Computer Vision for Augmented Reality

Jan-Michael Frahm (UNC, Chapel Hill)
Raphael Grasset (HITLabNZL)

Introduction

Computer vision enables the computer to visually perceive our world.

Introduction

Computer Vision

Computer vision enables the computer to visually perceive our world.

To achieve this goal, one needs to extract:

- the camera geometry (calibration)
- scene structure (surface geometry)

This tutorial will introduce:

- the basic mathematical tools (projective geometry)
- models for cameras, robust methods for 2D
- 3D camera tracking (marker based and markerless)
- methods to achieve these tasks in real-time

Schedule

- Introduction
- Multi-view Geometry
- Markerless Tracking
- Robust pose estimation
- Coffee Break
- AR-Toolkit
- Scene Modeling
- Applications

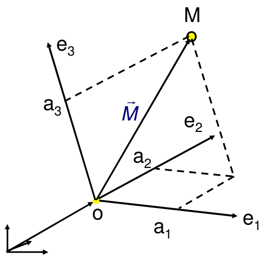
Schedule

- Introduction
- Multi-view Geometry
- Markerless Tracking
- Robust pose estimation
- Coffee Break
- AR-Toolkit
- Scene Modeling
- Applications

Schedule

- Introduction
- Multi-view Geometry
  - Coordinate systems and Geometric Entities
  - Definition and estimation of entities P, H, F, E
  - Structure Computation
- Markerless Tracking
- Robust pose estimation
- Coffee Break
- AR-Toolkit
- Scene Modeling
- Applications

Affine coordinates



e_i: affine basis vectors

o: coordinate origin

Vector relative to o:

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

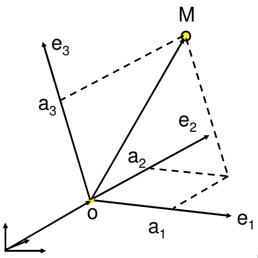
Point in affine coordinates:

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

Vector: relative to some origin

Point: absolute coordinates

Homogeneous coordinates



Unified notation:

include origin in affine basis

Homogeneous Coordinates of M

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

Affine basis matrix

Properties of 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

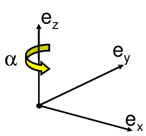
$$M' = T_{affine}M = \begin{bmatrix} A_{3x3} & t_3 \\ 0\ \ 0\ \ 0 & 1 \end{bmatrix} M \qquad T_{affine} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Parallelism is preserved
- ratios of length, area, and volume are preserved
- Transformations can be concatenated:
  if $M_1 = T_1M$ and $M_2 = T_2M_1 \Rightarrow M_2 = T_2T_1M = T_{21}M$

Special transformation: Rotation

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

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

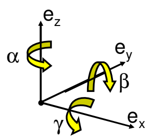


$$R_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation with angle α around e_z

Special transformation: Rotation

- Rotation around the coordinate axes can be concatenated:



$$R = R_zR_yR_x$$

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

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

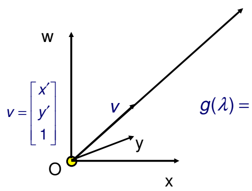
Inverse of rotation matrix is transpose:

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

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

Projective geometry in 2D

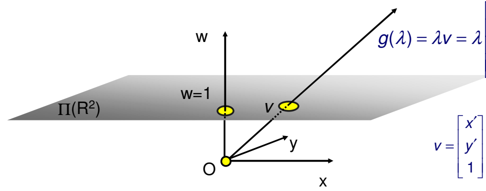
- Projective space is space of rays emerging from *O*
  - view point *O* forms projection center for all rays
  - rays *v* emerge from viewpoint into scene
  - *ray g* is called projective point, defined as scaled *v*: $g = \lambda v$



$$v = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \qquad g(\lambda) = \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \lambda v, \lambda \in \Re \neq 0$$

Projective and homogeneous points

- Given: Plane Π in R^2 embedded in R^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 $v = g/\lambda = g/w$

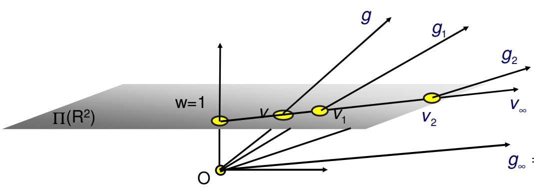


$$g(\lambda) = \lambda v = \lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

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

Finite and infinite points

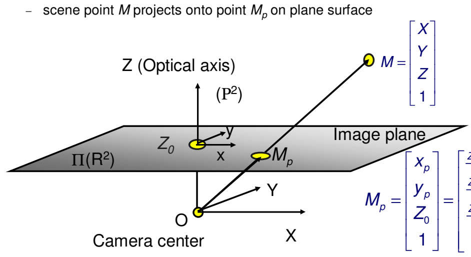
- All rays *g* that are not parallel to Π intersect at an affine point *v* on Π.
- The ray *g*(w=0) does not intersect Π. Hence v_∞ is not an affine point but a direction. Directions have the coordinates $(x,y,0)^T$
- Projective space combines affine space with infinite points (directions).



$$g_\infty = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

Perspective projection

- Perspective projection models pinhole camera:
  - scene geometry is affine 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 Π(R^2) aligned with plane (X,Y) at Z= Z_0
  - scene point *M* projects onto point M_p on plane surface

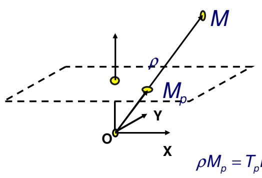


$$M = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

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

Projective Transformation

- Projective Transformation maps *M* onto M_p



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

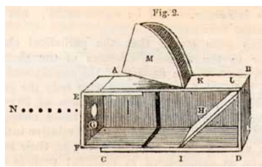
$$\rho = \frac{Z}{Z_0} = \text{projective scale factor}$$

- Projective Transformation linearizes projection

Pinhole Camera (Camera obscura)

Camera obscura

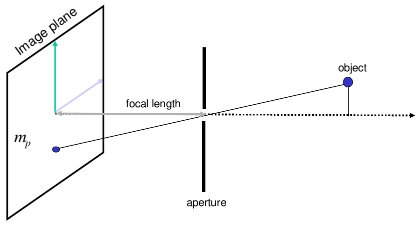
(France, 1830)



Interior of camera obscura

(Sunday Magazine, 1838)

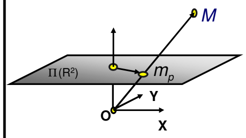
Camera model



from "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Perspective Projection

Dimension reduction from R³ into R² by projection onto Π(R²)

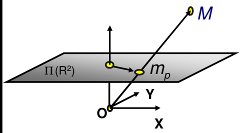


$$\begin{bmatrix} x_p \\ y_p \\ Z_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ Z_0 \\ 1 \end{bmatrix}$$

from "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Perspective Projection

Dimension reduction from R³ into R² by projection onto Π(R²)



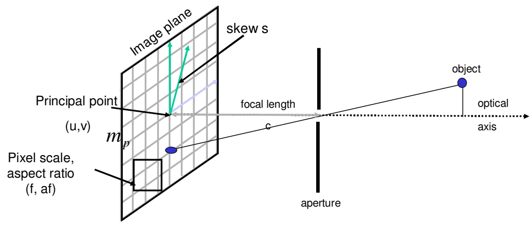
$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ Z_0 \\ 1 \end{bmatrix}$$

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

from "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

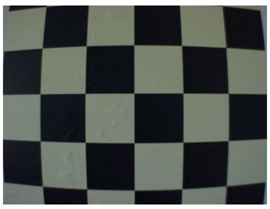
Camera model

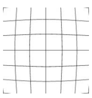
$$m = K \begin{bmatrix} m_p \\ 1 \end{bmatrix} = \begin{bmatrix} f & s & u \\ 0 & af & v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_p \\ 1 \end{bmatrix}$$



from "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Radial distortion





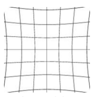
barrel distortion

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(\tilde{r}) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$$

$$L(\tilde{r}) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \kappa_4 r^4 + \dots$$

with $\tilde{r}, r$ radius in distorted image resp. undistorted image

$\kappa_1, \kappa_2, \dots$ radial distortion parameters

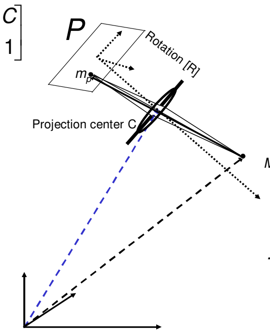


pincushion distortion

from "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Projection in general pose

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



Projection: $\rho m_p = PM$

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

from "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Projection matrix P

- Camera projection matrix P combines:
  - inverse affine transformation T_{cam}^{-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: $T_{scene} = \begin{bmatrix} R^T & -R^T C \\ 0^T & 1 \end{bmatrix}$

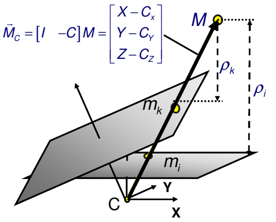
projection: $P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = [I \quad 0]$ sensor calibration: $K = \begin{bmatrix} f & s & u \\ 0 & af & v \\ 0 & 0 & 1 \end{bmatrix}$

$$\Rightarrow \rho m = PM, \quad P = KP_0 T_{scene} = K\left[R^T \quad -R^T C\right]$$

Rotating Camera

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

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



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

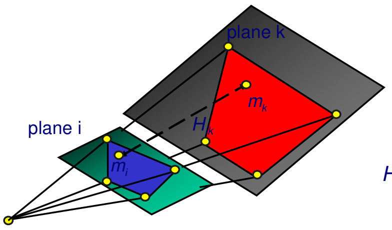
$$\rho_k m_k = K_k R_k^{-1} R_i K_i^{-1} \rho_i m_i = \rho_i H_{ik} m_i$$

