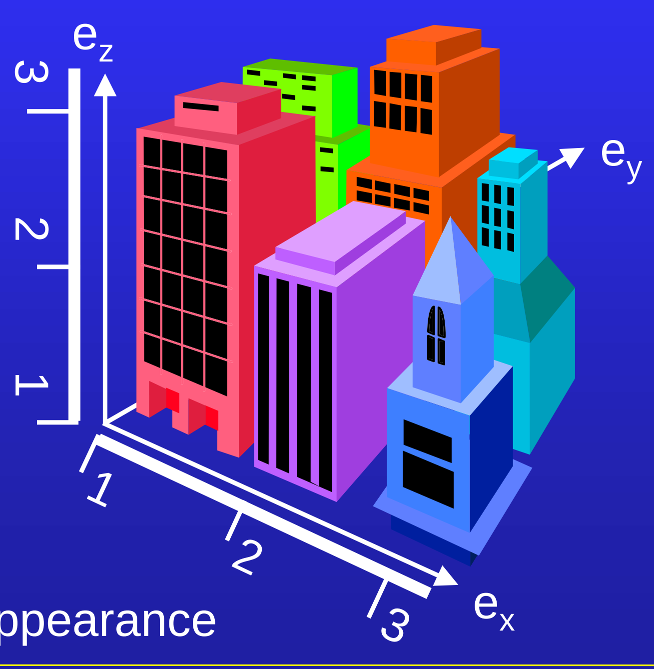
{rk | jmf}@mip.informatik.uni-kiel.de

www.mip.informatik.uni-kiel.de

Scene Reconstruction Method 1

- use ruler to measure scene geometry

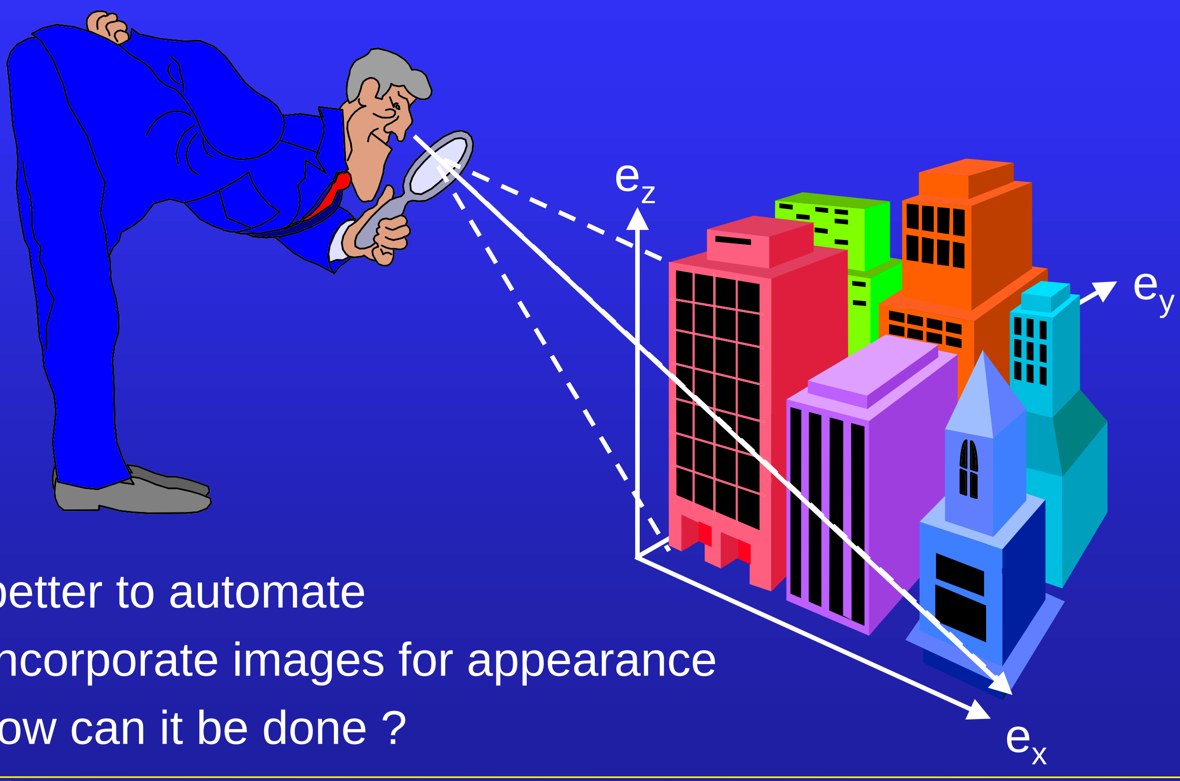




+ precize, absolute geometry
- tedious manual task
- difficult to incorporate visual appearance

Scene Reconstruction Method 2

- measure scene with camera, using image projections!



+ better to automate
+ incorporate images for appearance
- how can it be done ?

Goal of this tutorial

Computer vision enables us to reconstruct highly naturalistic computer models of 3D enviroments from camera images

We may need to extract the camera geometry (calibration), scene structure (surface geometry) as well as the visual appearance (color and texture) of the scene

This tutorial will

– introduce the basic mathematical tools (projective geometry)
– derive models for cameras, image mappings, 3D structure
– give examples for image-based panoramic modeling
– explain geometric and visual models of 3D scene reconstruction

Outline of Tutorial

1. Basics on affine and projective geometry
2. Image mosaicing and panoramic reconstruction

   Coffee break

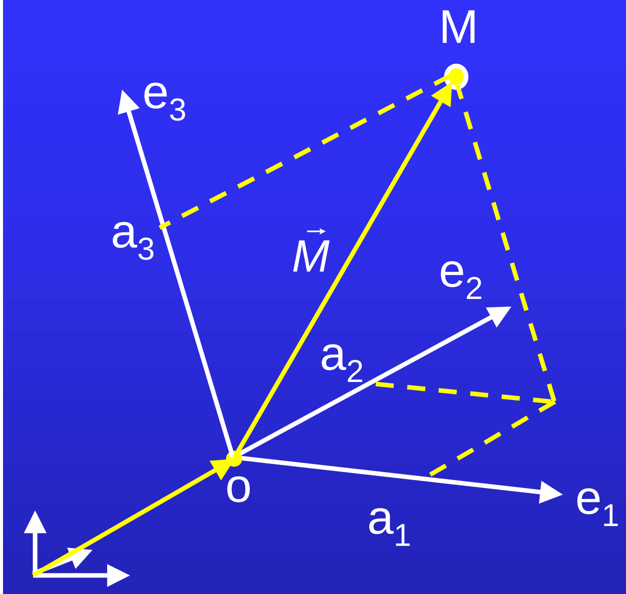
3. 3-D scene reconstruction from multiple views
4. Plenoptic modeling

Demonstrations on mosaicing and 3-D modeling

Part 1: Basics on affine and projective geometry

- Affine geometry
  - affine points and homogeneous coordinates
  - affine transformations
- Projective geometry
  - projective points and projective coordinates
  - projective transformations
- Pinhole camera model
  - Projection and sensor model
  - camera pose and calibration matrix
- Image mapping with planar homographies

Affine coordinates



e_i: affine basis vectors

o: coordinate origin

Vector relative to o:

$$\vec{M} = a_1\vec{e}_1 + a_2\vec{e}_2 + a_3\vec{e}_3$$

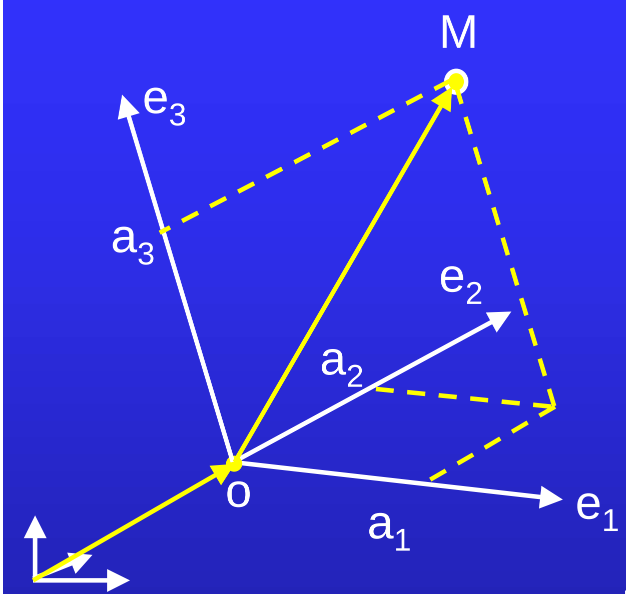
Point in affine coordinates:

$$M = \vec{M} + \vec{o} = a_1\vec{e}_1 + a_2\vec{e}_2 + a_3\vec{e}_3 + \vec{o}$$

Vector: relative to some origin

Point: absolute coordinates

Homogeneous coordinates



Unified notation:

include origin in affine basis

Homogeneous Coordinates of M

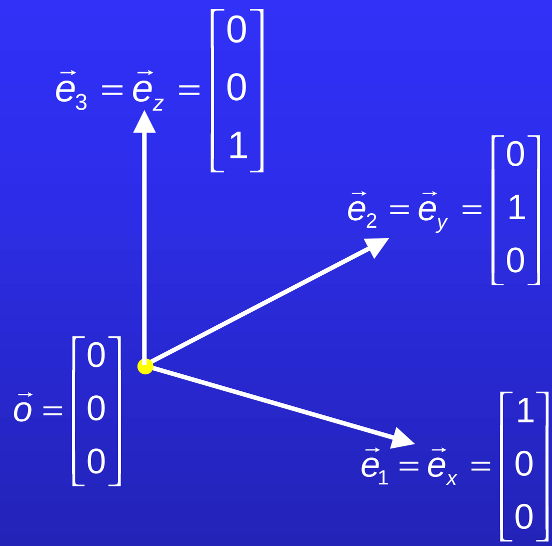
$$M = \begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 & \vec{o} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ 1 \end{bmatrix}$$

Affine basis matrix

Euclidean coordinates

Euclidean coordinates: affine basis is orthonormal

$$e_i^T e_k = \begin{cases} 1 \text{ for } i = k \\ 0 \text{ for } i \neq k \end{cases}$$



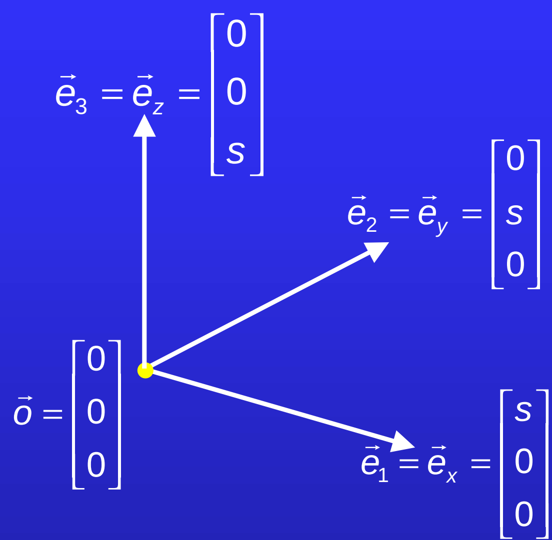
$$\vec{e}_3 = \vec{e}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \vec{e}_2 = \vec{e}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \vec{o} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \vec{e}_1 = \vec{e}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Euclidean basis: $\begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 & \vec{o} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Metric coordinates

Metric coordinates: affine basis is orthogonal

$$e_i^T e_k = \begin{cases} s \text{ for } i = k,\ s \neq 0 \\ 0 \text{ for } i \neq k \end{cases}$$



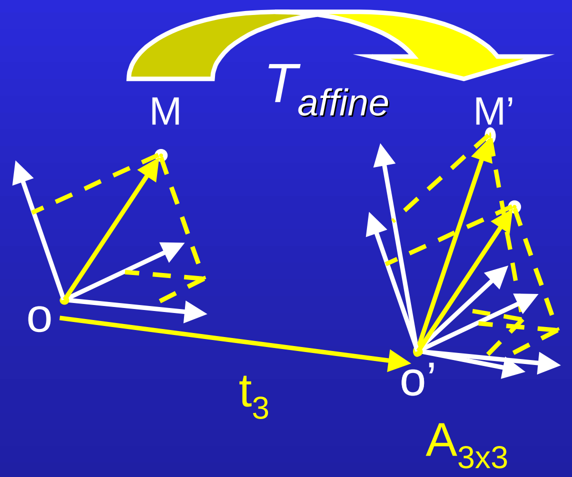
$$\vec{e}_3 = \vec{e}_z = \begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} \quad \vec{e}_2 = \vec{e}_y = \begin{bmatrix} 0 \\ s \\ 0 \end{bmatrix} \quad \vec{o} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \vec{e}_1 = \vec{e}_x = \begin{bmatrix} s \\ 0 \\ 0 \end{bmatrix}$$

Metric basis: $\begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 & \vec{o} \\ 0 & 0 & 0 & 1 \end{bmatrix} = s \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Affine Transformation

Transformation T_{affine} combines linear mapping and coordinate shift in homogeneous coordinates

- Linear mapping with A_{3x3} matrix
- coordinate shift with t_3 translation vector



$$M' = T_{affine} M = \begin{bmatrix} A_{3x3} & & & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} M$$

Linear mapping with A_{3x3}:

- Euclidean rigid rotation
- metric isotropic scaling
- affine skew and anisotropic scaling
- preserves parallelism and length ratio

Properties of affine transformation

$$M' = T_{affine} M = \begin{bmatrix} A_{3x3} & & & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} M \qquad T_{affine} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

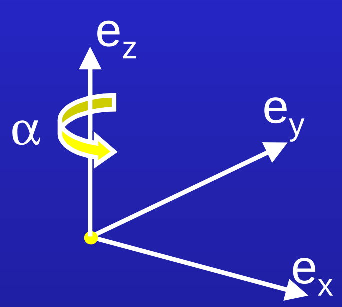
- Parallelism is preserved
- ratios of length, area, and volume are preserved
- Transformations can be concatenated:

if $M_1 = T_1 M$ and $M_2 = T_2 M_1 \Rightarrow M_2 = T_2 T_1 M = T_{21} M$

Special transformation: Rotation

$$T_{Rotation} = \begin{bmatrix} & & & 0 \\ & R_{3x3} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rigid transformation: Angles and length preserved
- R is orthonormal matrix defined by three angles around three coordinate axes

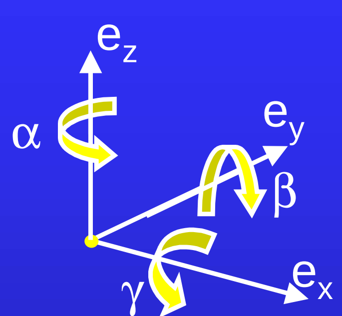


$$R_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation with angle α around e_z

Special transformation: Rotation

- Rotation around the coordinate axes can be concatenated:



$$R = R_z R_y R_x$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

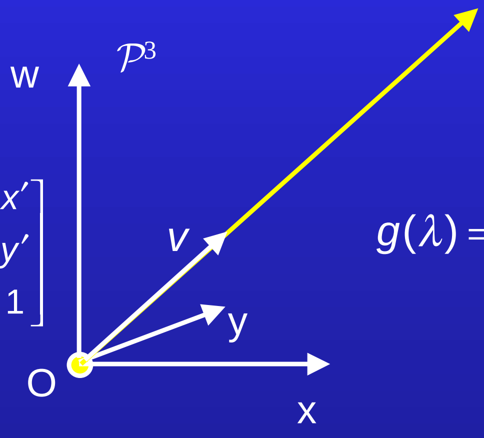
Inverse of rotation matrix:

$$R^{-1} = R^T$$

$$R_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Projective geometry

- Projective space $\mathcal{P}^3$ is space of rays emerging from *O*
  - view point *O* forms projection center for all rays
  - rays *v* emerge from viewpoint into scene
  - *ray g* is called projective point, defined as scaled *v*: $g=\lambda v$

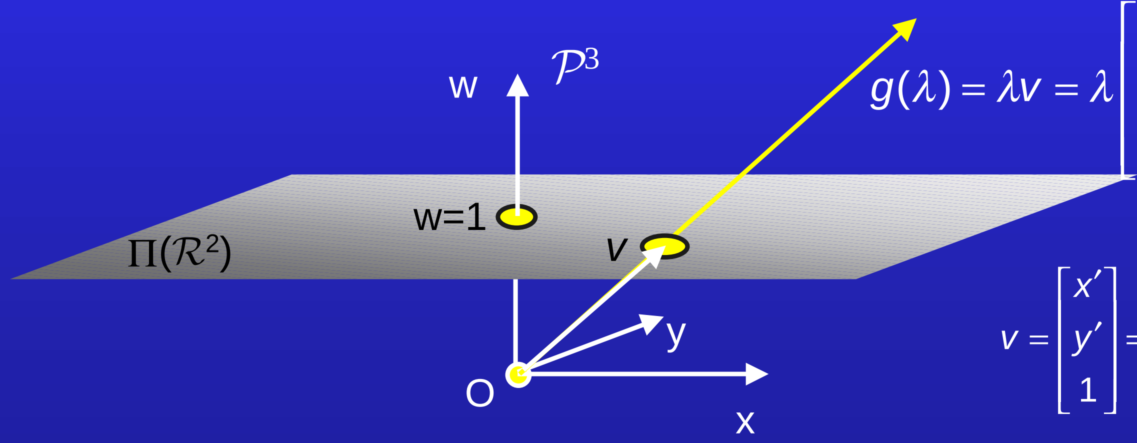


$$v = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$g(\lambda) = \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \lambda v, \lambda \in \Re \neq 0$$

Projective and homogeneous points

- Given: Plane Π in $\mathcal{R}^2$ embedded in $\mathcal{P}^3$ at coordinates *w*=1
  - viewing ray *g* intersects plane at *v* (homogeneous coordinates)
  - all points on ray *g* project onto the same homogeneous point *v*
  - projection of *g* onto Π is defined by scaling $v=g/\lambda = g/w$

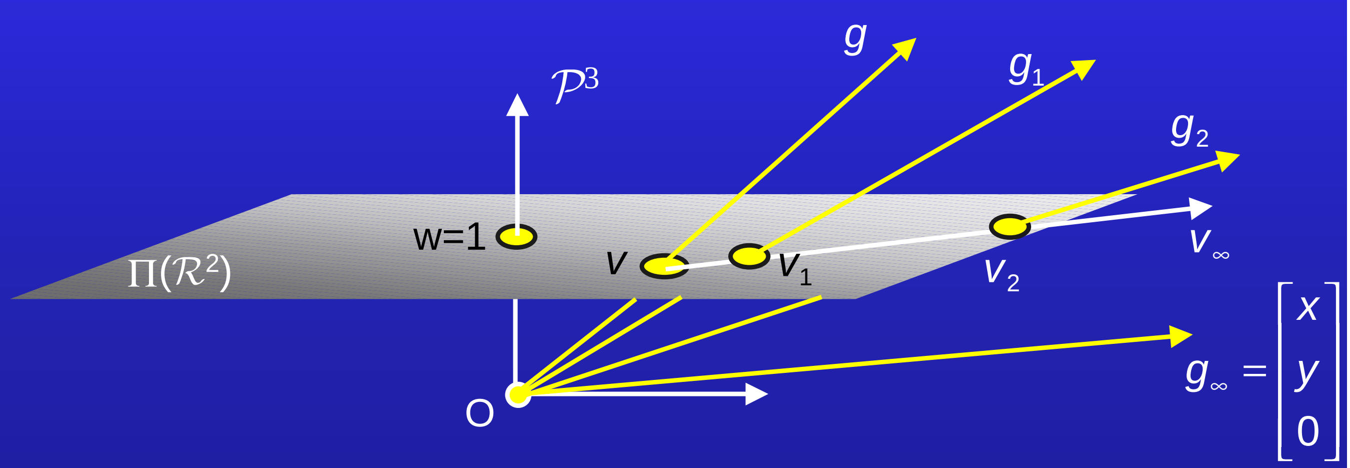


$$g(\lambda) = \lambda v = \lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$v = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \frac{1}{w} g = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \\ 1 \end{bmatrix}$$

Finite and infinite points

- All rays *g* that are not parallel to Π intersect at an affine point *v* on Π.
- The ray *g*(w=0) does not intersect Π. Hence v_∞ is not an affine point but a direction. Directions have the coordinates $(x,y,z,0)^T$
- Projective space combines affine space with infinite points (directions).



Relation between affine and projective points

- Affine space is embedded into projective space $(x,y,z,w)^T$ as hyperplane $\Pi = (x,y,z,1)^T$.
- Homogeneous coordinates map a projective point onto the affine hyperplane by scaling to $w=1$.
- Projective points with $w \neq 0$ are projected onto Π by scaling with 1/w.
- Projective points with w=0 form the infinite hyperplane $\Pi_\infty = (x,y,z,0)^T$. They are not part of Π.

Affine and projective transformations

- Affine transformation leaves infinite points at infinity

$$M_\infty' = T_{affine} M_\infty \quad \Rightarrow \begin{bmatrix} X' \\ Y' \\ Z' \\ 0 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix}$$

- Projective transformations move infinite points into finite affine space

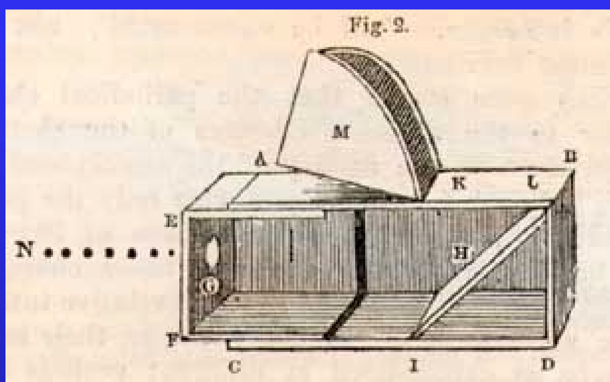
$$M' = T_{projective} M_\infty \quad \Rightarrow \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} X' \\ Y' \\ Z' \\ w' \end{bmatrix} = \lambda \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix}$$

Example: Parallel lines intersect at the horizon (line of infinite points).
We can see this intersection due to perspective projection!

Pinhole Camera (Camera obscura)

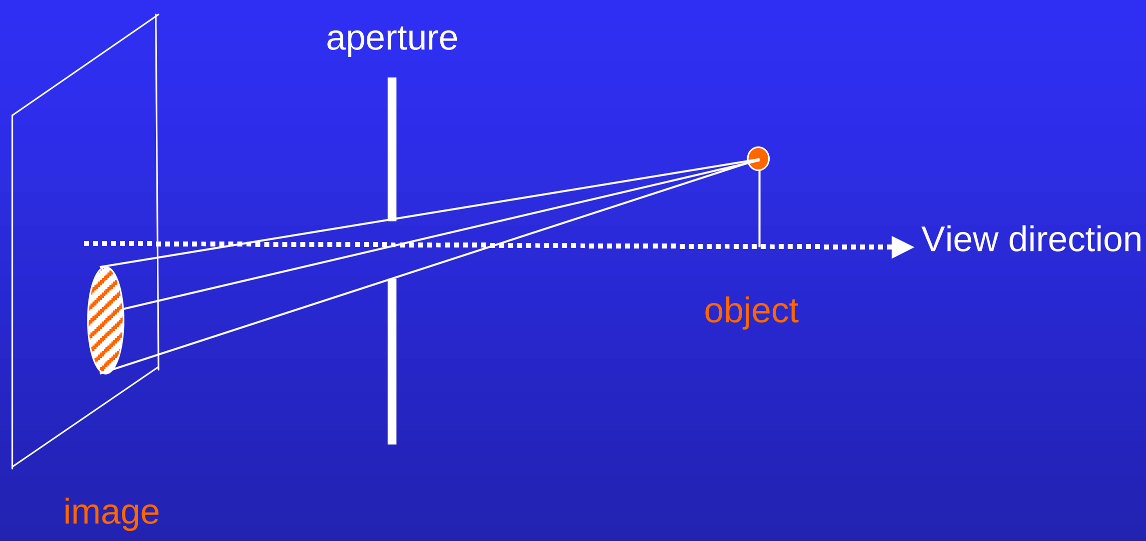
Camera obscura

(France, 1830)

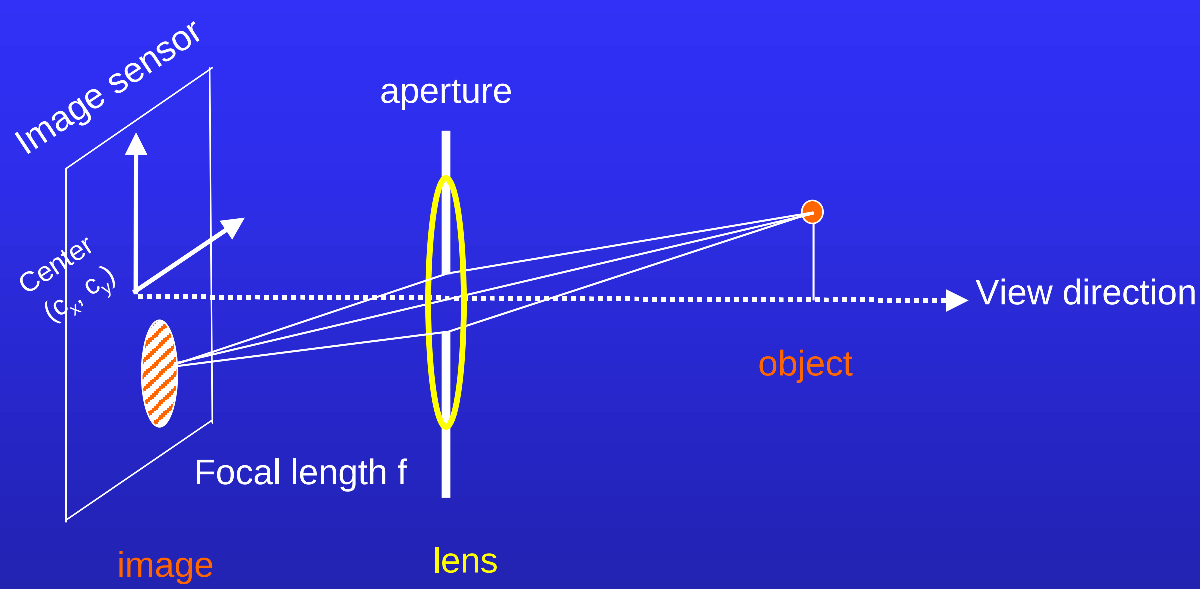


Interior of camera obscura
(Sunday Magazine, 1838)

Pinhole camera model

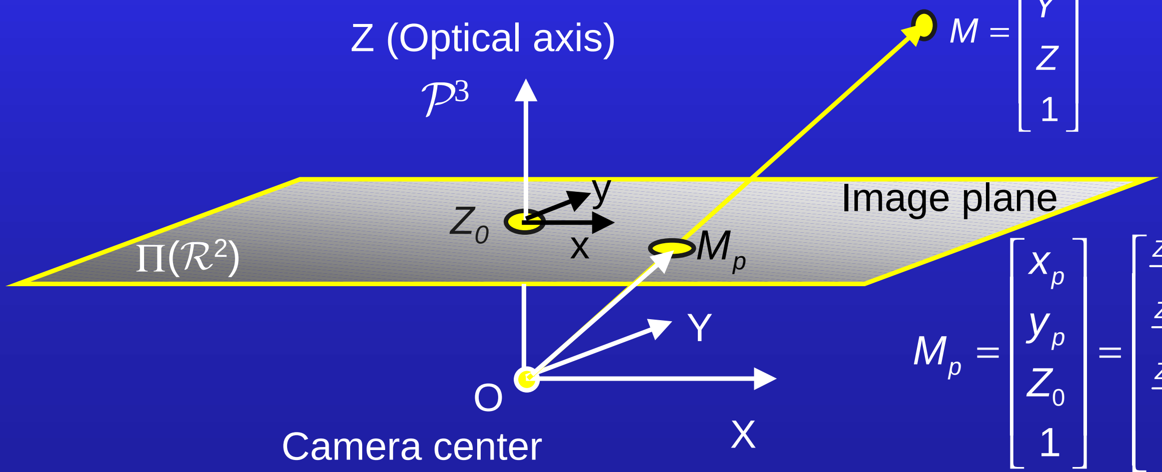


Pinhole camera model



Perspective projection

- Perspective projection in $\mathcal{P}^3$ models pinhole camera:
  - scene geometry is affine $\mathcal{R}^3$ space with coordinates $M=(X,Y,Z,1)^T$
  - camera focal point in $O=(0,0,0,1)^T$, camera viewing direction along Z
  - image plane (x,y) in $\Pi(\mathcal{R}^2)$ aligned with (X,Y) at Z= Z_0
  - Scene point M projects onto point M_p on plane surface

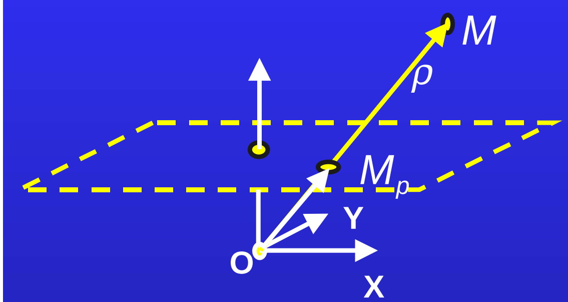


$$M=\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$M_p=\begin{bmatrix} x_p \\ y_p \\ Z_0 \\ 1 \end{bmatrix}=\begin{bmatrix} \frac{Z_0 X}{Z} \\ \frac{Z_0 Y}{Z} \\ \frac{Z_0 Z}{Z} \\ 1 \end{bmatrix}=\frac{Z_0}{Z}\begin{bmatrix} X \\ Y \\ Z \\ \frac{Z}{Z_0} \end{bmatrix}$$

Projective Transformation

- Projective Transformation maps M onto M_p in $\mathcal{R}^3$ space



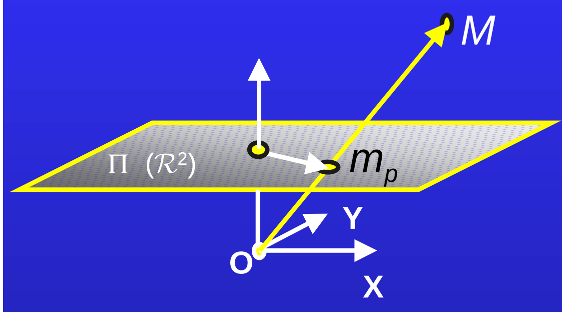
$$\rho M_p = T_p M \Rightarrow \rho\begin{bmatrix} x_p \\ y_p \\ Z_0 \\ 1 \end{bmatrix}=\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{Z_0} & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\rho=\frac{Z}{Z_0}=\text{projective scale factor}$$

- Projective Transformation linearizes projection

Perspective Projection

Dimension reduction from $\mathcal{R}^3$ into $\mathcal{R}^2$ by projection onto $\Pi(\mathcal{R}^2)$



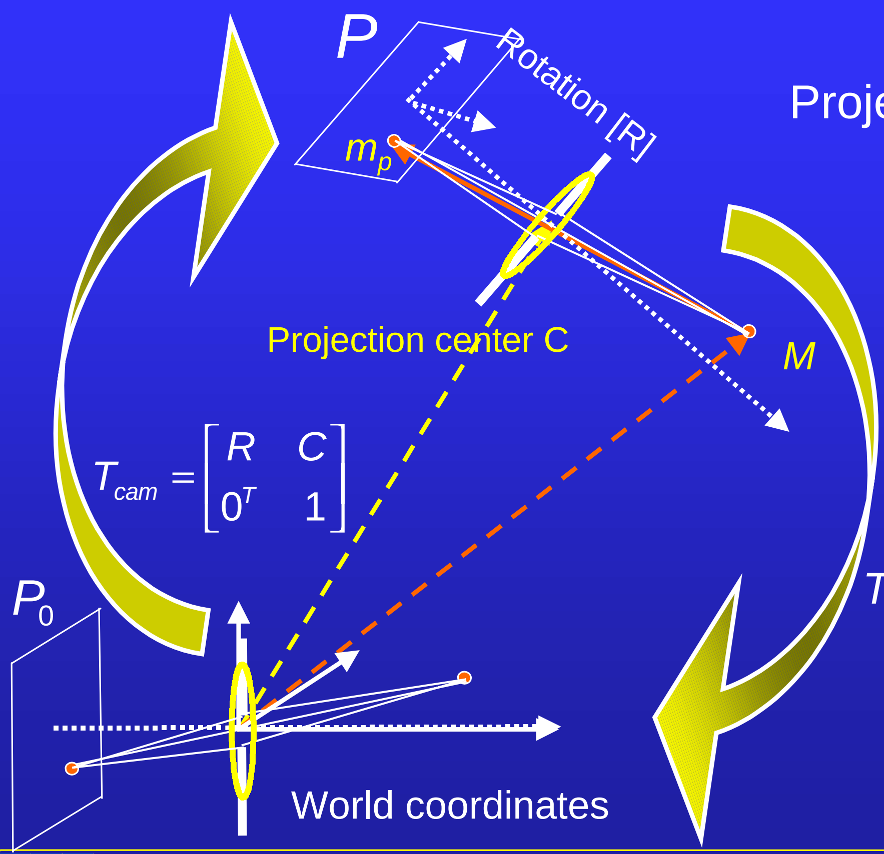
$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ Z_0 \\ 1 \end{bmatrix} \Rightarrow m_p = D_p M_p$$

Perspective projection P_0 from $\mathcal{R}^3$ onto $\mathcal{R}^2$:

$$\rho m_p = D_p T_p M = P_0 M \Rightarrow \rho \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{z_0} & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad \rho = \frac{Z}{Z_0}$$

Projection in general pose

Projection: $\rho m_p = PM$



$$T_{cam} = \begin{bmatrix} R & C \\ 0^T & 1 \end{bmatrix}$$

$$T_{scene} = T_{cam}^{-1} = \begin{bmatrix} R^T & -R^T C \\ 0^T & 1 \end{bmatrix}$$

Image plane and image sensor

- A sensor with picture elements (Pixel) is added onto the image plane

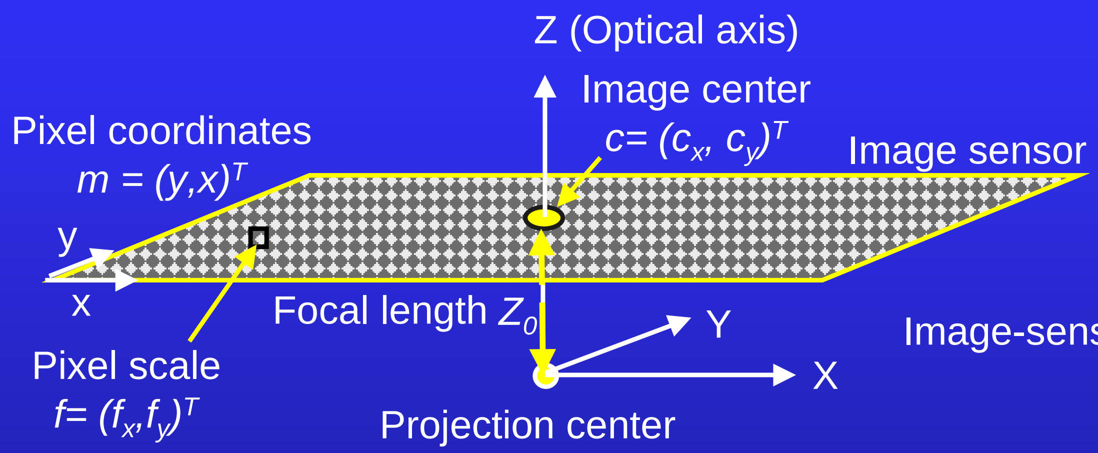


Image-sensor mapping:

$$m = Km_p$$

- Pixel coordinates are related to image coordinates by affine transformation K with five parameters:
  - Image center c at optical axis
  - distance Z_p (focal length) and Pixel size scales pixel resolution f
  - image skew s to model angle between pixel rows and columns

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Projection matrix *P*

- Camera projection matrix P combines:
  - inverse affine transformation T_a^{-1} from general pose to origin
  - Perspective projection P_0 to image plane at $Z_0 = 1$
  - affine mapping K from image to sensor coordinates