- H_{ik} is a planar projective 2D-transformation (3x3) that maps points m_i on plane i to points m_k on plane k

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

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



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

$$H_{ik} \cong \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Estimation of H from images

- H_{ik} can be estimated linearly 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 d is minimized

Projective mapping (linear):

$$m_k = \rho_k \begin{bmatrix} x_k \\ y_k \\ 1 \end{bmatrix} = Hm_i = \begin{bmatrix} h_1 x_i + h_2 y_i + h_3 \\ h_4 x_i + h_5 y_i + h_6 \\ h_7 x_i + h_8 y_i + h_9 \end{bmatrix}$$

Image coordinate mapping (nonlinear):

$$\rho x_k = \frac{h_1 x_i + h_2 y_i + h_3}{h_7 x_i + h_8 y_i + h_9}$$
$$\rho y_k = \frac{h_4 x_i + h_5 y_i + h_6}{h_7 x_i + h_8 y_i + h_9}$$

Error functional d:

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

$$H_{ik} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Estimation of H with DLT

$$m_k = \mathbf{H} \cdot m_i \Rightarrow \begin{bmatrix} x_k \\ y_k \\ w_k \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1^T \cdot m_i \\ \mathbf{h}_2^T \cdot m_i \\ \mathbf{h}_3^T \cdot m_i \end{bmatrix}, \; with \; \mathbf{H} = \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \mathbf{h}_3^T \end{bmatrix}$$

exploit collinearity: $m_{k,n} \times m_{k,n} = m_{k,n} \times (\mathrm{H} m_{i,n}) = \vec{0}$

$$m_{k,n} \times \mathbf{H} \cdot m_{i,n} = \begin{pmatrix} y_{k,n}\mathbf{h}_3^T \cdot m_{i,n} - w_{k,n}\mathbf{h}_2^T \cdot m_{i,n} \\ w_{k,n}\mathbf{h}_1^T \cdot m_{i,n} - x_{k,n}\mathbf{h}_3^T \cdot m_{i,n} \\ x_{k,n}\mathbf{h}_2^T \cdot m_{i,n} - y_{k,n}\mathbf{h}_1^T \cdot m_{i,n} \end{pmatrix} = \vec{0}$$

2 linear independent Equations per correspondence pair ($m_{i,n}$, $m_{k,n}$) gives a matrix **A** with (2n x 9) entries and solution vector **h** with 9 elements of Homography H. Solution **h** is the right Nullspace of **A**.

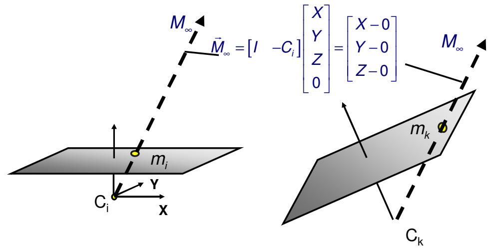
$$\mathbf{A} \cdot \mathbf{h} = \vec{0} \quad \Rightarrow \begin{bmatrix} \mathbf{0}^T & -w_{k,n} \cdot m_{i,n}^T & y_{k,n} \cdot m_{i,n}^T \\ w_{k,n} \cdot m_{i,n}^T & \mathbf{0}^T & -x_{k,n} \cdot m_{i,n}^T \\ \vdots & \ddots & \vdots \end{bmatrix}_{(2nx9)} \bullet \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix}_{(9)} = \vec{0}_{(2n)}$$

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Infinte Homography

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

$$\vec{M}_\infty = [I \quad -C_i] \begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix} = \begin{bmatrix} X - 0 \\ Y - 0 \\ Z - 0 \end{bmatrix}$$

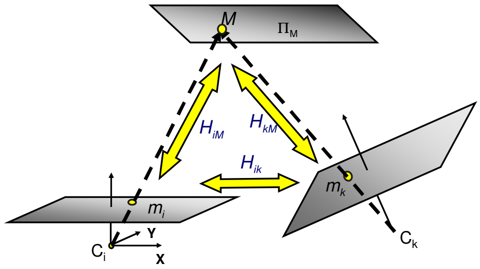


$$\vec{M}_\infty = R_i K_i^{-1} \rho_i m_i = R_k K_k^{-1} \rho_k m_k \qquad \Rightarrow \rho_k m_k = K_k R_k^{-1} R_i K_i^{-1} \rho_i m_i = \rho_i H_{ik} m_i$$

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Image mapping of planar scene Π_M

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

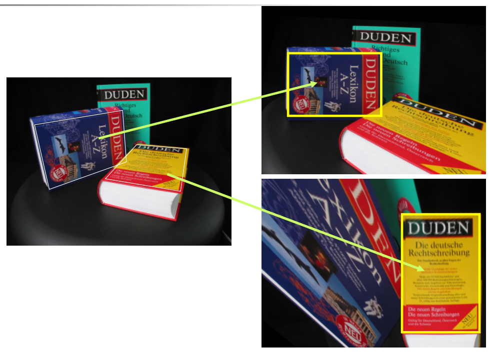


Transfer between images i,k over Π_M: $H_{ik} = H_{iM} H_{kM}^{-1}$

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.
- Homographies are used for projective texture mapping!

Homography mapping example



From: O.Schreer: Stereoanalyse und Bildsynthese. Springer 2005.

Application: Image mosaicing

- Original images are mapped onto virtual mosaic plane
- Interpolation and blending of color values

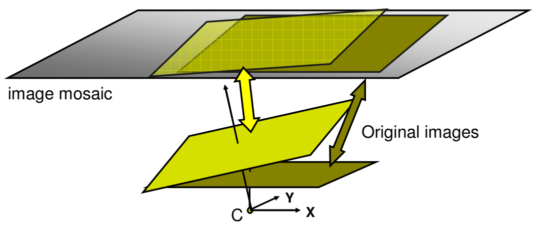
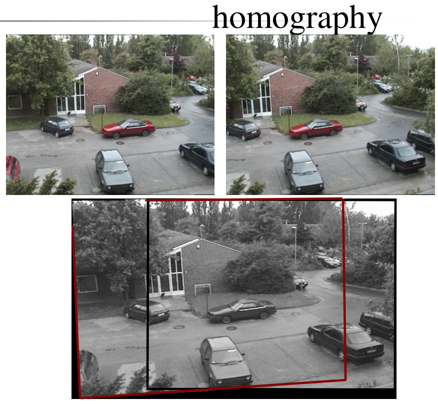
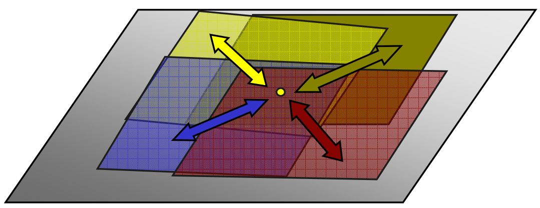


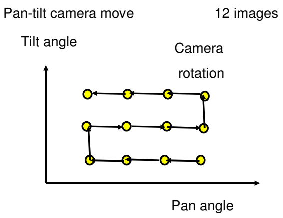
Image registration with homography



Global registration for mosaic

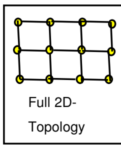


Global registration for mosaic



12 images

Global Registration and mapping



Multiview relations

- 2-view epipolar constraint
  - Uncalibrated cameras: Fundamental Matrix F
  - Calibrated cameras: Essential Matrix E
- Relative pose and structure
  - Relative pose estimation from E
  - 3D Structure triangulation
  - Pose estimation

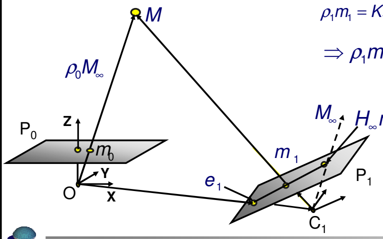
from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

The uncalibrated F-Matrix

Projection onto two views:

$$P_0 = K_0 R_0^T [I \quad 0]$$
$$\rho_0 m_0 = P_0 M = K_0 R_0^T [I \quad 0] M$$
$$\Rightarrow \rho_0 m_0 = K_0 R_0^T [I \quad 0] M_\infty$$

$$P_1 = K_1 R_1^T [I \quad -C_1]$$
$$\rho_1 m_1 = P_1 M = K_1 R_1^T [I \quad -C_1] M$$
$$= K_1 R_1^T [I \quad 0] M_\infty + K_1 R_1^T [I \quad -C_1] O$$
$$\rho_1 m_1 = K_1 R_1^T R_0 K_0^{-1} \rho_0 m_0 - K_1 R_1^T C_1$$
$$\Rightarrow \rho_1 m_1 = \underbrace{\rho_0 H_\infty m_0 + e_1}_{\text{Epipolar line}}$$



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

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

The Fundamental Matrix *F*

- The projective points e_1 and $(H_\infty m_0)$ define a plane in camera 1 (epipolar plane ⚐$_e$)
- the epipolar plane intersects the image plane 1 in a line (epipolar line u_e)
- the corresponding point m_1 lies on line u_e: $m_1^T u_e = 0$
- If the points (e_1),(m_1),$(H_\infty m_0)$ are all collinear, then the colinearity theorem applies: $m_1^T (e_1 \times H_\infty m_0) = 0$.