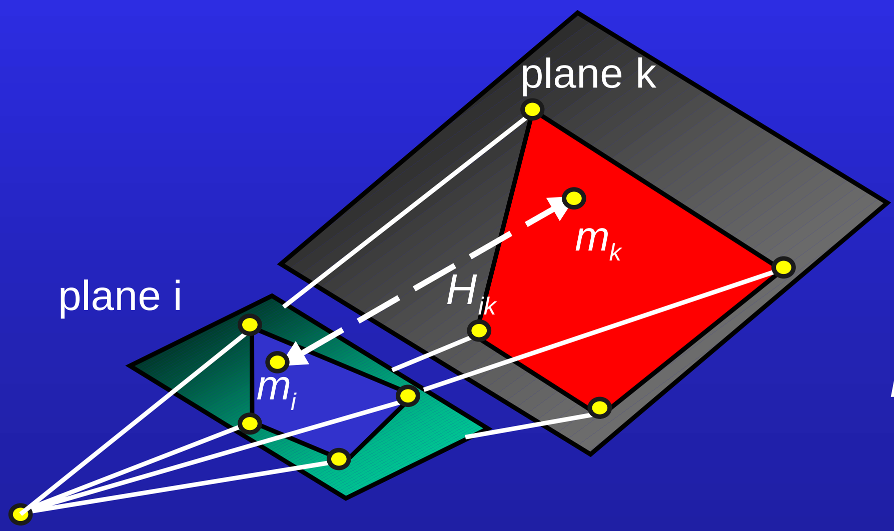
scene pose transformation: $T_{scene} = \begin{bmatrix} R^T & -R^T C \\ 0^T & 1 \end{bmatrix}$

projection: $P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix}$ sensor calibration: $K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$

$$\Rightarrow \rho m = PM, \quad P = KP_0 T_{scene} = K\begin{bmatrix} R^T & -R^T C \end{bmatrix}$$

The planar homography H

- The 2D projective transformation H_{ik} is a planar homography
  - maps any point on plane i to corresponding point on plane k
  - defined up to scale (8 independent parameters)
  - defined by 4 corresponding points on the planes with not more than any 2 points collinear



$$m_k \simeq H_{ik} m_i$$

$$H_{ik} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix}$$

Estimation of H from image correspondences

- H_{ik} can be estimated linearily from corresponding point pairs:
  - select 4 corresponding point pairs, if known noise-free
  - select N>4 corresponding point pairs, if correspondences are noisy
  - compute H such that correspondence error f is minimized

Projective mapping:

$$m_k = \rho_k \begin{bmatrix} x_k \\ y_k \\ w \end{bmatrix} = H m_i = \begin{bmatrix} h_1 x_i + h_2 y_i + h_3 \\ h_4 x_i + h_5 y_i + h_6 \\ h_7 x_i + h_8 y_i + 1 \end{bmatrix}$$

Image coordinate mapping:

$$x_k = \frac{h_1 x_i + h_2 y_i + h_3}{h_7 x_i + h_8 y_i + 1}$$

$$y_k = \frac{h_4 x_i + h_5 y_i + h_6}{h_7 x_i + h_8 y_i + 1}$$

Error functional f:

$$f = \sum_{n=0}^{N} (m_{k,n} - H_{ik} m_{i,n})^2 \Rightarrow \min!$$

$$H_{ik} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix}$$

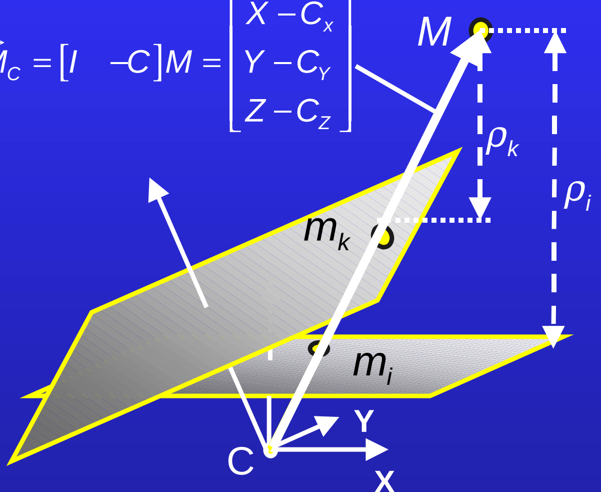
Image mapping with homographies

- Homographies are 2D projective transformations H_{3x3}
- Homographies map points between planes
- 2D homographies can be used to map images between different camera views for three equivalent cases:
  - (a) all cameras share the same view point C_i = C, or
  - (b) all scene points are at (or near to) infinity, or
  - (c) the observed scene is planar.
- The 2D homography is independent of 3D scene structure!

(a) Image mapping for single view point

- Camera with fixed projection center: $C_i = C$
- Camera rotates freely with R_i and changing calibration K_i

$$\vec{M}_C = [I \quad -C]M = \begin{bmatrix} X - C_x \\ Y - C_Y \\ Z - C_Z \end{bmatrix}$$



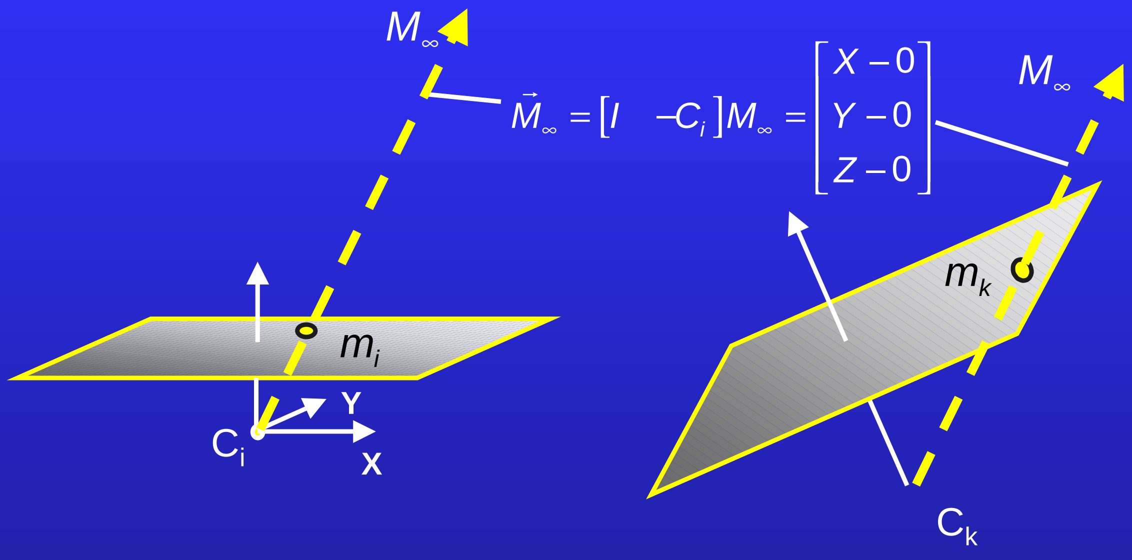
$$\rho_i m_i = P_i M = K_i \left[R_i^T \quad -R_i^T C_i \right] M$$
$$= K_i R_i^T [I \quad -C] M = K_i R_i^T \vec{M}_C$$
$$\rho_k m_k = K_k R_k^T [I \quad -C] M = K_k R_k^T \vec{M}_C$$
$$\Rightarrow \vec{M}_C = R_i K_i^{-1} \rho_i m_i = R_k K_k^{-1} \rho_k m_k$$

$$\boxed{\rho_k m_k = K_k R_k^{-1} R_i K_i^{-1} \rho_i m_i = \rho_i H_{ik} m_i}$$

- H_{ik} is a planar projective 3x3 transformation that maps points m_i on plane i to points m_k on plane k

(b) Image mapping for infinite scene with M_∞

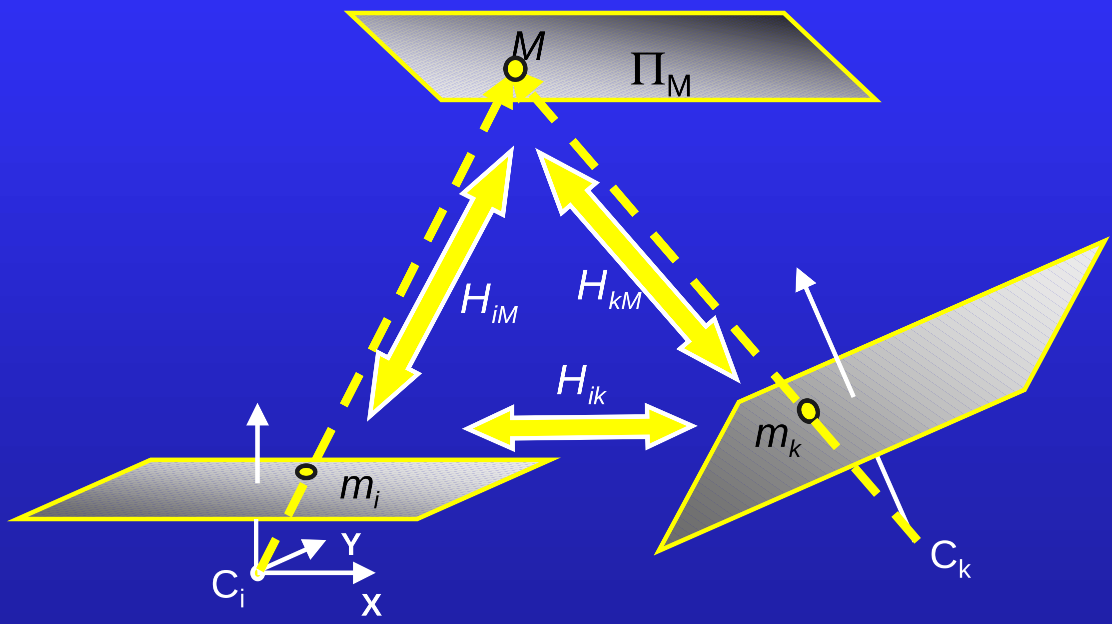
- All scene points are at or near infinity: M_∞ are points on Π_∞
- Camera rotates freely with R_i and changing calibration K_i



$$\vec{M}_\infty = R_i K_i^{-1} \rho_i m_i = R_k K_k^{-1} \rho_k m_k \qquad \Rightarrow \rho_k m_k = K_k R_k^{-1} R_i K_i^{-1} \rho_i m_i = \rho_i H_{ik} m_i$$

(c) Image mapping of planar scene Π_M

- All scene points are at on plane Π_M
- Camera is completely free in K,R,C



Transfer between images i,k over Π_M: $H_{ik} = H_{iM} H_{kM}^{-1}$

Part 2:
Mosaicing and Panoramic Images

- why mosaicing?
- geometries constraints for mosaicing
- mosaicing
- projective mainfolds
- stereo mosaicing

Why mosaicing?

- cameras field of view is always smaller than human field of view
- large objects can‘t be captured in a single picture

Solutions

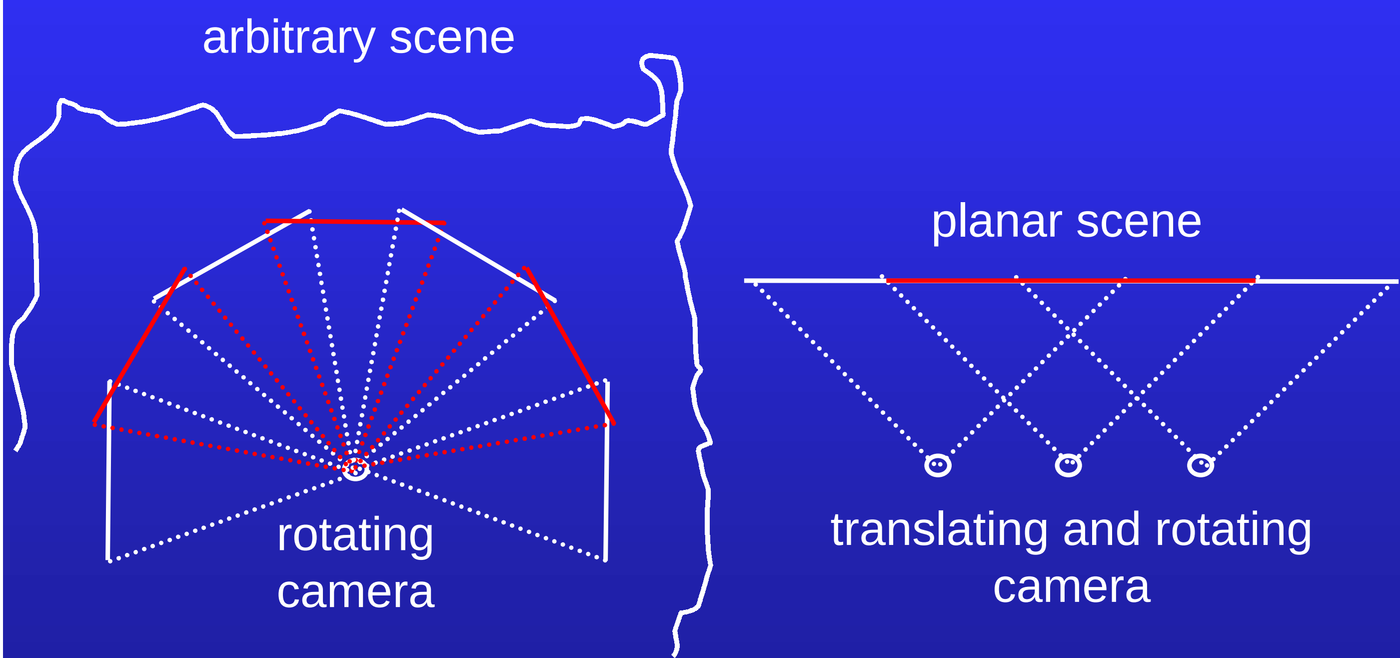
- devices with wide field of view
  - fish-eye lenses (distortions, decreases quality)
  - hyper- and parabolic optical devices (lower resolution)
- image mosaicing

What is mosaicing?

definition of mosaicing : Matching multiple images by aligning and pasting images to a wilder field of view image. Features that continue over the images must be "zipped" together, and the frame edges dissolved.

Kang et. al. (ICPR 2000)

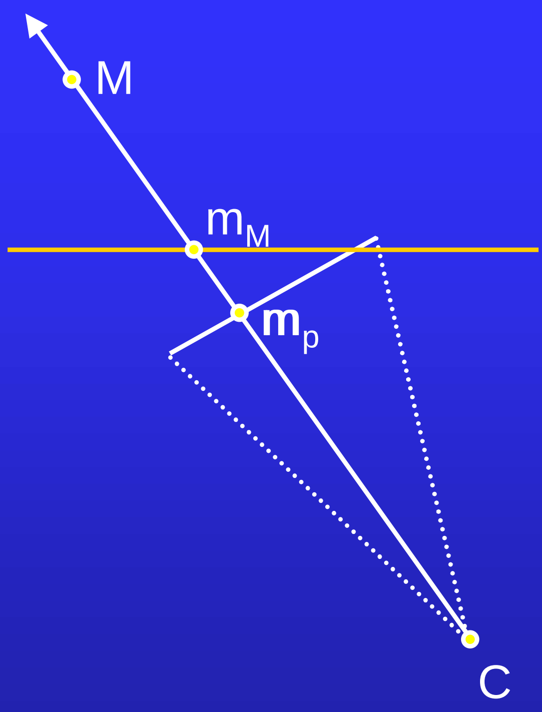
Geometries of mosaic aquisition



Steps of mosaicing

- **Image alignment:** estimation of homography *H* for each image pair
- **Image cut and paste:** selection of colorvalue for each mosaic pixel
- **Image blending:** overcome of intensity differences between images

Forward mapping to plane



pixel to direction

$$(P_0 T_{scene})^{-1} \underbrace{K^{-1}\mathbf{m}}_{m_p} = T_{scene}^{-1} \begin{bmatrix} I \\ 0^T \end{bmatrix} K^{-1} m$$

$$= M_\infty$$

direction to mosaic

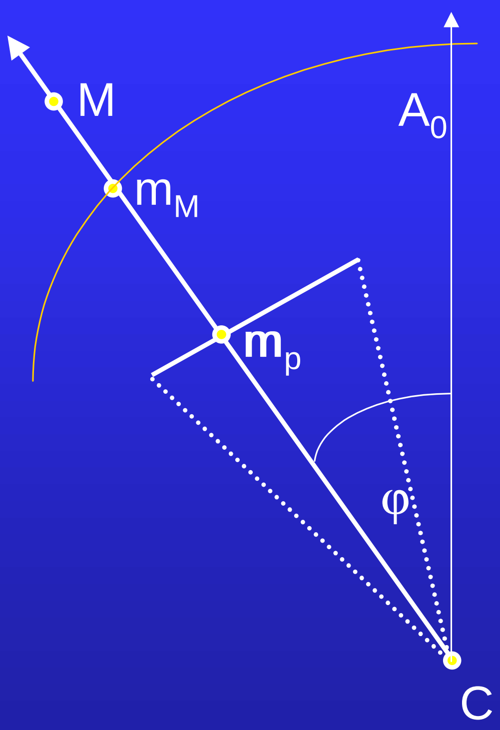
$$\rho m_M = K_{mosaic} P_0 T_{mosaic} M_\infty$$

usually $T_{mosaic} = \begin{bmatrix} I & 0 \\ 0^T & 1 \end{bmatrix}, K_{mosaic} = K$

$$\Rightarrow \rho m_M = \underbrace{K R^T K^{-1}}_{H} m$$

Forward mapping to plane

Forward mapping to sphere



pixel to direction

$$\left(P_0 T_{scene}\right)^{-1} \underbrace{K^{-1}\mathbf{m}}_{m_p} = T_{scene}^{-1}\begin{bmatrix} I \\ 0^T \end{bmatrix} K^{-1} m$$

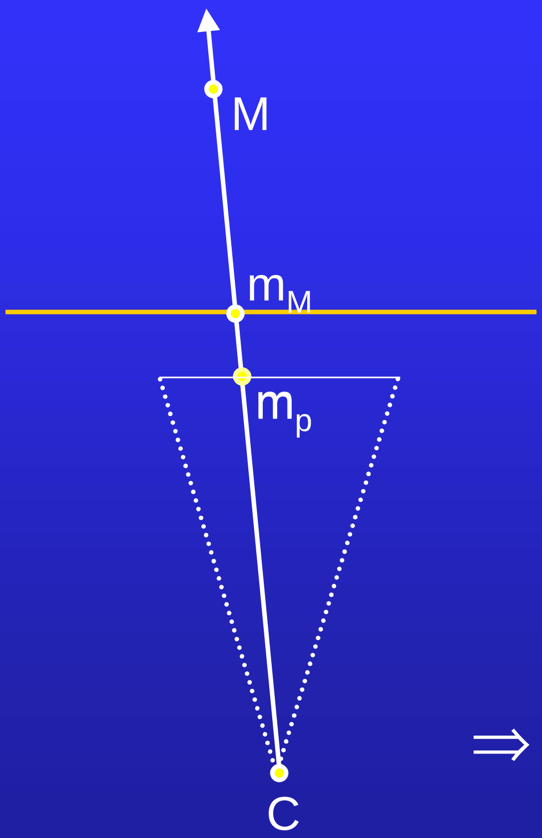
$$= M_\infty$$

direction to polar

$$\varphi = \tan^{-1}\left(\frac{X}{Z}\right)$$

$$\vartheta = \tan^{-1}\left(\frac{Y}{\sqrt{X^2 + Z^2}}\right)$$

Planar scene mapping



pixel to direction

$$\left(P_0 T_{scene}\right)^{-1} \underbrace{K^{-1}\boldsymbol{m}}_{m_p} = T_{scene}^{-1} \begin{bmatrix} I \\ 0^T \end{bmatrix} K^{-1} m$$

$$= M_\infty$$

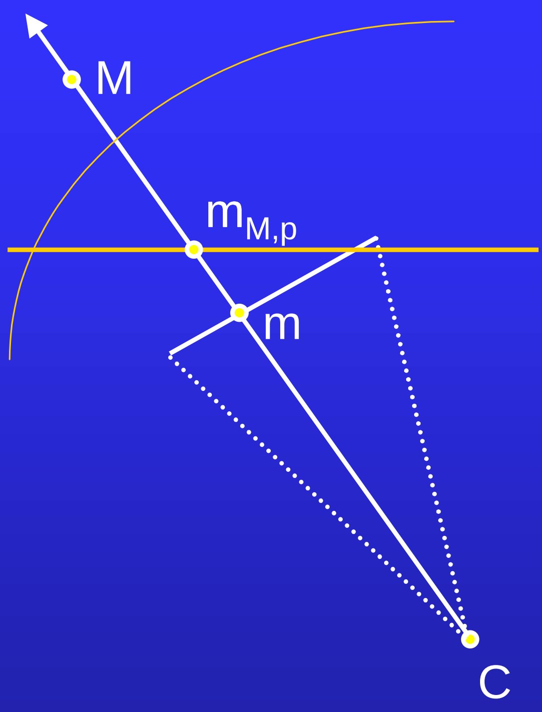
direction to mosaic

$$\rho m_M = K_{mosaic} P_0 T_{mosaic} M_\infty$$

usually $T_{mosaic} = \begin{bmatrix} I & 0 \\ 0^T & 1 \end{bmatrix}$, $K_{mosaic} = K$

$$\Rightarrow \rho m_M = K P_0 \left[R^T \middle| -R^T C \right] P_0^{-1} K^{-1} m$$

Backward mapping from plane



mosaic to direction

$$M_\infty = \left(P_0 T_{mosaic}\right)^{-1} \underbrace{K_{mosaic}^{-1} \boldsymbol{m}_M}_{m_{M,p}}$$

$$= T_{mosaic}^{-1} \begin{bmatrix} I \\ 0^T \end{bmatrix} K^{-1} m_M$$

direction to pixel

$$\rho m = K P_0 T_{scene} M_\infty$$

usually $T_{mosaic} = \begin{bmatrix} I & 0 \\ 0^T & 1 \end{bmatrix}$, $K_{mosaic} = K$

$$\Rightarrow \rho m = \underbrace{K R K^{-1}}_{H^{-1}} m_M$$

Backward mapping from plane

mosaics

image acquisition

- pure rotation around optical center
- planar scene translating and rotating camera

problem

- ensure camera motion constaints

mosaic demo

Live-Demonstration of Quicktime-VR Rotation-Panorama

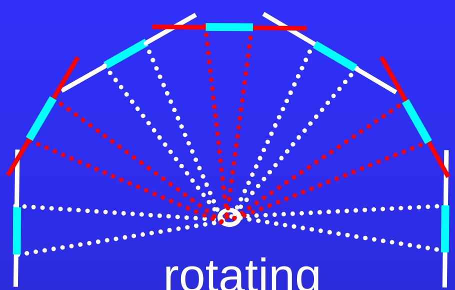
Neo-Gothic Building (Leuven, Belgium)

Rudy Knoops, AVD, K.U. Leuven

Manifold projection

- simulating sampling of scene with a 1-dim. sensor array
- sensor can arbitrary translated and rotated in arbitrary scenes

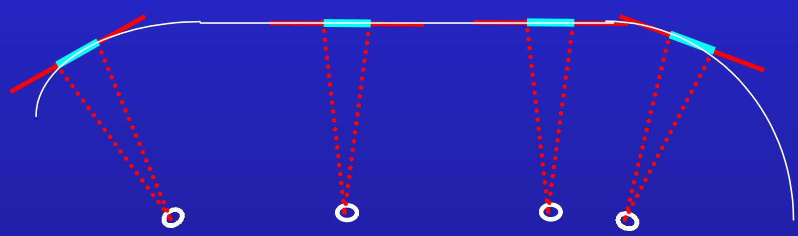
Manifold projection



rotating camera

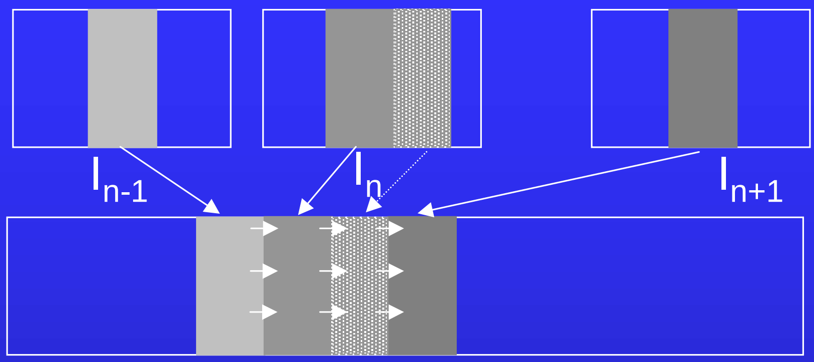
planar scene

pure translating camera



rotating and translating camera

Mosaic from strips



- stripes $F(x,y)=0$ perpendicular to optical flow

$$\Rightarrow \text{normal} \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y} \right)^T \text{ of stripe is parallel to optical flow}$$

- select stripe as close as possible to image center

Manifold projection

- overcomes some of the difficulties of mosaicing
- defined for any camera motion and scene structure
- resolution is the same as image resolution
- simplified computation (real time)
- visually good results

Problem

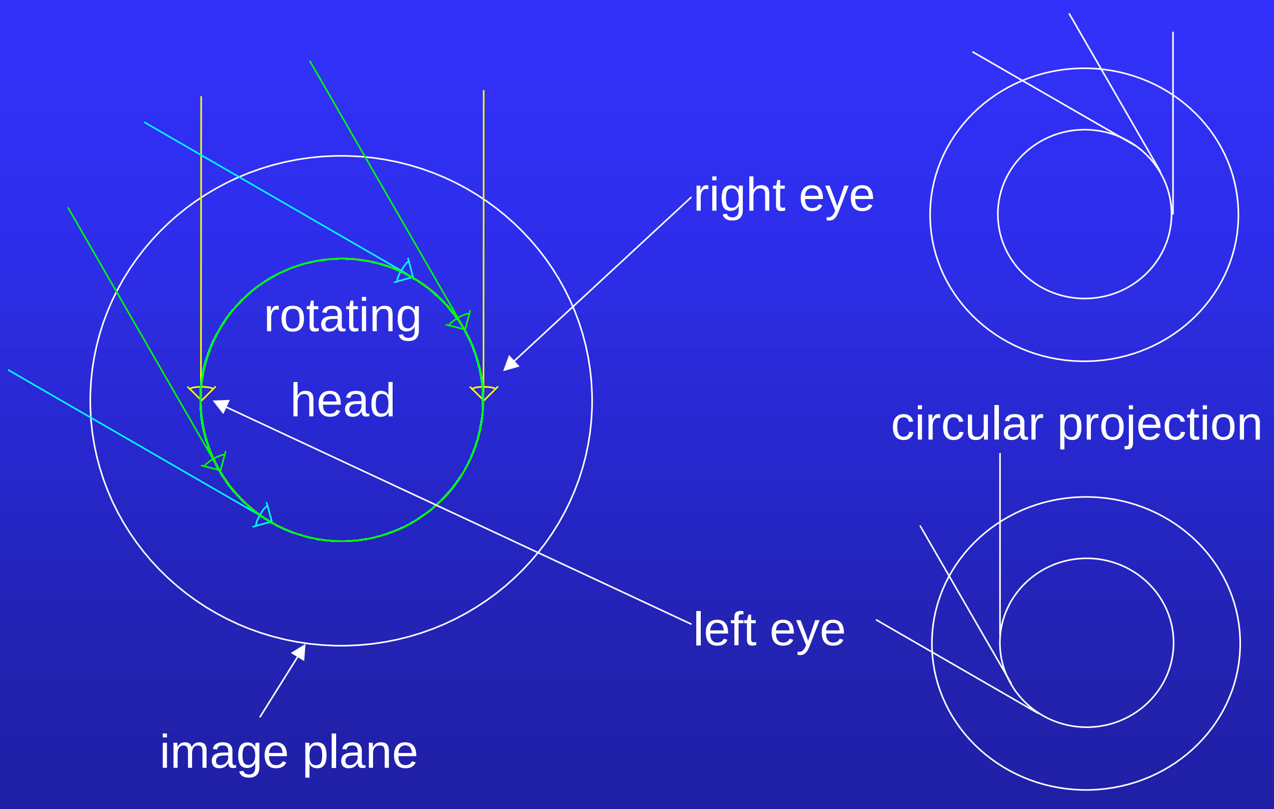
- No metric in mosaic

- Life-Demonstration VideoBrush

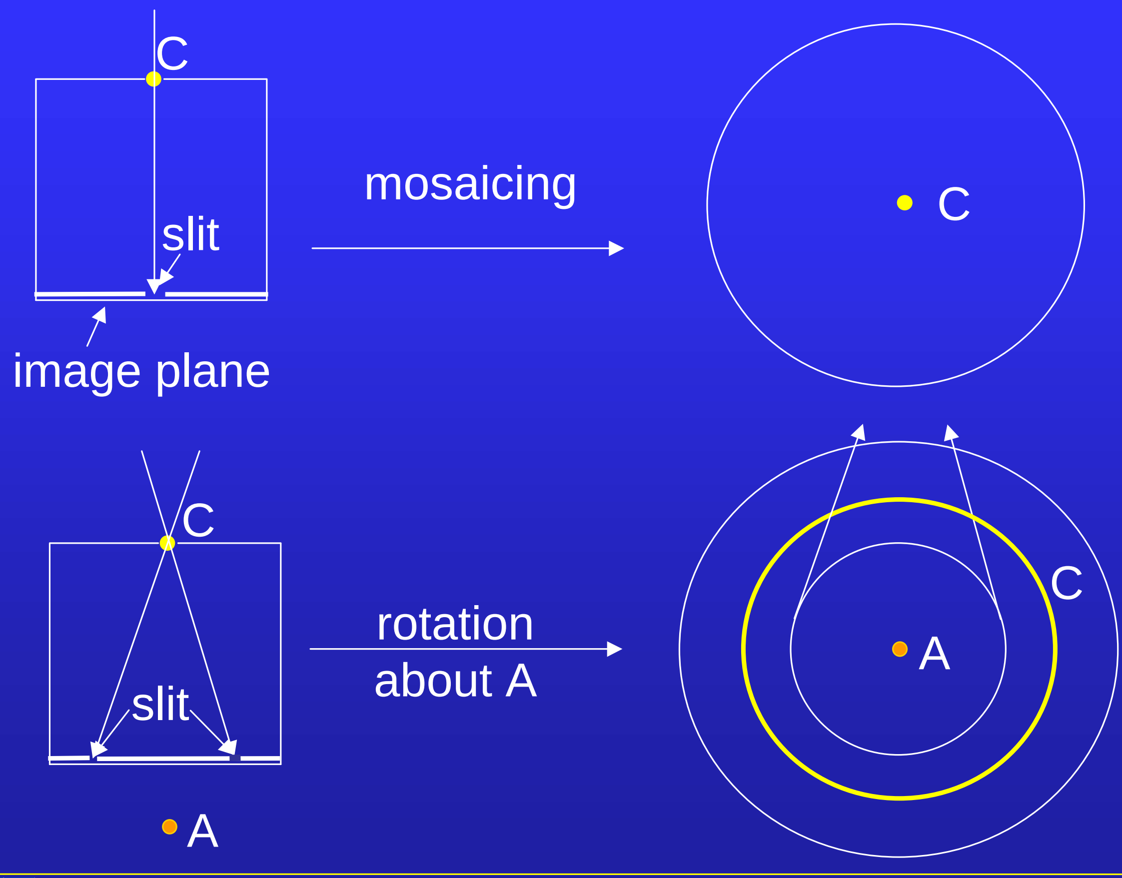
Stereo mosaics

- multiple viewpoint panorama
- for each eye a seperate multiple viewpoint panorama
- constructed from one rotating camera

Stereo perception

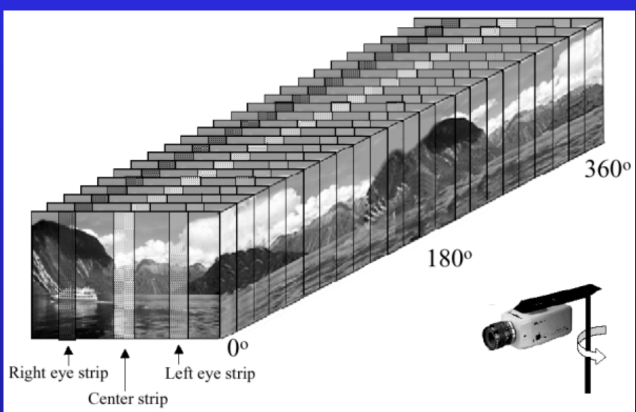


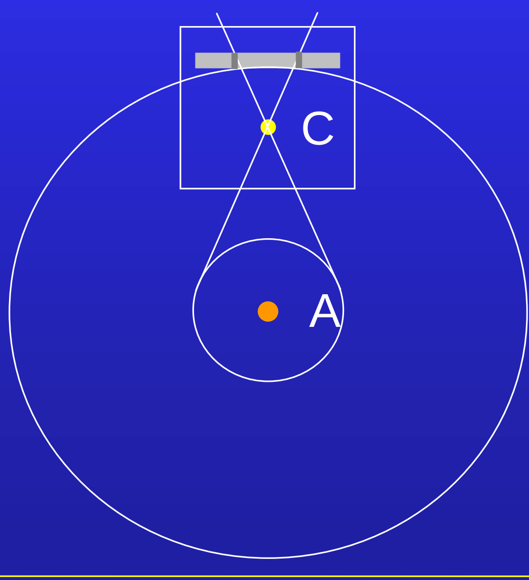
The camera



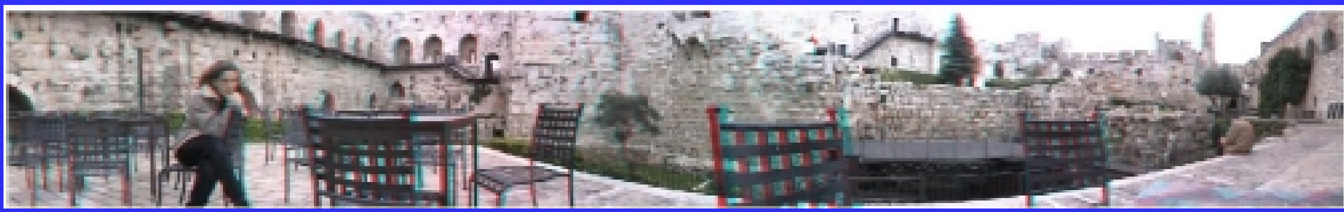
Using video camera

- rotate videocamera in the same way as split camera
- use two vertical image strips inplace of splits





Stereo panorama



S. Peleg (CVPR `99)

S. Peleg (CVPR `99)

Ommnistereo demo © OmniStereo Cooporation, 2000

Part 3:

3-D scene reconstruction from multiple views

- Projective and metric reconstruction
- 2-view epipolar constraint
- camera tracking from multiple views
- stereoscopic depth estimation
- 3-D surface modeling from multiple views

Scene reconstruction with projective cameras

- Calibrated Camera: K known, Pose (R,C) unknown (*metric camera*)

$$P = K[R^T \mid -R^T C]$$

- Uncalibrated camera: K,R,C unknown (*projective camera*)

$$P = [KR^T \mid -KR^T C] = [B \mid a]$$

Reconstruction from multiple projective cameras:

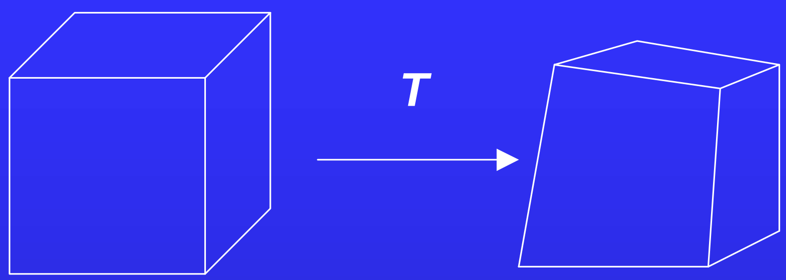
- unknown: Scene points M, projection matrices P
- known: image projections m_i of scene points M_i
- problem: reconstruction is ambiguous in projective space!

$$m_i \simeq PM_i \simeq P(T^{-1}T)M_i \simeq (PT^{-1})(TM_i) \simeq \tilde{P}\tilde{M}_i$$