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

$$[e]_x = \begin{bmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{bmatrix}$$

Fundamental Matrix F

$$F = [e_1]_x H_\infty$$

Epipolar constraint

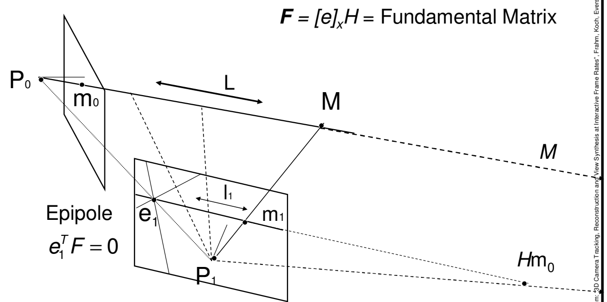
$$m_1^T F m_0 = 0$$

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

The Fundamental Matrix F

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

F = $[e]_x H$ = Fundamental Matrix



from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Estimation of F from images

- Given a set of corresponding points, solve linearly 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:

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

$$m_{1i}^T F m_{0i} = 0 \qquad \det(F) = 0 \ \ \text{(rank 2 constraint)}$$

Linear Estimation of F

solve F linearly with 8 correspondences using the normalized 8-point algorithm (Hartley 1995):

- normalize image coordinates of 8 correspondences for numerical conditioning
- solve the rank 8 equation **Af = 0** for the elements f_k of matrix **F**.
- apply the rank-2 constraint det(**F**)=0 as additional condition to fix epipole
- denormalize **F**.

$$\text{Foreach i = 1 to 8: } m_{1i}^T F m_{0i} = 0 \Rightarrow a_i^T \cdot f = 0$$

with $a_i = (x_{0i}x_{1i}, y_{0i}x_{1i}, w_{0i}x_{1i}, x_{0i}y_{1i}, y_{0i}y_{1i}, w_{0i}y_{1i}, x_{0i}w_{1i}, y_{0i}w_{1i}, w_{0i}w_{1i})$

and $f = (F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33})$

$$\text{Foreach i = 1 to 8: } a_i^T \cdot f = 0 \Rightarrow A_{(8x9)} f_{(9)} = \vec{0}_{(8)}$$

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

The Essential Matrix E

- F is the most general constraint on an image pair. If the camera calibration matrix K is known, then a calibrated matrix E can be computed using normalised coordinates $Km_p = m$:

$$m_1^T F m_0 = 0 \Rightarrow (Km_{p1})^T F (Km_{p0}) = 0$$

$$\Rightarrow m_{p1}^T (K^T F K) m_{p0} = m_{p1}^T (E) m_{p0} = 0$$

$$\Rightarrow E = K^T F K$$

$$F = [e]_x H_{ik} = [e]_x (K_k R_{ik} K_i^{-1})$$

$$E = [e]_x R_{ik} \qquad \det(E) = 0, \ \ EE^T E - \frac{1}{2} trace(EE^T) E = 0$$

- E holds the relative orientation of a calibrated camera pair. It has 5 degrees of freedom: 3 from rotation matrix R_{ik}, 2 from direction of translation e, the epipole.
- E has a cubic constraint that restricts E to 5 dof (Nister 2004)

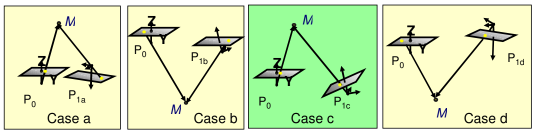
from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Relative Pose P from E

E holds the relative orientation between 2 calibrated cameras P_0 and P_1:

$$E = [e]_x R \Leftrightarrow P_0 = [I_{3x3} \quad 0_3], \ \ P_1 = [R \quad e]$$

Given $\mathbf{P_0}$ as coordinate frame, the relative orientation of $\mathbf{P_1}$ is determined directly from **E** up to a 4-fold rotation ambiguity ($\mathbf{P_{1a}}$ - $\mathbf{P_{1d}}$). The ambiguity is resolved by correspondence triangulation: The 3D point **M** of a corresponding 2D image point pair must be in front of both cameras. The epipolar vector **e** has norm 1.

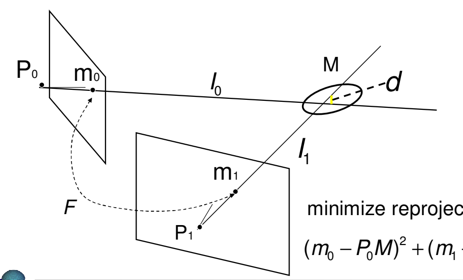


Relative Pose from E and correspondence: Case c is correct relative pose in this case

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

3D Structure Triangulation

- 3D Structure triangulation by intersection of rays from (m_0, m_1)
- M is reconstructed from rays (l_0, l_1)
- M has minimum distance of intersection between rays



$$\|d\|^2 \Rightarrow \min!$$

constraints:

$$l_0^T d = 0$$

$$l_1^T d = 0$$

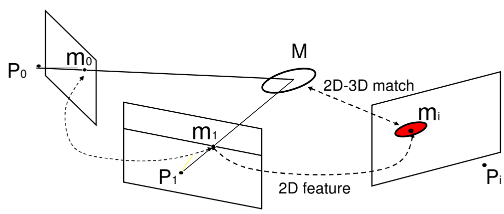
minimize reprojection error:

$$(m_0 - P_0 M)^2 + (m_1 - P_1 M)^2 \Rightarrow \min!$$

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Camera Pose from 2D-3D correspondences

- 3D point M from triangulation of 2D correspondences
- 2D feature tracking from image 1 to image i
- 3D Pose estimation of P_i with m_i - P_i M => min. with DLT



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

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

Gold Standard Methods

- Gold standard methods are the best method, given a specific noise model (e.g. Gaussian noise on correspondences yields a Maximum Likelhood estimate)
- Gold standard methods are in general nonlinear optimizations that yield the unbiased minimum reprojection error
- The Gold standard methods are initialized with the linear projective estimates (DLT) of the entity (F,E,H,P,M) as described before
- The Gold standard is in general slow, but fast approximations exist. All (nonlinear) constraints are directly exploited.

from: "3D Camera Tracking, Reconstruction and View Synthesis at Interactive Frame Rates", Frahm, Koch, Evers-Senne

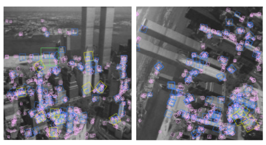
Schedule

- Introduction
- Multi-view Geometry
- Markerless Tracking
- Robust pose estimation
- Coffee Break
- AR-Toolkit
- Scene Modeling
- Applications

Matching vs. tracking

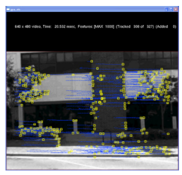
- Image-to-image correspondences are essential to 3D reconstruction

SIFT-matcher



Extract features independently and then match by comparing descriptors [Lowe 2004]

KLT-tracker



Extract features in first images and find same feature back in next view [Lucas & Kanade 1981] , [Shi & Tomasi 1994]
- Small difference between frames
- potential large difference overall

Optical flow

- Brightness constancy assumption

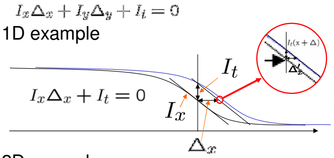
$$I(x+\Delta_x, y+\Delta_y, t+1) = I(x,y,t)$$

$$I(x+u, y+v, t+1) = I(x,y,t) + I_x\Delta_x + I_y\Delta_y + I_t$$ (small motion)

$$I_x\Delta_x + I_y\Delta_y + I_t = 0$$

- 1D example

$$I_x\Delta_x + I_t = 0$$

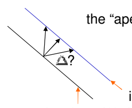


futher iterative refinement possible

- 2D example

$$I_x\Delta_x + I_y\Delta_y + I_t = 0$$ (1 constraint)

Δ_x, Δ_y (2 unknowns)



the "aperture" problem

isophote I(t+1)=I

isophote I(t)=I

© animation Pollefeys

Optical flow

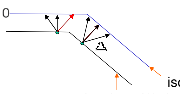
- How to deal with aperture problem?
  - 3 constraints if color gradients are different

$$R_x\Delta_x + R_y\Delta_y + R_t = 0$$
$$G_x\Delta_x + G_y\Delta_y + G_t = 0$$
$$B_x\Delta_x + B_y\Delta_y + B_t = 0$$

- Assume neighbors have same displacement

$$I_x(\mathrm{x})\Delta_x + I_y(\mathrm{x})\Delta_y + I_t(\mathrm{x}) = 0$$
$$I_x(\mathrm{x}')\Delta_x + I_y(\mathrm{x}')\Delta_y + I_t(\mathrm{x}') = 0$$

$I_x\Delta_x + I_y\Delta_y + I_t = 0$



isophote I(t+1)=I

isophote I(t)=I

© animation Pollefeys

Lucas-Kanade

- Assume neighbors have same displacement

$$I_x(\mathrm{x})\Delta_x + I_y(\mathrm{x})\Delta_y + I_t(\mathrm{x}) = 0$$
$$I_x(\mathrm{x}')\Delta_x + I_y(\mathrm{x}')\Delta_y + I_t(\mathrm{x}') = 0$$

least-squares:

$$\begin{bmatrix} I_x(\mathrm{x}) & I_y(\mathrm{x}) \\ I_x(\mathrm{x}') & I_y(\mathrm{x}') \\ I_x(\mathrm{x}'') & I_y(\mathrm{x}'') \end{bmatrix} \Delta = \begin{bmatrix} -I_t(\mathrm{x}) \\ -I_t(\mathrm{x}') \\ -I_t(\mathrm{x}'') \end{bmatrix} \qquad \mathrm{A}\Delta = \mathrm{b}$$

$$\left(\sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix}\right) \Delta = -\sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} I_t \qquad \mathrm{A}^\top\mathrm{A}\Delta = \mathrm{A}^\top\mathrm{b}$$

$$\Delta = (\mathrm{A}^\top\mathrm{A})^{-1}\mathrm{A}^\top\mathrm{b}$$

The small motion assumption

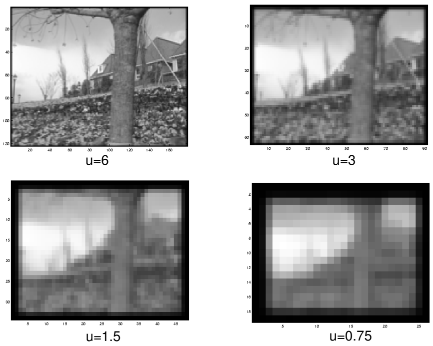
Is this motion small enough?