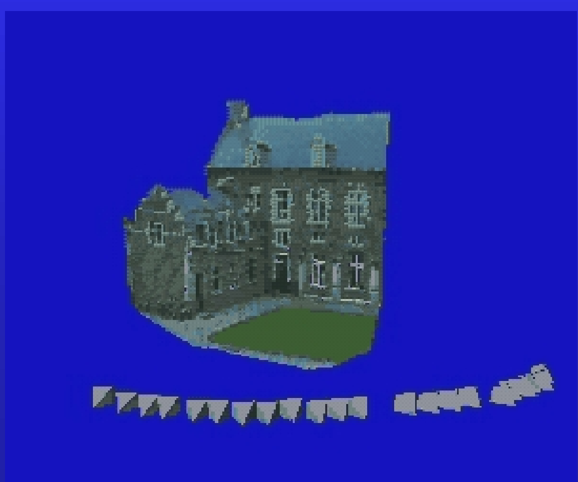
- scene is defined only up to a *projective Transformation T*
- *camera is skewed by inverse Transformation T^{-1}*

Ambiguity in projective reconstruction

Euclidean scene



Projective reconstruction

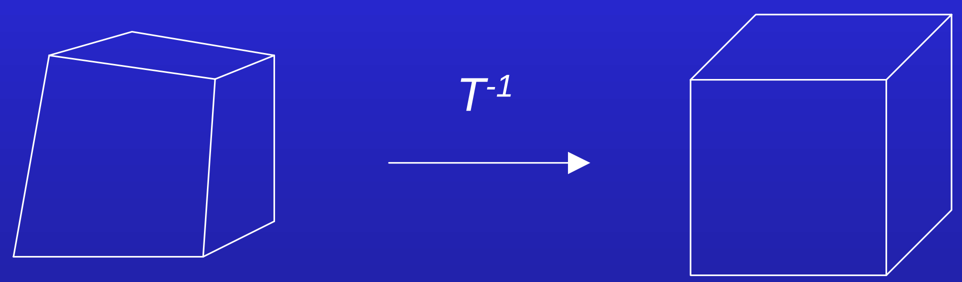


Valid projective reconstructions

Self-Calibration

- Recover metric structure from projective reconstruction
- Use constraints on the calibration matrix *K*
- Utilize invariants in the scene to estimate *K*

Apply self-calibration to recover T^{-1}



Projective reconstruction   Metric reconstruction

Camera Self-Calibration from *H*

- Estimation of *H* between image pairs gives complete projective mapping (8 parameter).
- Problem: How to compute camera projection matrix from *H*
  - since *K* is unknown, we can not compute *R*
  - *H* does not use constraints on the camera (constancy of *K* or some parameters of *K*)
- Solution: self-calibration of camera calibration matrix *K* from image correspondences with *H*
- imposing constraints on *K* may improve calibration

Interpretation of *H* for metric camera:
$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} = K_k R_k^{-1} R_i K_i^{-1}$$

Self-calibration of K from H

- Imposing structure on H can give a complete calibration from an image pair for constant calibration matrix K

homography $H_{ik} = K_k R_k^{-1} R_i K_i^{-1}$

relative rotation: $R_k^{-1} R_i = R_{ik}$, constant camera: $K_i = K_k = K$

$$H_{ik} = K R_{ik} K^{-1} \qquad \Rightarrow R_{ik} = K^{-1} H_{ik} K$$

since $R_{ik}^{-1} = R_{ik}^T \Rightarrow R_{ik} = R_{ik}^{-T} \Rightarrow R_{ik} = K^{-1} H_{ik} K = K^T H_{ik}^{-T} K^{-T}$

$$\boxed{\Rightarrow KK^T = H_{ik} (KK^T) H_{ik}^T}$$

- Solve for elements of (KK^T) from this linear equation, independent of R
- decompose (KK^T) to find K with Choleski factorisation
- 1 additional constraint needed (e.g. s=0) (Hartley, 94)

Self-calibration for varying K

- Solution for varying calibration matrix K *possible*, if
  - at least 1 constraint from K is known (s= 0)
  - a sequence of n image homographies H_{0i} exist

homography $H_{0i} = K_0 R_0^{-1} R_i K_i^{-1} \qquad \Rightarrow R_{ik} = K_0^{-1} H_{0i} K_i = K_0^T H_{0i}^{-T} K_i^{-T}$

$$\boxed{\Rightarrow K_i K_i^T = H_{0i} (K_0 K_0^T) H_{0i}^T}$$

solve by minimizing constraint $\Rightarrow \sum_{i=1}^{n-1} \left\| K_i K_i^T - H_{0i} (K_0 K_0^T) H_{0i}^T \right\|^2 \Rightarrow \min!$

- Solve for varying K (e.g. Zoom) from this equation, independent of R
- 1 additional constraint needed (e.g. s=0)
- different constraints on K_i can be incorporated (Agapito et. al., 01)

Multiple view geometry

Projection onto two views:

$$\boxed{P_0 = K_0 R_0^{-1}[I \quad 0]}$$

$$\rho_0 m_0 = P_0 M = K_0 R_0^{-1}[I \quad 0]M$$

$$\Rightarrow \rho_0 m_0 = K_0 R_0^{-1}[I \quad 0]M_\infty$$

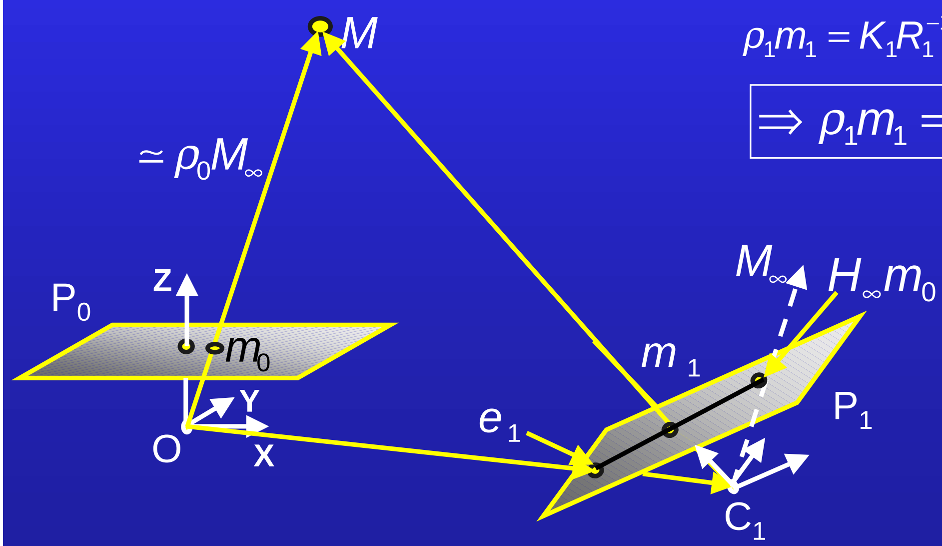
$$\boxed{P_1 = K_1 R_1^{-1}[I \quad -C_1]}$$

$$\rho_1 m_1 = P_1 M = K_1 R_1^{-1}[I \quad -C_1]M$$

$$= K_1 R_1^{-1}[I \quad 0]M_\infty + K_1 R_1^{-1}[I \quad -C_1]O$$

$$\rho_1 m_1 = K_1 R_1^{-1} R_0 K_0^{-1} \rho_0 m_0 - K_1 R_1^{-1} C_1$$

$$\boxed{\Rightarrow \rho_1 m_1 = \underbrace{\rho_0 H_\infty m_0 + e_1}_{\text{Epipolar line}}}$$



$$M = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = M_\infty + O$$

The Fundamental Matrix *F*

- The projective points e_1 and $(H_\infty m_0)$ define a plane in camera 1 (epipolar plane Π_e)
- the epipolar plane intersect the image plane 1 in a line (epipolar line l_e)
- the corresponding point m_1 lies on that line: $m_1^T l_e = 0$
- If the points (e_1),(m_1),$(H_\infty m_0)$ are all collinear, then the collinearity theorem applies:

$$\text{collinearity of } m_1, e_1, H_\infty m_0 \Rightarrow m_1^T(\underbrace{[e_1]_x H_\infty}_{F_{3x3}} m_0) = 0$$

$[\ ]_x =$ cross operator

Fundamental Matrix F

$$\boxed{F = [e_1]_x H_\infty}$$

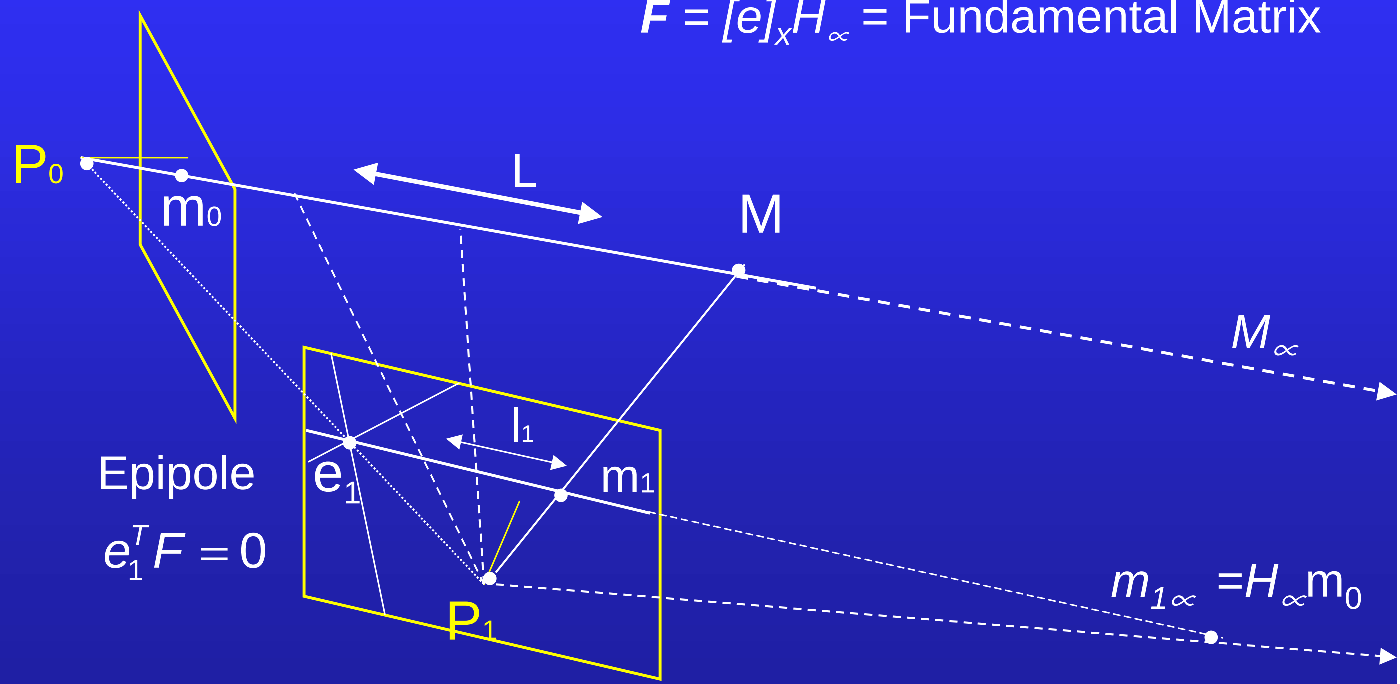
Epipolar constraint

$$\boxed{m_1^T F m_0 = 0}$$

The Fundamental Matrix F

$$m_1^T l_1 = 0 \qquad l_1 = F m_0 \qquad m_1^T F m_0 = 0$$

$\boldsymbol{F} = [e]_x H_\infty$ = Fundamental Matrix



Estimation of *F* from image correspondences

- Given a set of corresponding points, solve linearily for the 9 elements of *F* in projective coordinates
- since the epipolar constraint is homogeneous up to scale, only eight elements are independent
- since the operator $[e]_x$ and hence *F* have rank 2, *F* has only 7 independent parameters (all epipolar lines intersect at e)
- each correspondence gives 1 collinearity constraint

=> solve F with minimum of 7 correspondences

for N>7 correspondences minimize distance point-line:

$$m_{1i}^T F m_{0i} = 0 \qquad \sum_{n=0}^{N} (m_{1,n}^T F m_{0,n})^2 \Rightarrow \min!$$

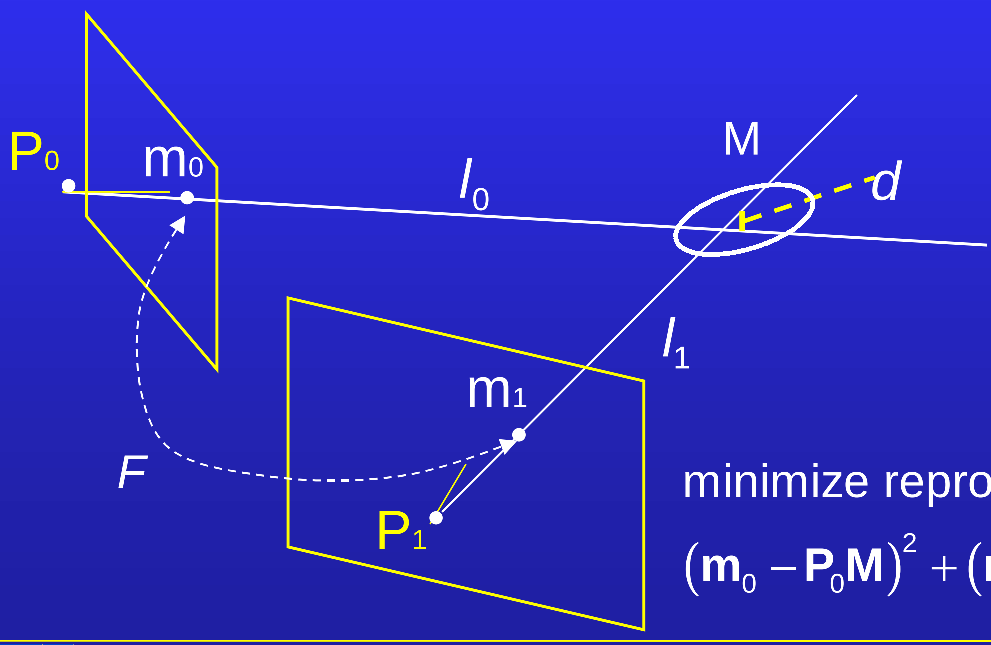
Estimation of *P* from *F*

- From F we can obtain a camera projection matrix pair:
  - Set P_0 to identity
  - compute P_1 from F (up to projective transformation)
  - reduce projective skew by initial estimate of *K* to obtain a quasi-euclidean estimate (Pollefeys et.al., '98)
  - compute self-calibration to obtain metric *K* similar to self-calibration from *H* (Pollefeys et.al, '99, Fusiello '00)

Fundamental matrix	$\leftrightarrow$	Projective camera
$m_1^T F m_0 = 0$		$P_0 = [I \mid 0]$
$e_1^T F = 0$		$P_1 = [[e_1]_x F + e_1 a^T \mid e]$

3D Feature Reconstruction

- corresponding point pair (m_0, m_1) is projected from 3D feature point M
- M is reconstructed from by (m_0, m_1) triangulation
- M has minimum distance of intersection



$\|d\|^2 \Rightarrow \text{min!}$

constraints:

$l_0^T d = 0$

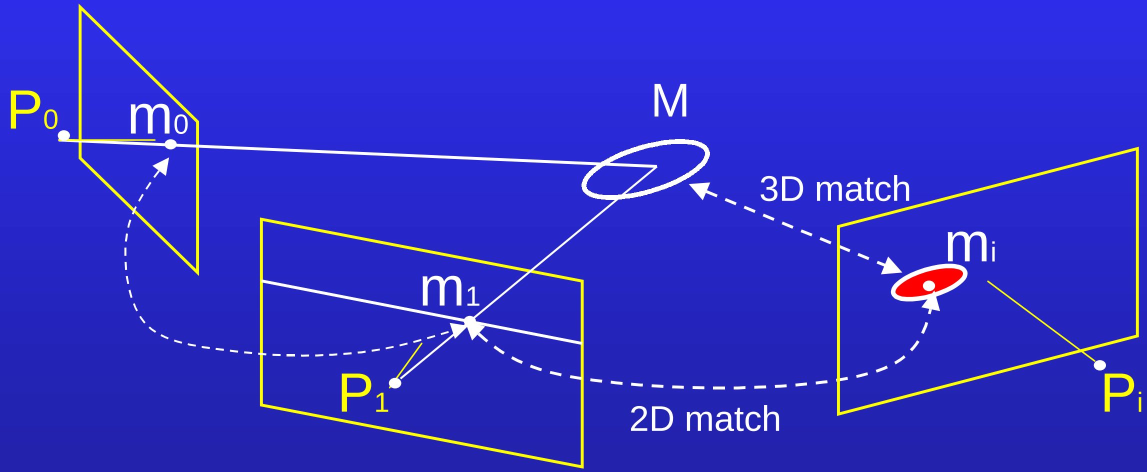
$l_1^T d = 0$

minimize reprojection error:

$$(\mathbf{m}_0 - \mathbf{P}_0\mathbf{M})^2 + (\mathbf{m}_1 - \mathbf{P}_1\mathbf{M})^2 \Rightarrow \text{min.}$$

Multi View Tracking

- 2D match: Image correspondence (m_1, m_i)
- 3D match: Correspondence transfer (m_i, M) via P_1
- 3D Pose estimation of P_i with m_i - P_i M => min.

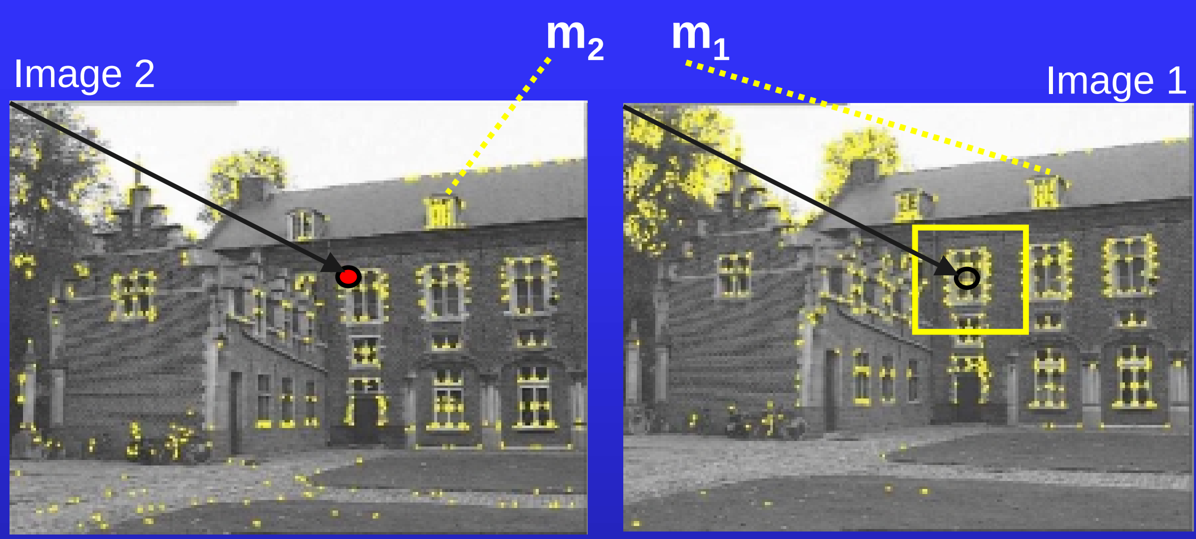


Minimize lobal reprojection error: $\sum_{i=0}^{N}\sum_{k=0}^{K}\left\|m_{k,i}-P_i M_k\right\|^2 \Rightarrow$ min!

Structure from motion: an example

Image Sequence

Extraction of image features



- features $\mathbf{m}_{1,2}$ (Harris Cornerdetector)
- Select candidates based on cross correlation
- Test candidates

Robust selection of correspondences

RANSAC (RANdom Sampling Consensus)

Hypothesis test:

- For N Trials

  DO:

  – random selection of possible correspondences (minimum set)

  – compute **F**

  – test all correspondences [*inliers/outliers*]

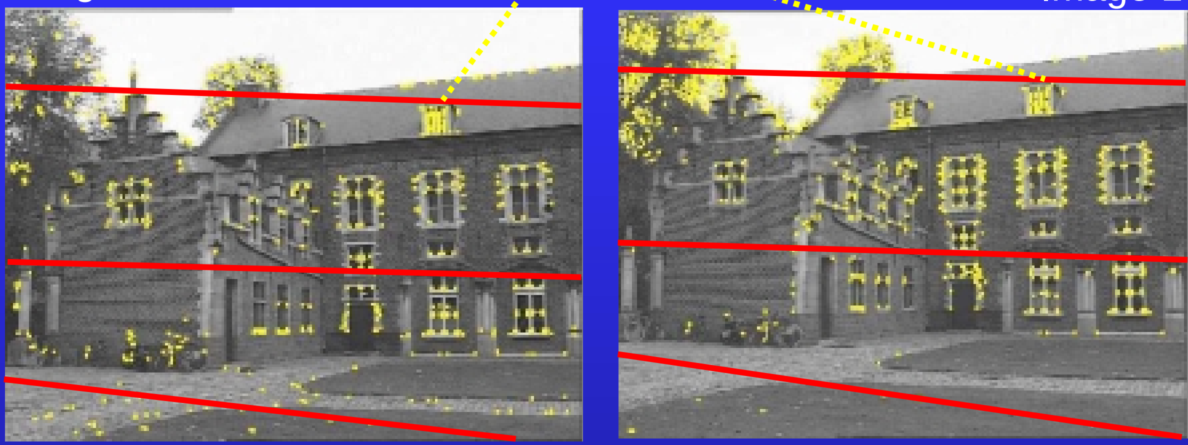
  UNTIL (Probability[*#inliers, #Trials*] > 95%)

- Refine **F** with ML-estimate

Estimation of Fundamental Matrix

$$m_2^T F\, m_1 = 0$$

Image 2 Image 1



Robust correspondence selection $\mathbf{m_1}$ **<-> $\mathbf{m_2}$**

Camera and feature tracking

reconstruction of 3D features and cameras

3D surface modeling

- camera calibration from feature tracking
- dense depth estimation from stereo correspondence
- depth fusion and generation of textured 3D surface model

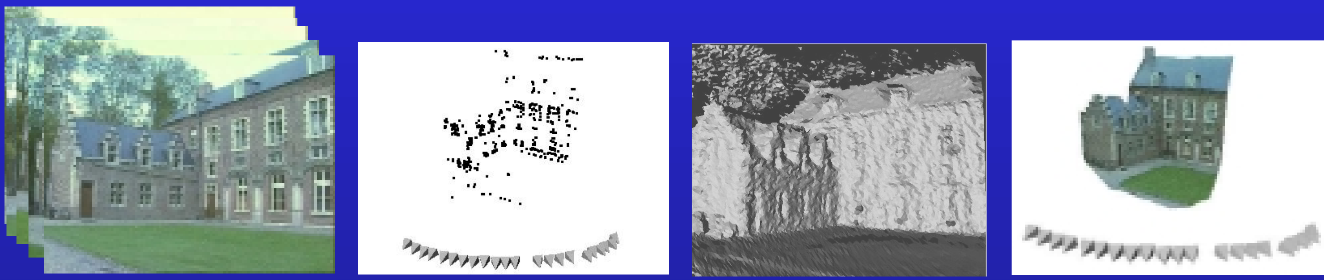
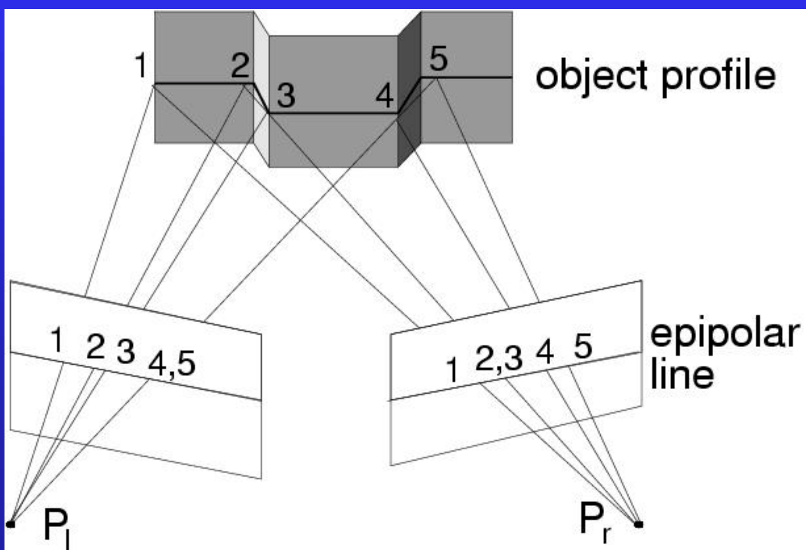


Image sequence camera calibration scene geometry 3D surface model

Dense depth estimation

- use of constraints along epipolar line (ordering, uniqueness)
- matching with normalized cross correlation
- fast matching in rectified image pair



Epipolar line search

Image rectification with planar mappings

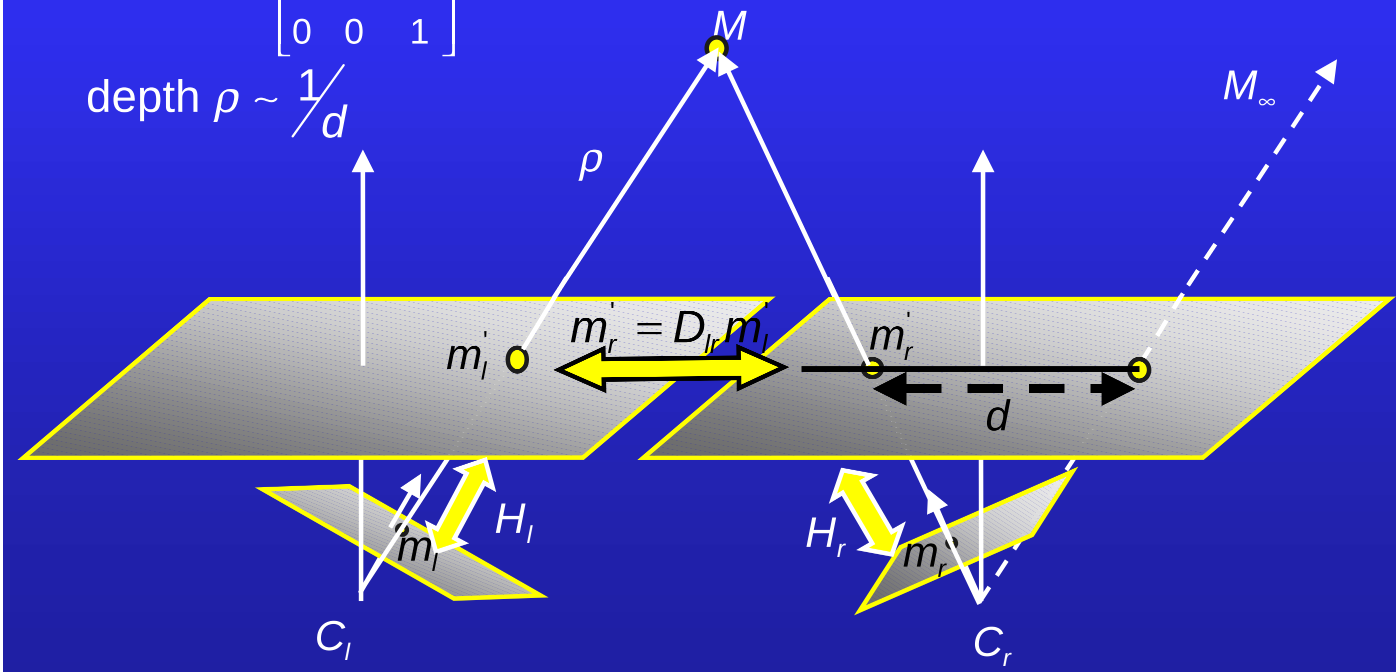
estimate disparity D

$$D_{lr}(m_l^{'}, m_r^{'}) = \begin{bmatrix} 1 & 0 & -d \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

depth $\rho \sim 1/d$

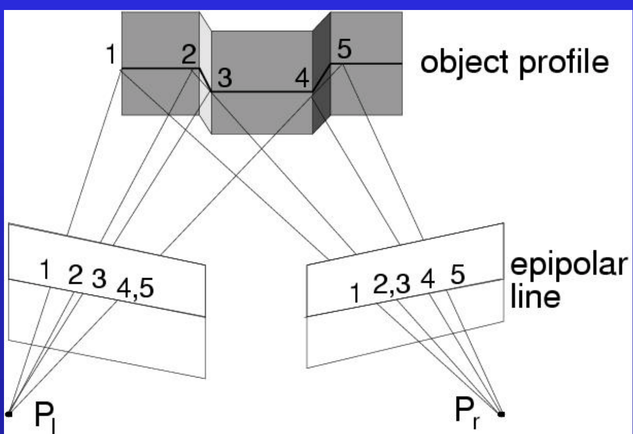
map between images

$$m_r = H_r^{-1} m_r^{'} = H_r^{-1} D_{lr} m_l^{'} = H_r^{-1} D_{lr} H_l m_l$$

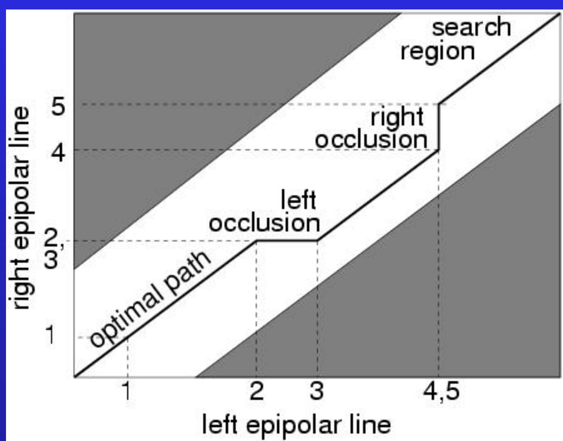


Dense search for *D*

- use of constraints along epipolar line (ordering, uniqueness)
- matching with normalized cross correlation
- constrained epipolar search with dynamic programming



Epipolar lines

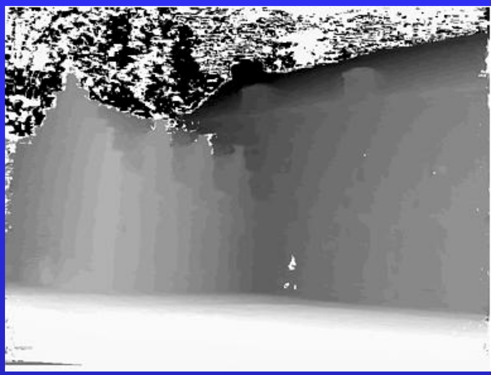


Search path for constrained matching

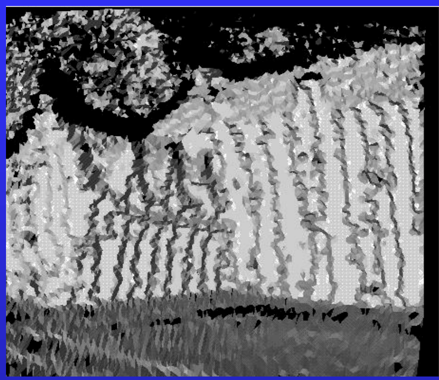
Depth map

Originals

Depth map ρ(m)



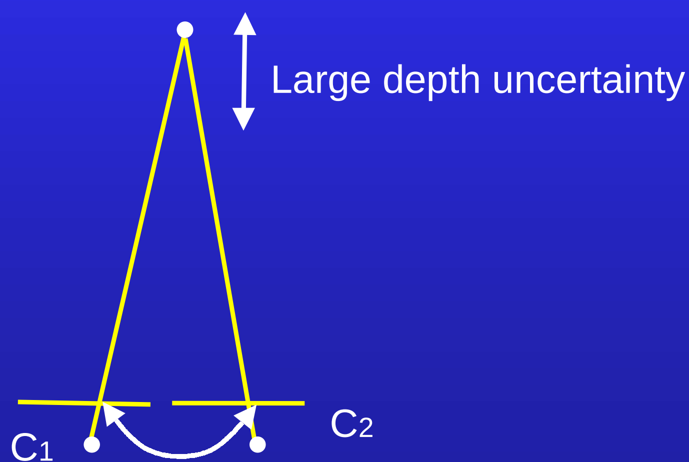
shaded surface model



Problem: Triangulation angle limits depth resolution

The Baseline problem

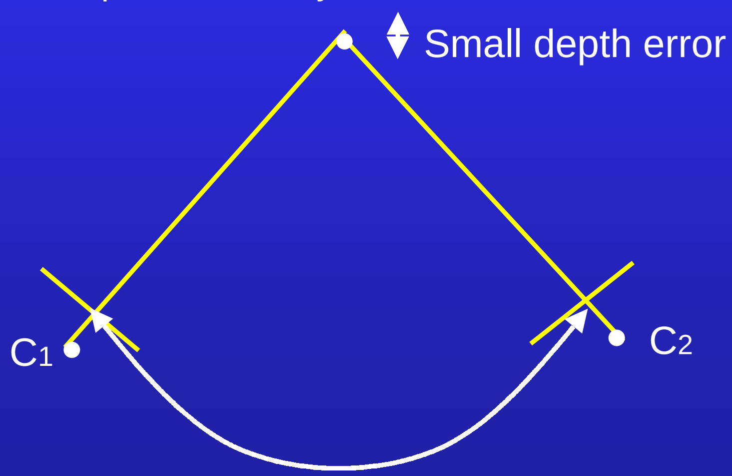
- small baseline stereo:
  - + Correspondence is simple (images are similar)
  - + few occluded regions
  - - large depth uncertainty



Small camera baseline (small triangulation angle)

The Baseline problem

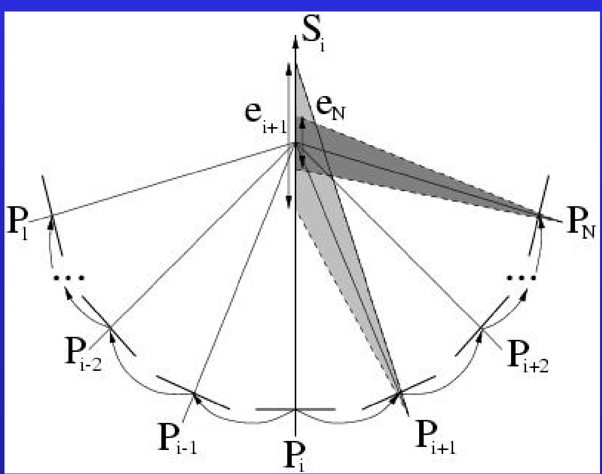
- wide baseline stereo:
  - Correspondence is difficult (images are not similar)
  - many occluded regions
  + small depth uncertainty



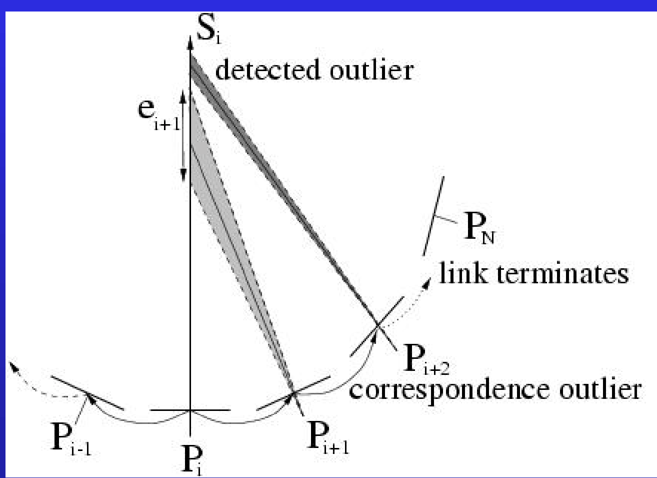
Large camera baseline (large triangulation angle)

Multi-Viewpoint Depth Fusion

- Concatenate correspondences over adjacent image pairs
- depth triangulation along line of sight
- depth fusion of all triangulated correspondences, remove outliers