Most likely not—it's much larger than one pixel (not linear)

Solution?

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Reduce with Gaussian Pyramid!



*images from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Good feature to track

- Tracking

$$\left(\iint_W \begin{bmatrix}\frac{\partial I}{\partial x}\\ \frac{\partial I}{\partial y}\end{bmatrix}\begin{bmatrix}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\end{bmatrix} w(x,y)\mathrm{d}x\mathrm{d}y\right)\Delta = \iint_W \begin{bmatrix}\frac{\partial I}{\partial x}\\ \frac{\partial I}{\partial y}\end{bmatrix}(J-I)w(x,y)\mathrm{d}x\mathrm{d}y$$

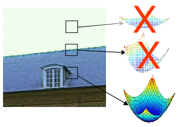
- Use same window in feature selection as for tracking itself

$$\mathrm{M} = \iint_W \begin{bmatrix}\frac{\partial I}{\partial x}\\ \frac{\partial I}{\partial y}\end{bmatrix}\begin{bmatrix}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\end{bmatrix} w(x,y)\mathrm{d}x\mathrm{d}y$$

maximize minimal eigenvalue of M

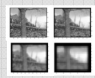
Strategy:

- Look for strong well distributed features, typically few hundreds
- initialize and then track, renew feature when too many are lost



KLT-Tracking Flow

Build-Pyramids
- build intensity pyramids from images ***I*, *J***
- build and gradient



Track

For all pyramid levels from coarse to fine
 For each feature ***f***
  For multiple iterations
  solve tracking equation **A d = b**
   evaluate **d** and update track of feature

If (replace needed)

Re-select-Features

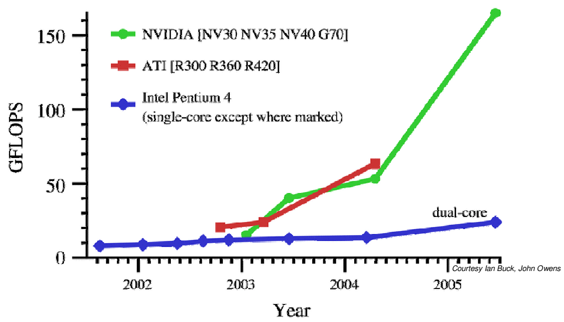
mask = mask_out_region (***ft_list***)
c_map = evaluate cornerness measure ***c*** over whole image
// Perform non-maximal suppression
pts = find_features (***#max_feats***, ***mask***, sort (***c_map***))
add_new_features (***ft_list***, ***pts***)

Results

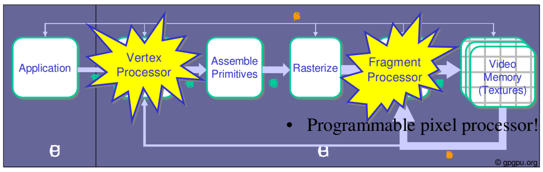
On CPU with 1000 features in 1024x768 video in 570-630 ms

2 weeks for
2.5M frames!

GP-GPU



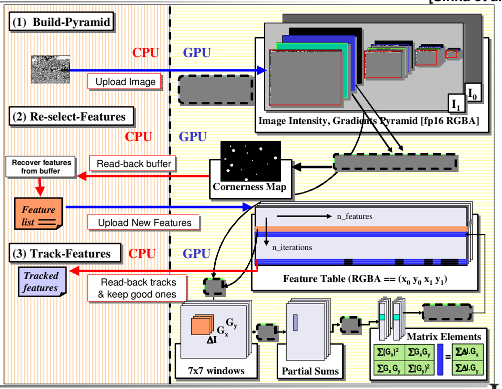
GPU



- Programmable pixel processor!
- Programmable vertex processor!

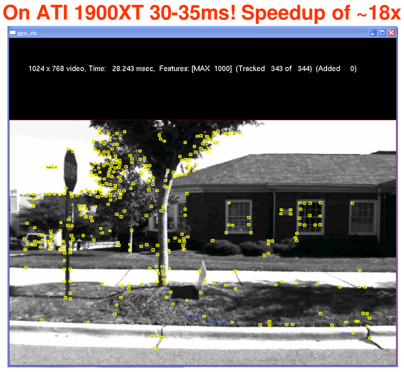
GPU-KLT

[Sinha et al. 2006]

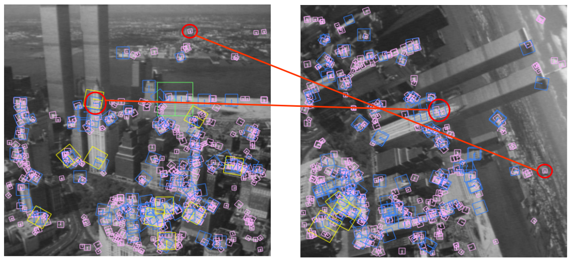


Results KLT-tracking

On ATI 1900XT 30-35ms! Speedup of ~18x



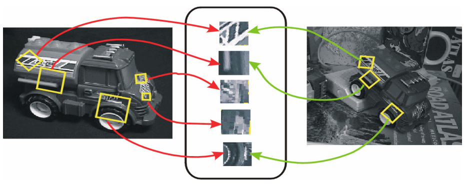
SIFT-detector



Problem: want to detect features at different scales (sizes) and with different orientations!

SIFT-detector

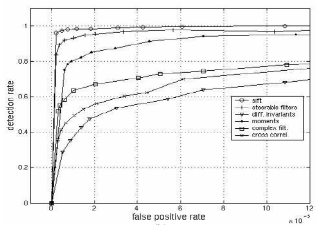
- Scale and image-plane-rotation invariant feature descriptor [Lowe 2004]
  - Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



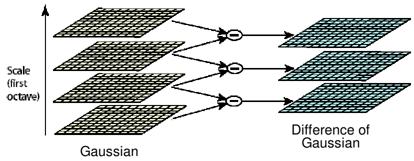
SIFT-detector

- Empirically found to perform very good **[Mikolajczyk 2003]**

Scale = 2.5
Rotation = 45⁰



Difference of Gaussian



- Difference-of-Gaussian with constant ratio of scales is a close approximation to Lindeberg's scale-normalized Laplacian [Lindeberg 1998]

Difference of Gaussian

- Difference-of-Gaussian with constant ratio of scales is a close approximation to Lindeberg's scale-normalized Laplacian [Lindeberg 1998]

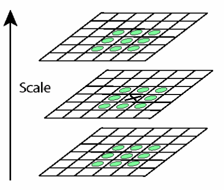
Key point localization

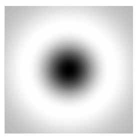
- Detect maxima and minima of difference-of-Gaussian in scale space
- Fit a quadratic to surrounding values for sub-pixel and sub-scale interpolation (Brown & Lowe, 2002)
- Taylor expansion around point:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

- Offset of extremum (use finite differences for derivatives):

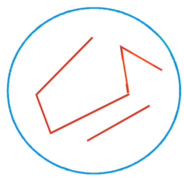
$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}$$





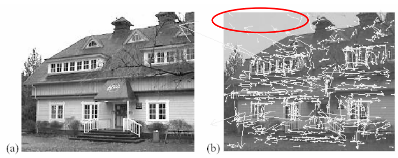
Orientation normalization

- Histogram of local gradient directions computed at selected scale
- Assign principal orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)



Example of keypoint detection

Threshold on value at DOG peak and on ratio of principle curvatures (Harris approach)

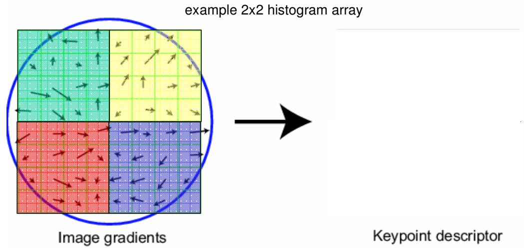


(a) 233x189 image
(b) 832 DOG extrema

SIFT vector formation

- Thresholded image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms
- 8 orientations x 4x4 histogram array = 128 dimensions

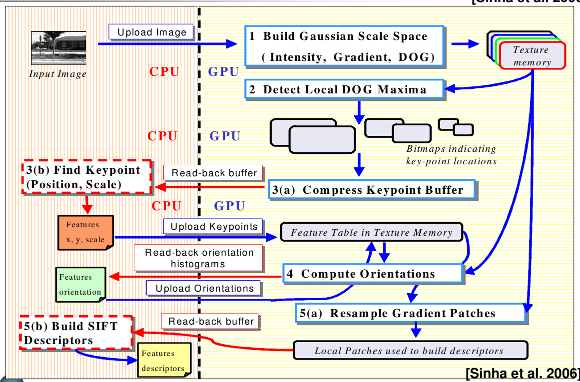
example 2x2 histogram array



Sift feature detector

GPU-SIFT-detector

[Sinha et al. 2006]



Robust pose estimation

- Problem: 2D Feature tracking/matching is not perfect!