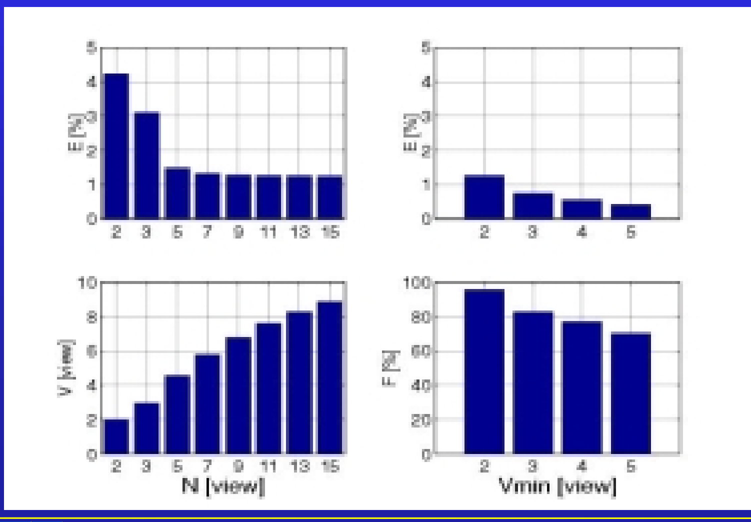
Depth fusion



Detection of outliers

Depth uncertainty after sequence integration

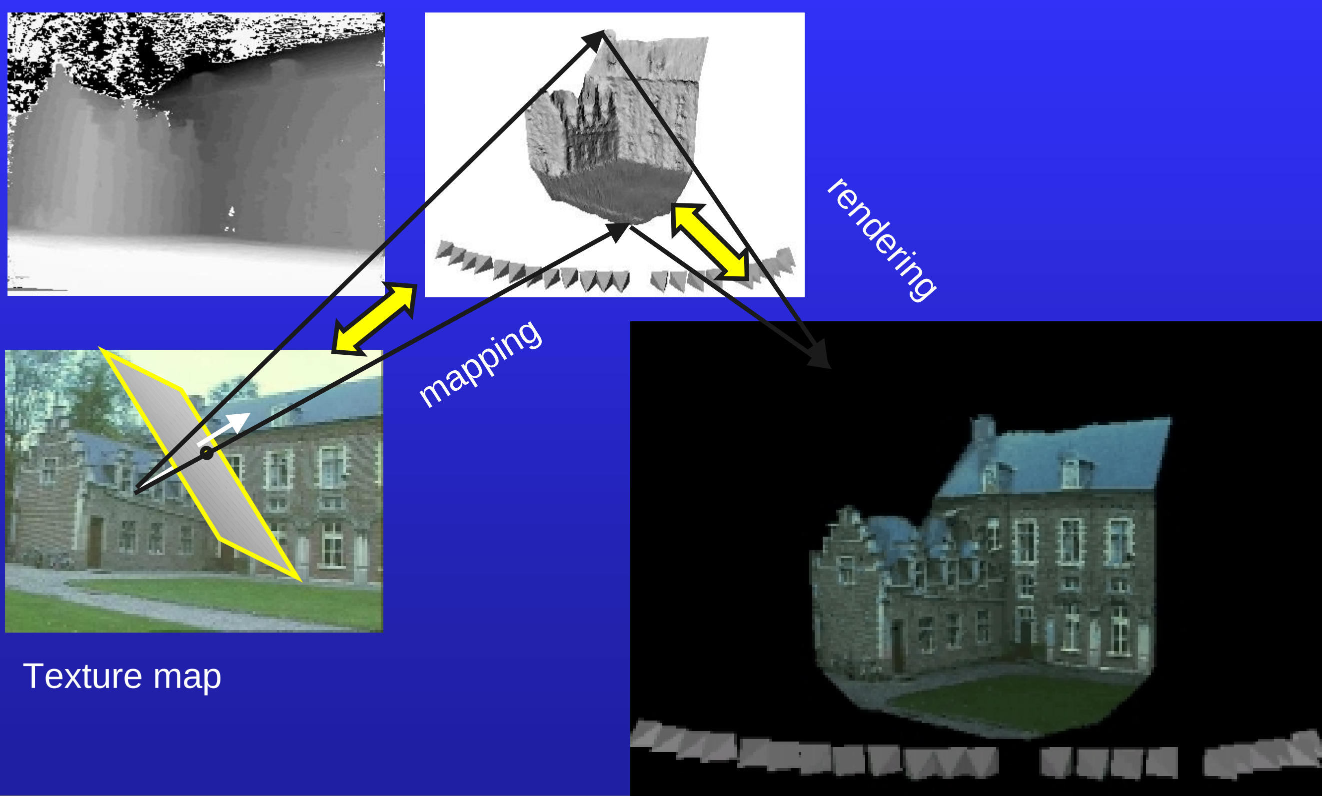
- Improved density, fewer occlusions
- Depth error decreases
- Improved surface geometry



3D surface Modeling

Depth map

surface mesh



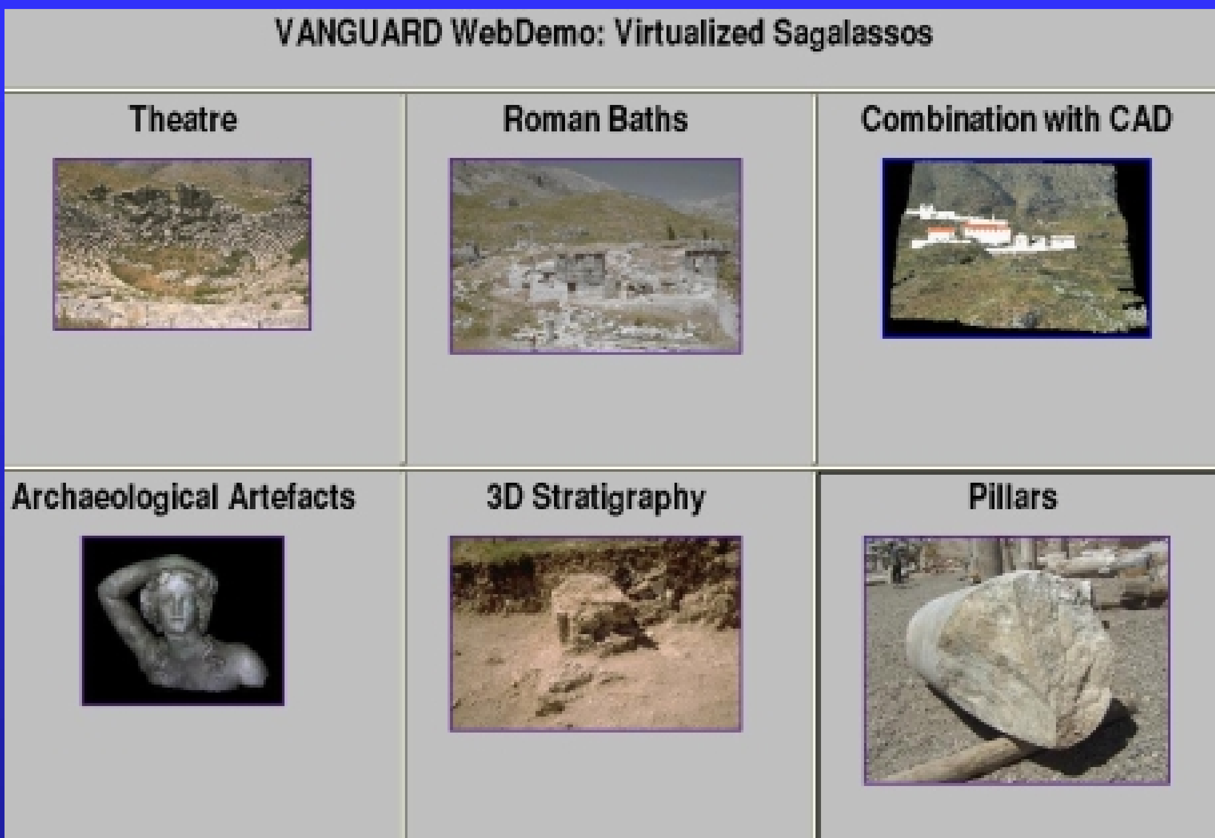
Texture map

Jain Tempel (Ranakpur, Indien)

Images

3D-model

Sagalassos: Virtual Museum



Open Questions from geometric modeling

- How much geometry is really needed for visualisation ?
- Trade-off: modeling vs. texture mapping (reflectance, microstructure)
- how can we model surface reflections ?
- How can we efficiently store and render the scenes ?

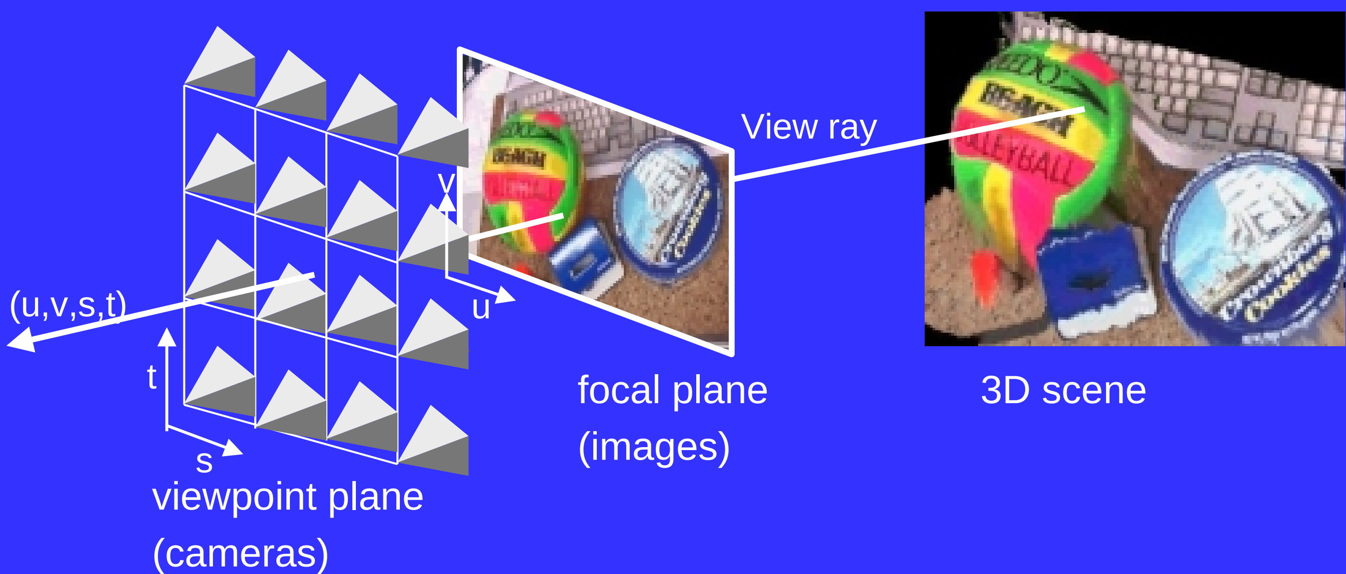
Part 4: Plenoptic Modeling

- The lightfield
- depth-dependent view interpolation
- The uncalibrated Lumigraph
- Visual-geometric modeling: view-dependent interpolation and texture mapping

Modeling of scenes with surface reflections

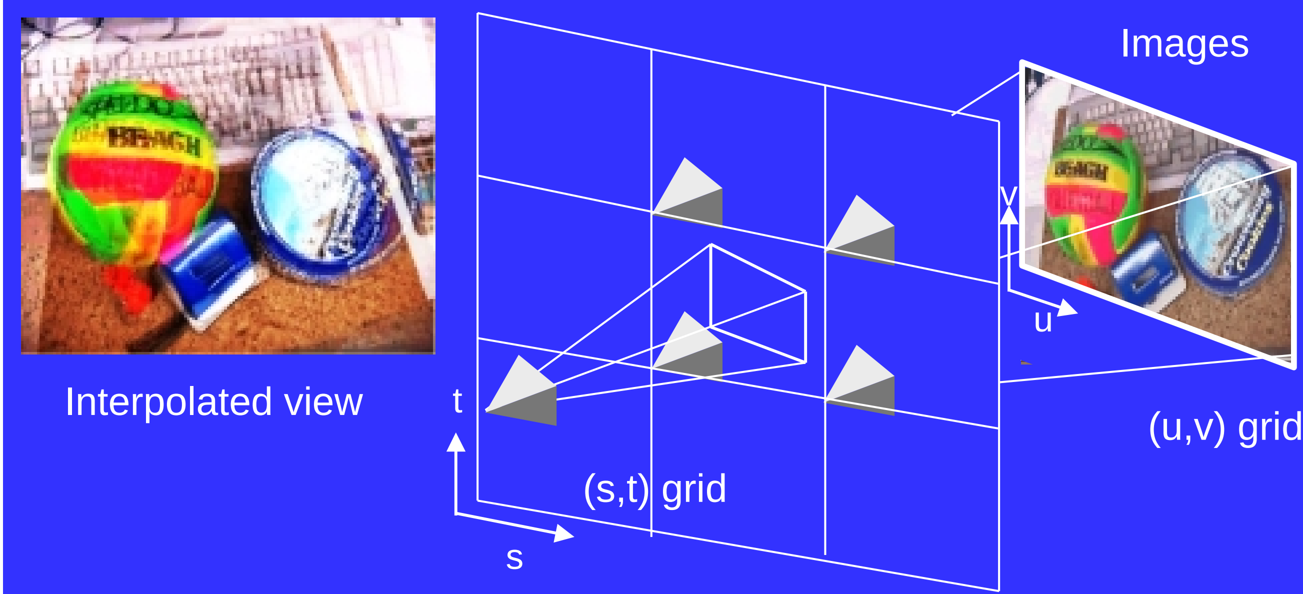
- Problem when modeling scenes with reflections
  - view dependent reflections can not be handled by single texture map
  - may not be able to recover geometry
- Approach: Lightfield rendering

Image-based Acquisition: The Lightfield



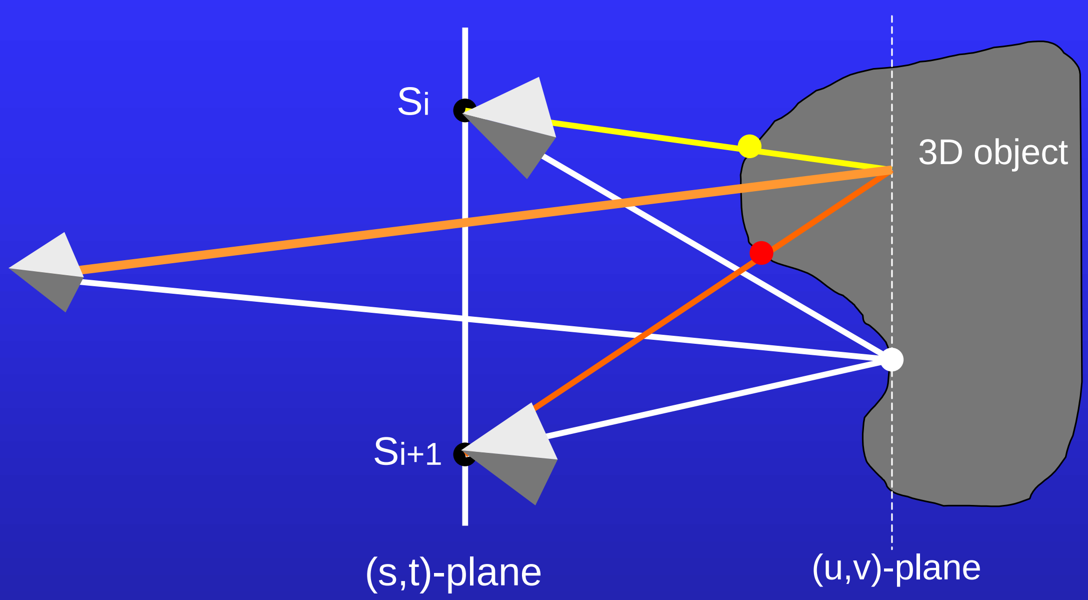
4-D Lightfield Data structure: Grid of cameras (s,t) and grid of images (u,v) store all possibe surface reflections of the scene

Image-based Rendering from Lightfield



Interpolation errors (ghosting) due to unmodelled geometry!

Depth-dependent Rendering Error



Combining Lightfield and Structure from Motion

Lightfield (LF):

– needs camera calibration and depth estimation

+ efficient rendering of view-dependent scenes

Structure from Motion (SFM):

+ delivers camera calibration and depth estimation

– can not handle view-dependent scenes

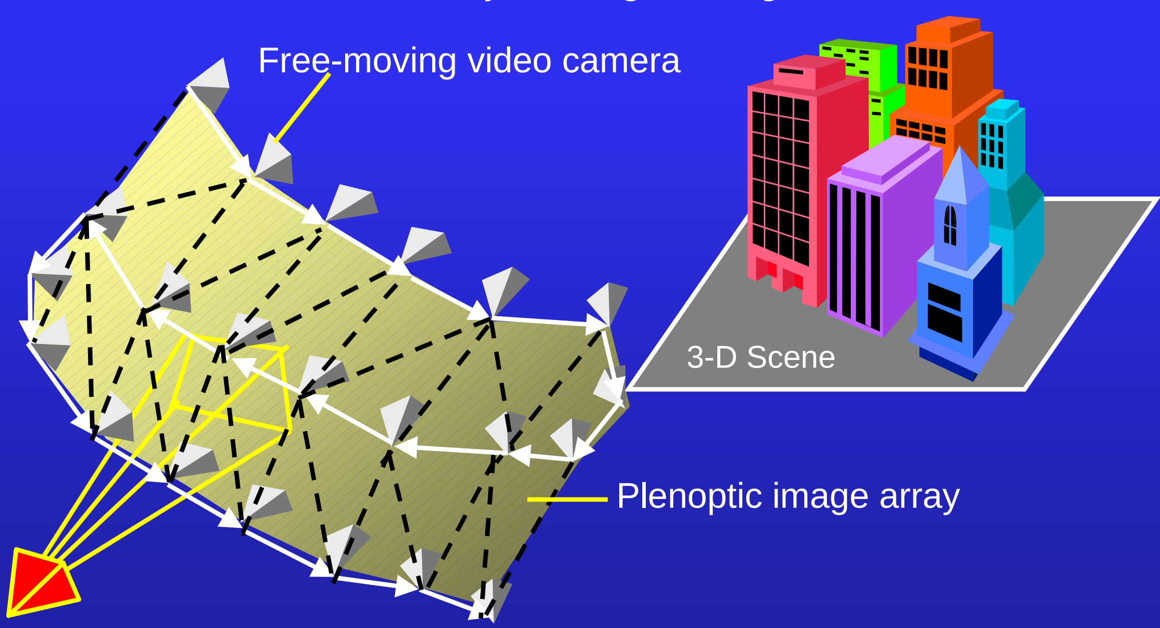
Uncalibrated Lumigraph:

+ tracking and scene structure with SFM

+ view-dependent rendering with generalized LF

Uncalibrated plenoptic modeling

Plenoptic modeling: Build an array of images by concatenating views from an uncalibrated, freely moving moving video camera