⟹ data selection needed

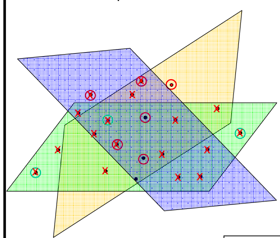
Schedule

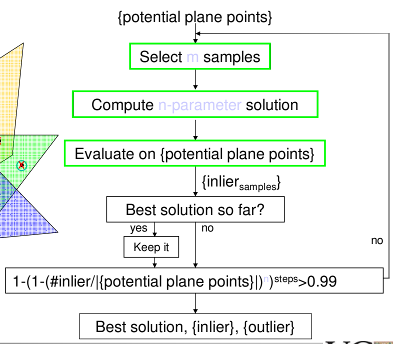
- Introduction
- Multi-view Geometry
- Markerless Tracking
- Robust pose estimation
- Coffee Break
- AR-Toolkit
- Scene Modeling
- Applications

Robust data selection: RANSAC

- Estimation of plane from point data

- outlier not on plane
- inlier not on plane

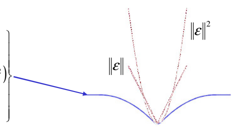




RANSAC: Evaluate Hypotheses

- Evaluate cost function

$$\begin{cases} \lambda^2\varepsilon^2 & 0 \le \lambda^2\varepsilon^2 \le \frac{c}{1+c} \\ 2\lambda\|\varepsilon\|\sqrt{c+c^2} - c\left(1+\lambda^2\varepsilon^2\right) & \text{if } \frac{c}{1+c} \le \lambda^2\varepsilon^2 < \frac{1+c}{c} \\ 1 & \text{else} \end{cases}$$



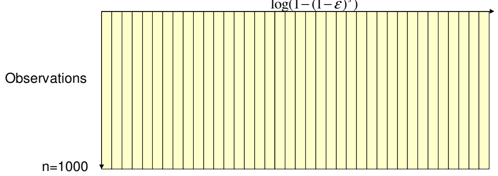
RANSAC Adaptive Stopping

Hypothesis Generator

Observation Likelihood correct sample

$P_s = 1-\left(1-\varepsilon_d^8\right)^S$ with ε_d inlier fraction

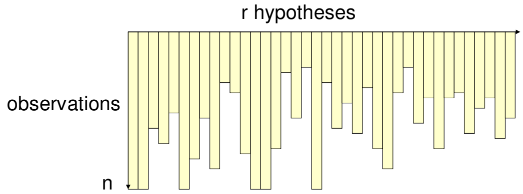
r= #required hypotheses $= \dfrac{\log(1-p)}{\log(1-(1-\varepsilon)^s)}$ with *p desired certainty*



r x n cost function evaluations for example r =500: 500 x 1000 = 500K

Preemptive RANSAC

Depth-first Preemption

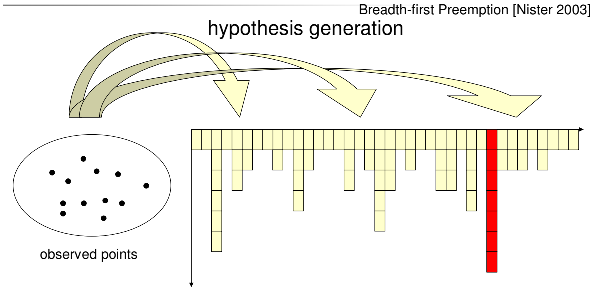


r x ???? = ???????

© animation D. Nister

Preemptive RANSAC

Breadth-first Preemption [Nister 2003]



Total Time for Preemptive RANSAC:

r * hypothesis generator+ 2*r*chunk size * observation likelihood

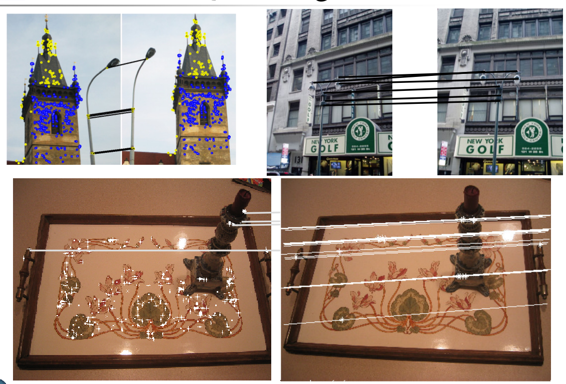
Overhead (~100 microseconds) (+Iterative Refinement)

© animation D. Nister

RANSACs

- Fast RANSACs
  - WaldSAC – Optimal Randomised RANSAC **[WaldSAC 2005]**
  - PROSAC - Progressive Sampling and Consensus **[Prosac 2005]**
  - LO-RANSAC – Locally optimized RANSAC **[LO-RANSAC 2003]**
- RANSACs for (Quasi-)degenerate data
  - DEGENSAC –Epipolar geometry for quasi-degenerate data **[DEGENSAC 2005]**
  - QDEGSAC – RANSAC for (quasi-)degenerate data **[QDEGSAC 2006]**

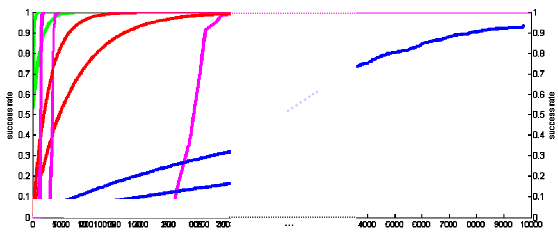
Quasi-degnerate data



Problem

$P_s = 1-\left(1-\varepsilon_d^8\right)^S$ with $\varepsilon_d = 0.9$

$P_s = 1-\left(1-\sum_{j=0}^{6}\binom{m}{j}\varepsilon_d^j(\varepsilon-\varepsilon_d)^{m-j}\right)^S = 1-(1-0.02)^S$

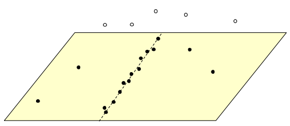


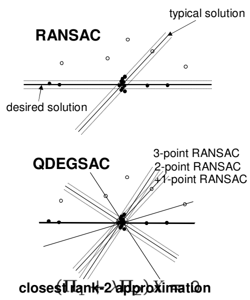
QDEGSAC

[Frahm and Pollefeys 2006]

- Robust estimation for (quasi-)degenerate data configurations

  Example: points on plane, but mostly on line





Robust rank estimation of data-matrix

$$\mathbf{A}X = 0$$

(i.e. large % of rows approx. fit in a lower dim. subspace)
(works for any linear estimation problem)

closest rank-2 approximation

Robust Pose

Bootstrap	known 3D points/markers
{2D-2D correspondences}	{2D-3D correspondences}
↓	↓
E-RANSAC(5-point algorithm)	RANSAC(3-point algorithm)
↓ E	↓ P_3
Estimate P_0,P_1	nonlinear refinement with all inliers
↓ P_0,P_1	↓ P_3
nonlinear refinement with all inliers	
↓ P_0,P_1	
triangulate points	

Assumption: Camera is internally calibrated

Coffee Break

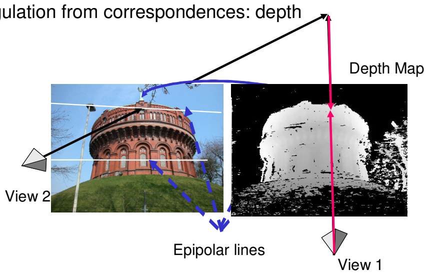
Please be back at 10:45 am.

Schedule

- Introduction
- Multi-view Geometry
- Markerless Tracking
- Robust pose estimation
- Coffee Break
- Scene Modeling
- AR-Toolkit
- Applications

Dense Depth Estimation

- Required: Calibration for all views
- pairwise: correspondence search along epipolar lines
- triangulation from correspondences: depth



Known Stereo Algorithms

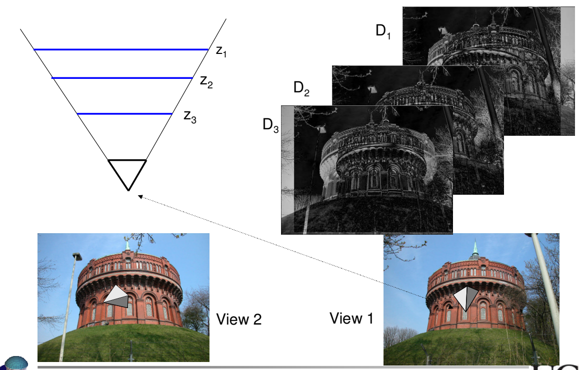
- Multiview Stereo results [http://vision.middlebury.edu/mview]
  - Dino, Sparse Ring, 16 images, comparable quality, normalized @3GHz
    - Furukawa UIUC 2006: 360 min
    - Hernandez CVIU 2004: 106 min
    - Pons CVPR 2005: 3 min
    - Vogiatzis CVPR 2005: 40 min
- (Near-) Interactive
  - [Woetzel, Koch 04] 4 images 1280x960: 760 ms
  - UNC Plane Sweep
- Here only: Plane Sweep Multiview Stereo

Correspondence Search

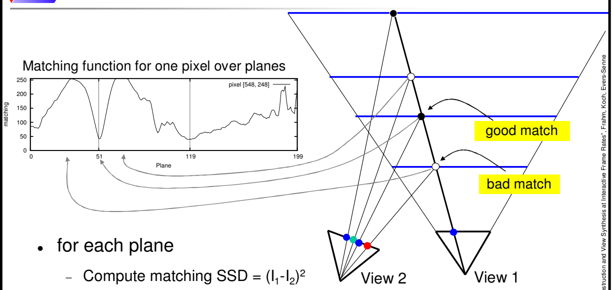
- Classic stereo
  - for each pixel x in I_1
    - for each pixel y on epipolar line in I_2
      - compute similarity of regions around x and y
      - similarity function: SAD, SSD, NCC, ...
    - chose correspondence with maximum similarity
    - add some constraints
- Plane Sweep Stereo [Collins 96]
  - for planes with distance z_i coplanar to I_1
    - project I_1 and I_2 onto plane
    - compute similarity image D_i from projected I_1 and I_2 ($D_i = || I_1 - I_2 ||$)
  - per pixel: chose maximum over all similarity images
- Plane Sweep: Perfectly suited for GPU usage [Yang, Welch, Bishop, 02]

SAD: Sum of Absolute Differenes
SSD: Sum of Squared Differences
NCC: Normalized Cross Correlation

Plane Sweep Stereo



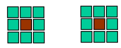
Match Selection

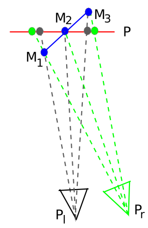


- for each plane
  - Compute matching SSD = $(I_1 - I_2)^2$
- Chose minimum dissimilarity as best match
- Avoid multiple minima

Region Matching

- Block Matching (SSD, SAD)
  - possible but expensive on GPU
  - 3x3: 18 texture look-ups instead of 2 (bilinear filtered!)
  - problems with perspective distortion
- Pyramid matching
  - create resolution pyramid image
  - match on every level $(I_1 - I_2)^2$
  - sum-up all levels i SSD = $\Sigma_i(I_1 - I_2)^2$
  - implicit correlation window
  - Supported by MIPMAP-textures

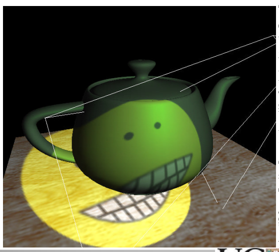




[Yang, Pollefeys 03]

Projective Texture Mapping

- Project texture onto geometry
  - use projection matrix $P = K [R^T | -R^T C]$ from calibration
  - adapt K to K_{Tex} to map to [0,1]x[0,1]: $P_{tex} = K_{tex} [R^T | -R^T C]$
  - compute texture coordinates from vertices: $m_{tex} = P_{tex} M$
  - Result: Homography
    - polygon <=> image plane
- Can be automated on GPU
  - texture coordinate generation facility



Courtesy: NVidia

Plane Sweep on the GPU

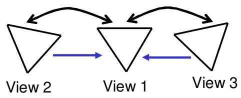
For all planes i at depth z_i
do {

- First Pass:
  - set virtual camera according to view 1
  - setup projective texture mapping for two texture units
  - setup similarity-shader
  - render quad as plane at distance z_i
  - store result as difference image D_i
- Second Pass:
  - Set virtual camera to ortho
  - load difference image (1.pass) as texture (D_i)
  - load accumulation image as texture (A)
  - render quad with shader for each pixel x:
    - if $D_i(x) < A(x)$ then (accept fragment)
      - $A(x) = D_i(x)$;
      - $Z(x) = z_i$ (Update z-buffer)

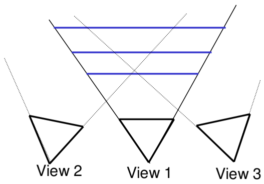
}
Read depth map from z-buffer

Fusion vs. Multiview P-S

- Classic Multiview Fusion
  - compute disparity maps pairwise
  - fuse disparities into single depth map



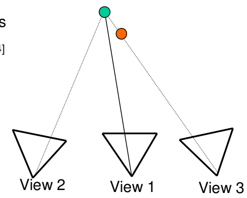
- Multiview Plane Sweep
  - use multiple support views at once
  - Similarity metric has to be adapted (Shader)



[Woetzel, Koch 04],[Nozick, Michelin, Arques 06]

Multiview Plane Sweep

- For each plane
  - compute pairwise matching for I1,I2,I3
  - I1-I2, I1-I3
  - select best combined matching as score for this plane
- Problem: Occlusions (outlier)
  - combine matches with small differences
    - discard up to two outlier [Woetzel, Koch 04]
    - statistical approach using average and variance [Nozick, Michelin, Arques 06]



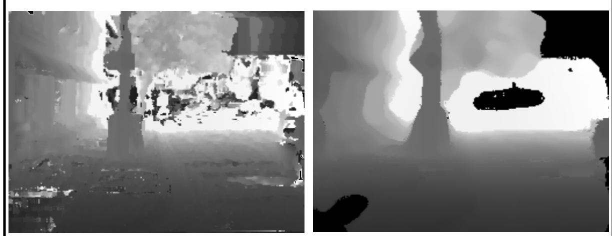
Plane Sweep Results

- Performance:
  - 11 Images @ 512 x 384 RGB
  - Out: 512 x 384, 48 planes
  - 7Hz (140ms)

NVidia GeForce FX 7900

Additional Fusion

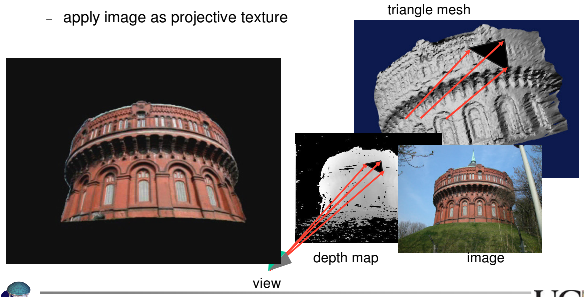
- Each depth map from 11 views
- fuse many depth maps (more details in section Applications)



What now?

Surface Modelling

- Generate 3D mesh from depth map
  - triangles based on 2D neighbourhood
  - backproject each vertex with depth value
  - apply image as projective texture

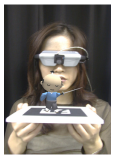


Schedule

- Introduction
- Multi-view Geometry
- Markerless Tracking
- Robust pose estimation
- Coffee Break
- Scene Modeling
- AR-Toolkit
- Applications

What is ARToolKit?

- Library for vision-based AR applications
  - Open Source, multi-platform
- Overlays 3D virtual objects on real markers
  - Uses single tracking marker (or multiple planar markers)
  - Determines camera pose information (6 DOF)
- Includes utilities, samples for marker-based interaction
- ARToolKit Website
  - http://artoolkit.sourceforge.net/ sourceforge project
  - http://www.hitl.washington.edu/artoolkit/ documentation (API, installation, tutorials,etc.)

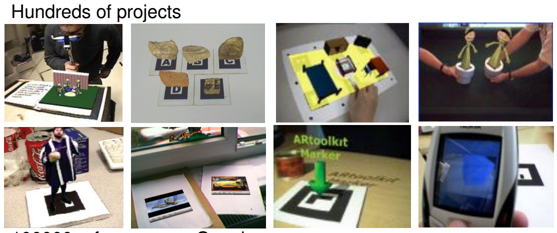


ARToolKit Characteristics

- Enabling technology
- Solves two significant problems in AR
  - Tracking
  - Interaction
- Tracking
  - Cheap vision based tracking
- Interaction
  - Object-based AR (Tangible AR)

ARToolKit in the World

- The most used AR library
- Hundreds of projects



- 100000 references on Google
- ~ 1000 downloads each month
- Company providing ARToolKit support (ARToolWorks)

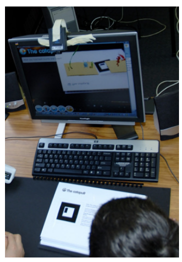
Overview

- Hardware
- Software
- Tracking Principle
- Performances
- Next Generation: ART4 and OSGART

Hardware (Desktop Setup)

- Camera
  - 320x240+
- Computer
  - Pentium IV
  - 3D graphics video card
  - USB/Firewire input

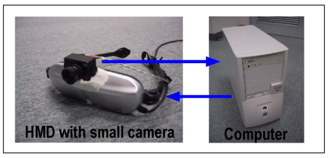
  (or Video capture card)



Note: the software can run on a simple Pentium 500Mhz !!

Hardware (HMD Setup)

- In addition: HMD
  - Video see-through or optical
  - Monocular or binocular



Typical ARToolKit System

- Pentium IV 1Ghz - $1400
- GeForce FX Graphics - $65
- Logitech/Creative USB Camera - $50

  Total Cost ~ NZ$ 1500

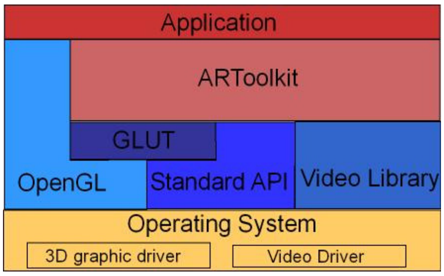
- eMagin 3D Visor Display - $1000

  Total Cost ~ NZ$ 2500

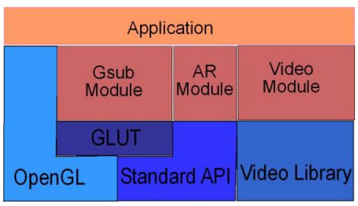
Software

- ARToolKit : version 2.40 or later
  - libAR – tracking
  - libARVideo – video capturing
  - libARgsub – image drawing
- Since 2.7: change in drawing lib
  - libARgsub_lite - image drawing (more efficient)
- OS: Windows, Linux, Macintosh
  (also IRIX, windows CE, symbian OS)
- Language: C (Wrapper for C++, .NET, Java etc.)

ARToolKit Structure



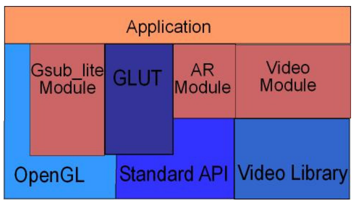
ARToolKit Structure with gsub



Three key libraries:

- libAR.lib – ARToolKit image processing functions (AR MOD.)
- libARvideo.lib – Wrapper to platform dependent video grabber class (VIDEO MOD.)
- libARgsub.lib/libARgsubUtil.lib – ARToolKit graphics functions (GSUB MOD.)