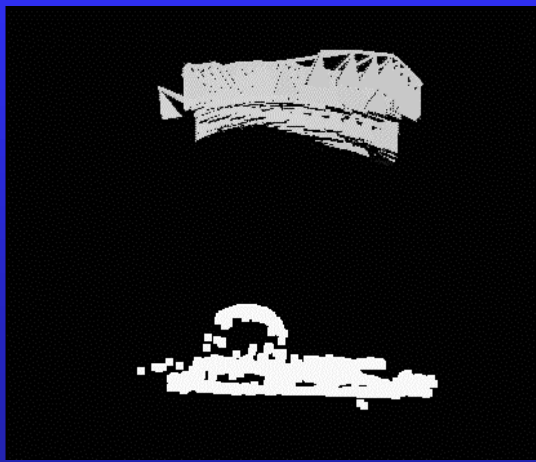


Plenoptic rendering: Render novel views of the observed scene by image interpolation from the plenoptic image array

Plenoptic Modeling from uncalibrated Sequence

Office sequence: sweeping a camera freely over cluttered desk environment

Input sequence

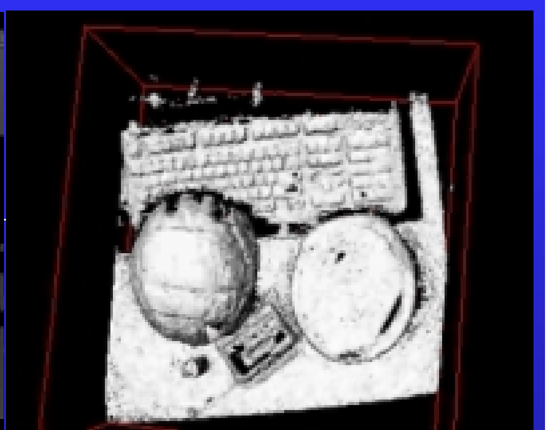


Viewpoint surface mesh calibration

Depth Maps and geometry

Calibrated image pairs

local depth maps



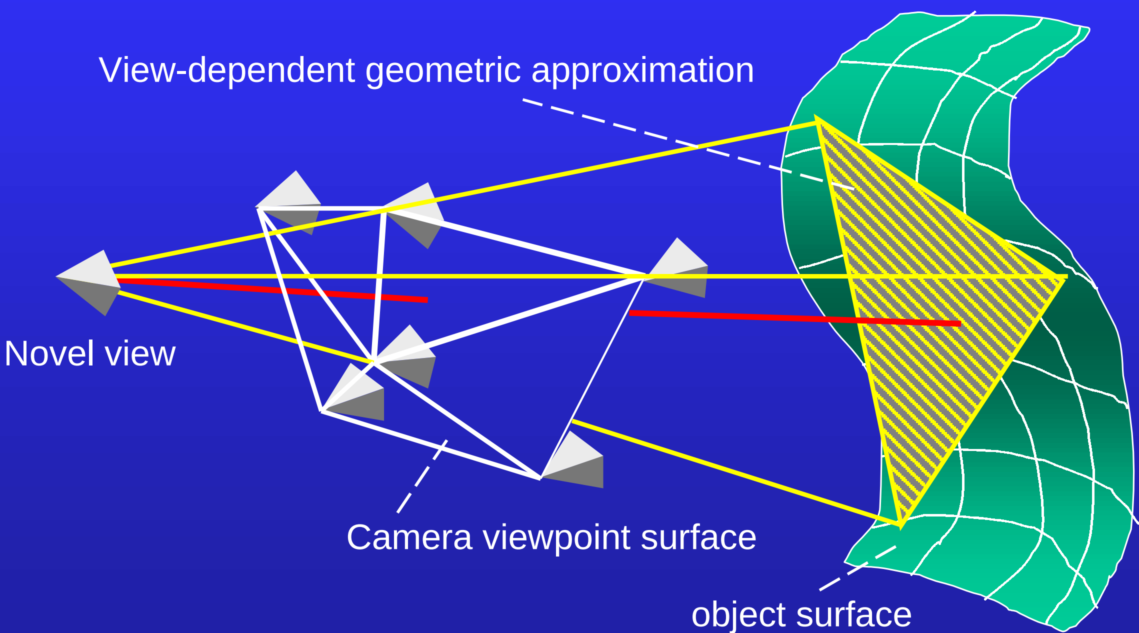
fused geometry

3D surface model

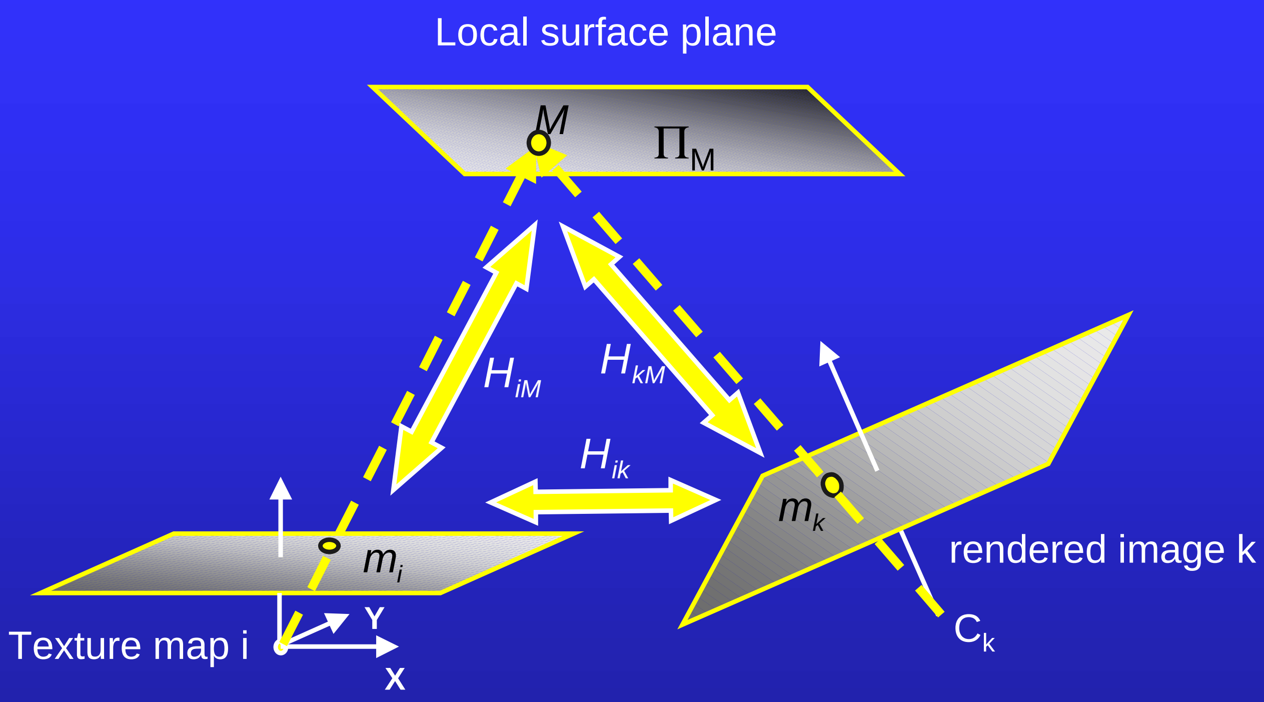
Calibrated image pairs   local depth maps   3D surface model

Viewpoint-Adaptive Rendering

- Render images directly from calibrated camera viewpoints
- Interpolate by approximating local scene geometry



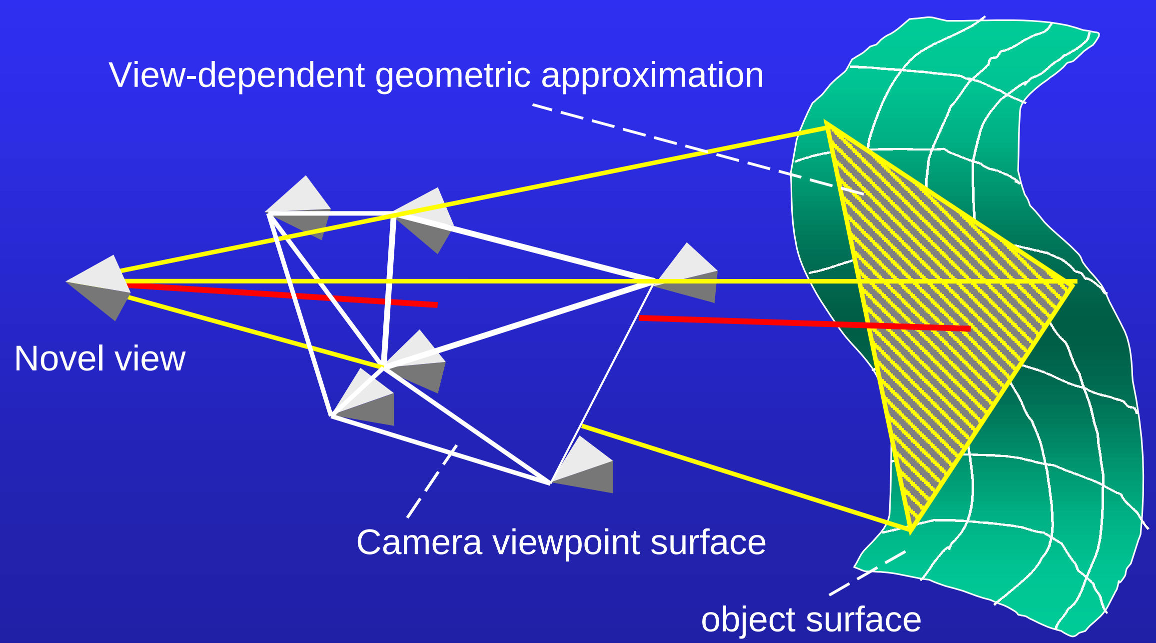
Projective texture mapping on planar scene



Transfer between images i,k over Π_M: $H_{ik} = H_{iM} H_{kM}^{-1}$

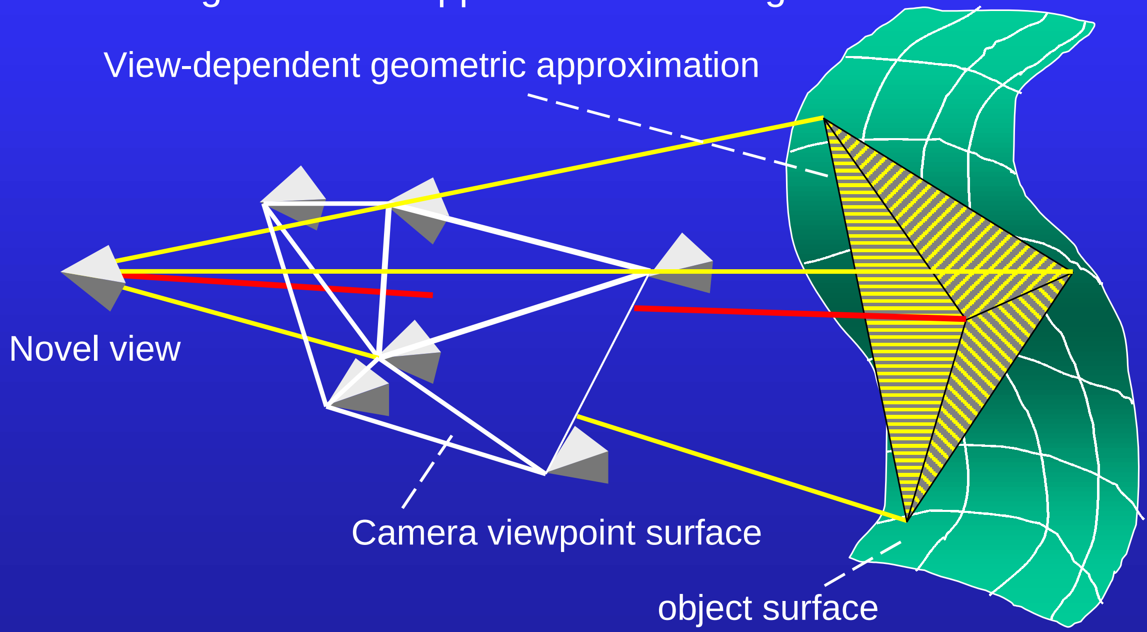
Viewpoint-Adaptive Rendering

- Render images directly from calibrated camera viewpoints
- Interpolate by approximating local scene geometry



Viewpoint-Adaptive Rendering

- Render images directly from calibrated camera viewpoints
- Interpolate by approximating local scene geometry
- Scalable geometrical approximation through subdivision



Rendering: 3 projective mappings and α-blending per triangle (OpenGL Hardware)

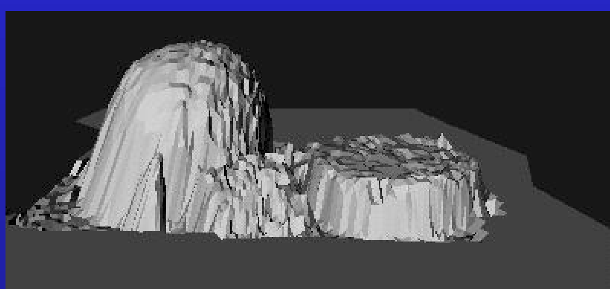
Scalable Geometry for View Interpolation

- Adaptation of geometric detail for Lightfield Interpolation
- Geometry is computed online for every rendering viewpoint

1 Surface plane per viewpoint triangle

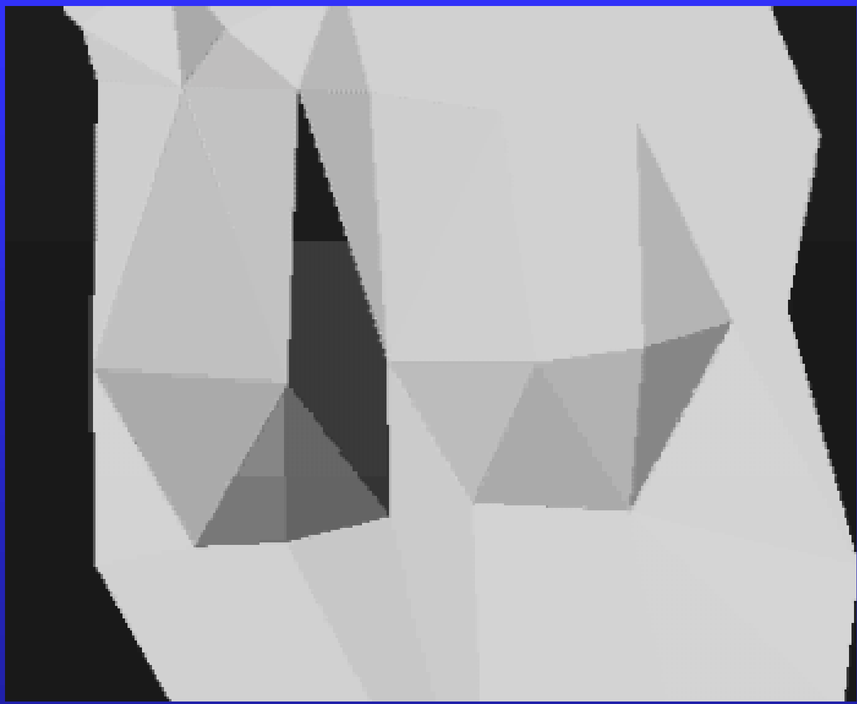
2 Subdivisions

1 Subdivision per viewpoint triangle



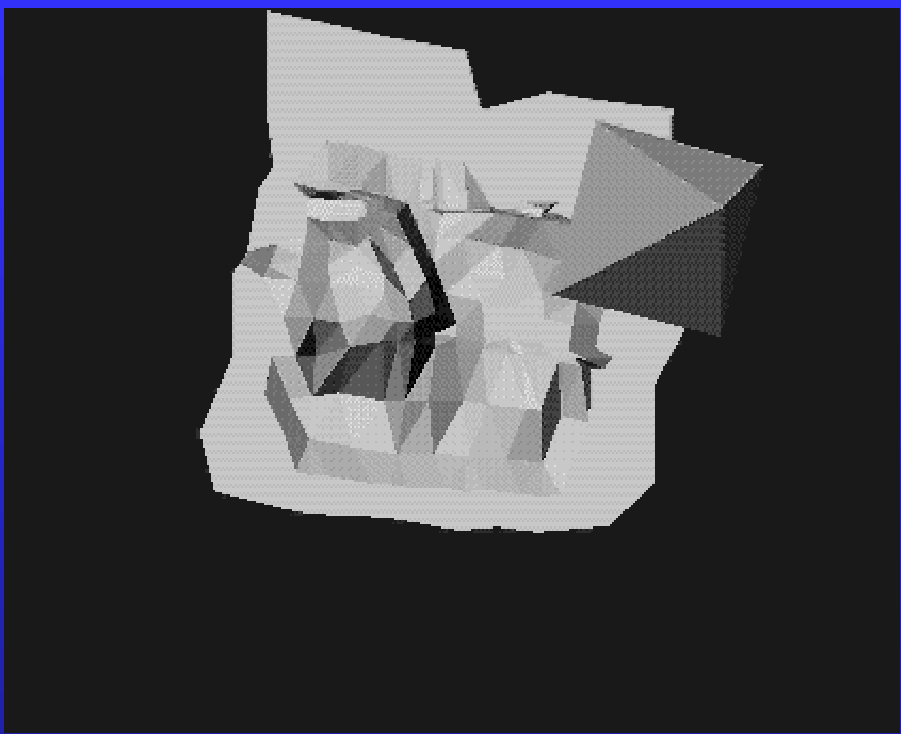
4 Subdivisions

Scalable Geometry



Adaptation of Geometric detail to interpolation error

Viewpoint-Adaptive Geometry



Adaptation of Geometry to viewpoint (2 subdivisions)

Uncalibrated Lumigraph: Rendering Results

Rendering from lightfield calibration with planar approximation

Rendering with locally adapted geometry (viewpoint mesh 2 x subdivided)

Conclusions

A complete framework for automatically reconstructing and rendering of 3D scenes from uncalibrated image sequences was presented. It combines *structure from motion* with *Image-based rendering*.

Properties of the approach:

- handle uncalibrated sequences from freely moving hand-held cameras
- calibrate lightfield sequences with viewpoint mesh connectivity
- exploit scalable geometric complexity
- rendering of surface reflections