ARToolKit Structure with gsub_lite



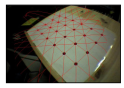
- Gsub_lite replace gsub:
  - libARgsub_lite.lib – ARToolKit graphics functions (GSUB_LITE MOD.)
- Differences:
  - More efficient main loop, OpenGL-transparent, Optimized graphics subroutines

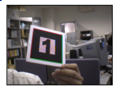
Software: Needs and Useful

- Additional basic libraries
  - Video capture library (DsVideoLib, Video4Linux, Quicktime)
  - OpenGL
  - GLUT
- Other useful libraries
  - Open VRML, Open Inventor, WTK, etc
  - FMod, OpenAL, etc.

Utilities

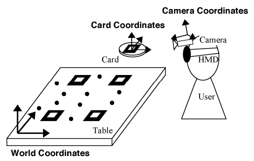
- Calibrate the camera: Two camera calibration methods
  - Accurate 2 step method (*calib_camera2.exe*)
  - Easy 1 step method (*calib_cparam.exe, calib_dist2.exe*)
- Test your system:
  - Video test (*videoTest.exe*)
  - OpenGL test (*graphicsTest.exe*)
- Make Patterns
  - Interactive program (*mk_patt.exe*)



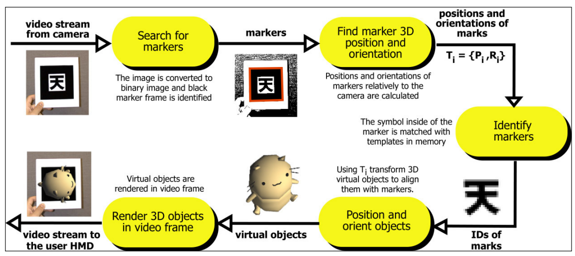



Interaction Support

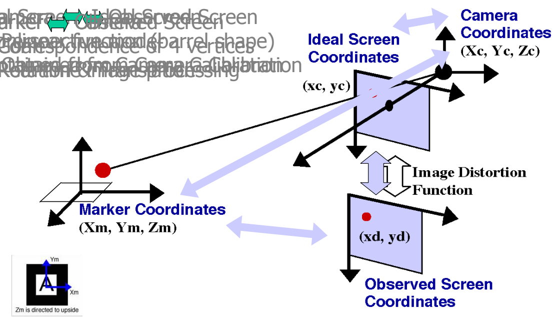
- Two types of Interaction mode
  - **Local**: Actions determined from single camera to marker transform
    - shaking, appearance, relative position, range
  - **Global**: Actions determined from two relationships
    - marker to camera, world to camera coords.
    - Marker transform determined in world coordinates
      - object tilt, absolute position, absolute rotation, hitting



ARToolKit Tracking: Main Principle



Coordinates for marker tracking



Relationships: Marker & Camera

- Rotation & Translation

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix}$$

$$= \mathbf{T}_{CM} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix}$$

Relationships: Camera & Ideal Screen

- Perspective Projection

$$\begin{bmatrix} hX_I \\ hY_I \\ h \end{bmatrix} = \begin{bmatrix} sf_x & 0 & x_c & 0 \\ 0 & sf_y & y_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \mathbf{C} \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix}$$

C : Camera Parameter

Relationships: Marker & Screen

$$\begin{bmatrix} hX_I \\ hY_I \\ h \end{bmatrix} = \mathbf{C} \cdot \mathbf{T}_{\mathbf{CM}} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} sf_x & 0 & x_c & 0 \\ 0 & sf_y & y_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix}$$

C : Camera Parameter

Scaling Parameter for Size Adjustment

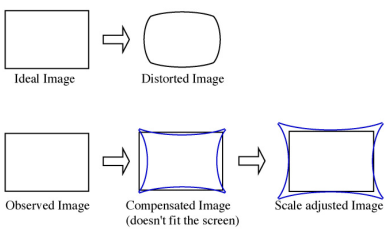


Image Distortion Parameters

- Relationships between Ideal and Observed Screen Coordinates

$$d^2 = (x_I - x_0)^2 + (y_I - y_0)^2$$
$$p = \{1 - fd^2\}$$
$$x_O = p(x_I - x_0) + x_0, \quad y_O = p(y_I - y_0) + y_0$$

(x_0, y_0) : Center Coordinates of Distortion

f: Distortion Factor

Image Distortion parameters

$$x = s(x_i - x_0), y = s(y_i - y_0)$$
$$d^2 = x^2 + y^2$$
$$p = \{1 - fd^2\}$$
$$x_d = px + x_0, \quad y_d = py + y_0$$
$$dist\_factor[0] = x_0$$
$$dist\_factor[1] = y_0$$
$$dist\_factor[2] = 100000000.0 * f$$
$$dist\_factor[3] = s$$

External Parameters: summary

- **Camera parameters**
  - Transformation from camera coordinates to the ideal screen coordinates
  - Image distortion function

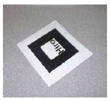
  => Camera calibration utility program
- **Definition of marker coordinates**
  - Origin and Size

  => Define in the code or in a configuration file
- **Pattern in the marker**
  - Pattern template

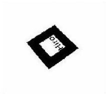
  => Pattern Maker utility program

First Step: Image Processing

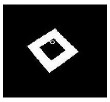
- Thresholding
- Labeling
  Feature Extraction (area position)
- Contour Extraction
- Four Straight Lines Fitting=>Rectangle
- Estimation of vertex positions



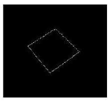
Original image



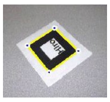
Thresholding



Connected Components



Contours

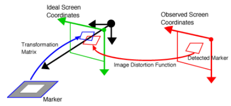


Extracted marker edges and corners

UC

Second Step: Getting T_{CM}

- Known Parameters
  - Camera Parameter: C
  - Image Distortion Parameters: x_0, y_0, f, s
  - Coordinates of 4 Vertices in Marker Coordinates Frame
- Obtained Parameters by Image Processing
  - Coordinates of 4 Vertices in Observed Screen Coordinates
- Goal
  - Getting Transformation Matrix from Marker to Camera



UC

Est. of Transformation Matrix

1st step: Geometrical calculation

- Rotation & Translation

2nd step: Optimization

- Iterative processing
  - Optimization of Rotation Component
  - Optimization of Translation Component

UC

Optimization of Rotation

- Observed positions of 4 vertices
- Calculated positions of 4 vertices
  - Positions in marker coordinates

    ↓ Estimated transformation matrix & Perspective matrix
  - Ideal screen coordinates

    ↓ Distortion function
  - Positions in observed screen coordinates
- Minimizing distance between observed and calculated positions by changing rotation component in estimated transformation matrix

UC

Search T_{cm} by Minimizing Error

- Optimization
  - Iterative process

$$\begin{bmatrix} h\hat{x}_i \\ h\hat{y}_i \\ h \end{bmatrix} = \mathbf{C} \cdot \mathbf{T}_{\mathrm{CM}} \begin{bmatrix} X_{Mi} \\ Y_{Mi} \\ Z_{Mi} \\ 1 \end{bmatrix}, \quad i = 1,2,3,4$$

$$err = \frac{1}{4} \sum_{i=1,2,3,4} \left\{ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right\}$$

UC

(2) Use of estimation accuracy

$$\begin{bmatrix} h\hat{x}_i \\ h\hat{y}_i \\ h \end{bmatrix} = \mathbf{C} \cdot \mathbf{T}_{\mathrm{CM}} \begin{bmatrix} X_{Mi} \\ Y_{Mi} \\ Z_{Mi} \\ 1 \end{bmatrix}, \quad i = 1,2,3,4$$

$$err = \frac{1}{4} \sum_{i=1,2,3,4} \left\{ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right\}$$

arGetTransMat() minimizes the *'err'*.

It returns this minimized *'err'*.

If *'err'* is still big,

Miss-detected marker.

⇨ Use of camera parameters by bad calibration.

UC

Initialization of Optimization Process

- Geometrical calculation based on 4 vertices coordinates
  - Independent in each image frame: Good feature.
  - Unstable result (Jitter occurs.): Bad feature
- Use of information from previous image frame
  - Needs previous frame information.
  - Cannot use for the first frame.
  - Stable results. (This does not mean accurate results.)
- ARToolKit supports both

Two types of initial condition

1. Geometrical calculation based on 4 vertices in screen coordinates

```
double arGetTransMat( ARMarkerInfo *marker_info,
                      double center[2], double width,
                      double conv[3][4] );
```

2. Use of information from previous image frame

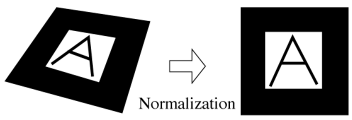
```
double arGetTransMatCont( ARMarkerInfo *marker_info,
                          double prev_conv[3][4],
                          double center[2], double width,
                          double conv[3][4] );
```

Third Step: Pattern Recognition

- Use of Inside pattern
- Why?
  - Square has symmetries in 90 degree rotation
    - 4 templates are needed for each pattern
  - Enable the use of multiple markers
- How?
  - Template matching
  - Normalizing the shape of inside pattern
  - Normalized correlation

Pattern Normalization

Getting projection parameter from 4 vertices position



$$\begin{bmatrix} hx_p \\ hy_p \\ h \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

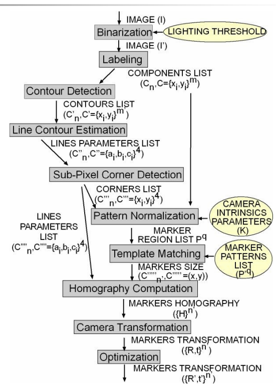
(x_p, y_p):pixel position in normalized image

(x_i, y_i):pixel position in input image

Normalization Correlation

$$s^{(l)} = \frac{\sum_{i=1}^{N} \left(x_i - \bar{x}\right) \cdot \left(x_i^{(l)} - \bar{x}^{(l)}\right)}{\sqrt{\sum_{i=1}^{N} \left(x_i - \bar{x}\right)^2} \sqrt{\sum_{j=1}^{N} \left(x_j^{(l)} - \bar{x}^{(l)}\right)^2}}$$

The "Big" Algo

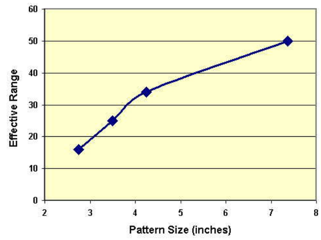


Accuracy vs. Speed

- Pattern normalization takes much time.
- This is a problem when using many markers.
- Normalization process.
- Digital Encoding solution (ex: ARTag, ARToolKit-Plus)

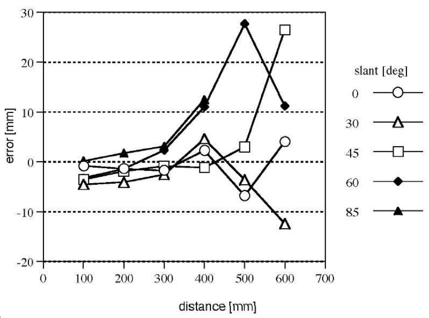
Normalization Resolution convert

Tracking Range with Pattern Size

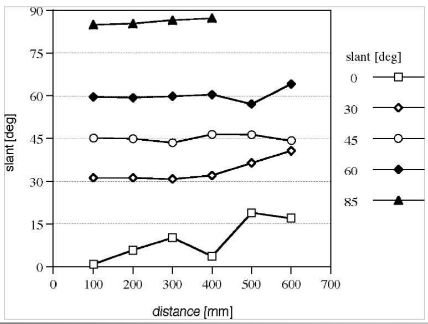


Rule of thumb – range = 10 x pattern width

Tracking Error with Range



Tracking Error with Angle



New Generation

- From ARToolKit 2.7.x to..
- ARToolKit 4
  - New Commercial Version
- OSGART
  - Multimedia Enhanced version

OSGART

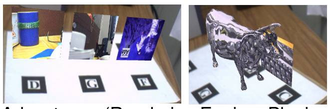
- Integration of ARToolKit on a High-Level Rendering Engine

  OSGART= OpenSceneGraph + ARToolKit
- Supporting Geometric+Photometric Registration

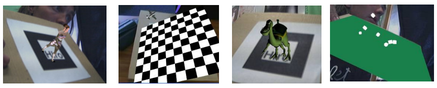
- ARToolkit “Spirit”
  - Multiplatforms (Windows, Linux, MacOS X)
  - “On the shelf”, “Plug’n Run”, Tutorials/Samples
  - Free for Academic, Licence for Industrial

OSGART:Features

- C++ (but also Python, Lua, etc).
- Multiple Video Input supports: Direct (Firewire/USB Camera), Files, Network by ARvideo, PtGrey, CVCam, VideoWrapper, etc.
- Video Objects



- Benefit of OSG Advantages (Rendering Engine, Plugins, etc)



More information

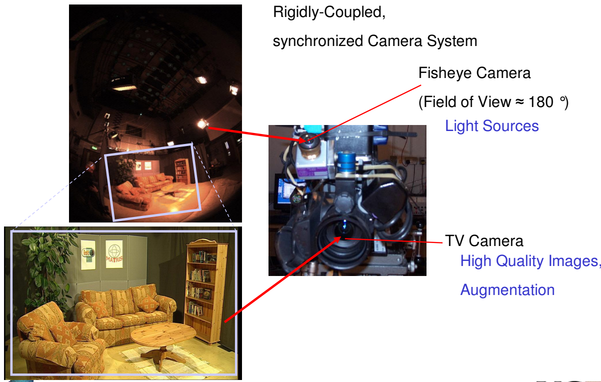
- Download:
  - http://artoolkit.sourceforge.net/
- Documentation:
  - online: http://www.hitl.washington.edu/artoolkit/
- Help:
  - forum: http://www.hitlabnz.org/forum/
  - Mailing list: http://www.hitl.washington.edu/artoolkit/community/

Schedule

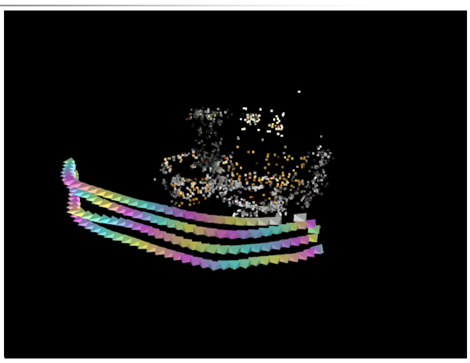
- Introduction
- Multi-view Geometry
- Markerless Tracking
- Robust pose estimation
- Coffee Break
- AR-Toolkit
- Scene Modeling
- Applications

1. System Overview

Frahm, Köser, Grest and Koch CVMP '05

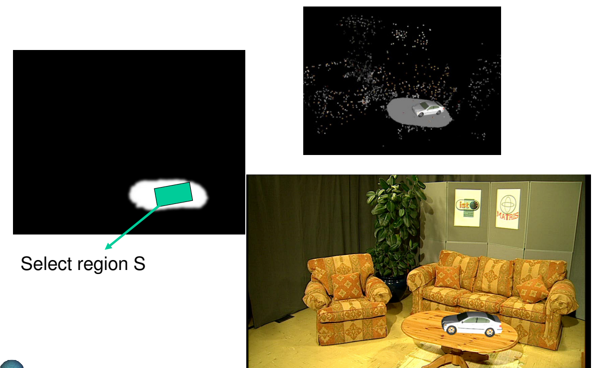


2. Camera Pose Estimation

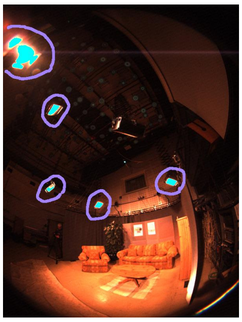


Poses and Reconstructed Scene Point Cloud

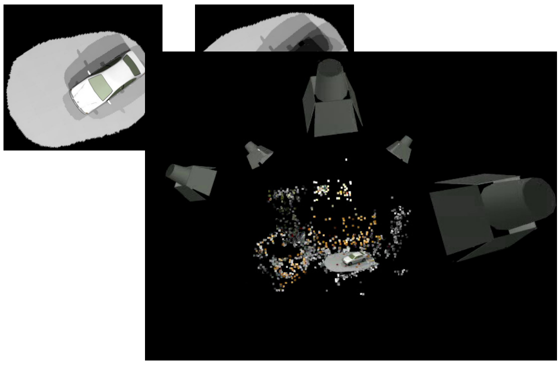
3. Object Positioning



4. Illumination

4. Illumination



5. Results

Augmentation (Talking people)

References

[Agapito 01] L. Agapito, E. Hayman, I. Reid. Self-Calibration of Rotating and Zooming Cameras. International Journal of Computer Vision, Volume 45 (2), November 2001.

[Baker & Matthews 04] S. Baker and I. Matthews, Lucas-Kanade 20 Years On: A Unifying Framework.International Journal of Computer Vision, 56(3):221–255, March 2004.

[Collins 96] Collins, R.T., "A Space-SweepApproach to True Multi-Image Matching", CVPR 1996

[DEGENSAC 2005], Chum et al.,Two-view geometry estimation unaffected by a dominant plane, *CVPR 2005*, vol. 1, pp 772-780

[Evers, Petersen, Koch 06] Evers-Senne, J.-F., Petersen, A. and Koch, R., " A Mobile Augmented Reality System with Distributed Tracking", 3DPVT 2006, Chapel Hill, NC,USA

[Fischler & Bolles 81] M.A. Fischler and R.C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6), June 81.

[Hartley 94] R. Hartley, Self-Calibration from Multiple Views with a Rotating Camera , in Proc. European Conf. Computer Vision, pp. 471-478, 1994.

[Hartley 95] R Hartley. In Defense of the 8-Point Algorithm, Proceedings of the 5th International Conference on Computer Vision, Cambridge (MA), pp. 1064-1070, 1995.

References

[Hartley, Zisserman 03] R. Hartley and A. Zisserman, Multiple View Geometry in computer vision, 2nd edition. Cambridge University Press, 2003.

[LO-RANSAC 2003], Chum, Matas, and Kittler, Locally Optimized RANSAC, *DAGM 2003*

[Lowe 04] David Lowe, Distinctive Image Features from Scale-Invariant Keypoints, **IJCV**, 60(2), 2004, pp91-110

[Lucas & Kanade 81] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In Proceedings International Joint Conference on Artificial Intelligence, 1981.

[Lindeberg 98], T. Lindeberg, "Feature detection with automatic scale selection," International Journal of Computer Vision, vol. 30, no. 2, 1998

[Mikolajczyk 03], K.Mikolajczyk, C.Schmid. “A Performance Evaluation of Local Descriptors”. CVPR 2003

[Nister 03] D. Nistér, Preemptive RANSAC for live structure and motion estimation, IEEE International Conference on Computer Vision (ICCV 2003), pp 199-206, 2003.

[Nister 04] D. Nistér,An efficient solution to the 5-point relative pose problem. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 26(6):756-770, June 2004.

[Nozick, Michelin, Arques 06] Nozick, V., Michelin, S. and Arques D., “Real-time Plane-Sweep with local startegy”, WSCG 2006, Plzen, Czech Republic

[Pollefeys 99] M. Pollefeys, R. Koch and L. Van Gool. Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters, International Journal of Computer Vision, 32(1), 7-25, 1999.

[Prosac 2005], O. Chum and J. Matas, Matching with PROSAC - progressive sample consensus, *CVPR 2005*, vol. 1, pp 220-226

References

[QDEGSAC 2006], Frahm and Pollefeys, RANSAC for (Quasi-)Degenerate Data (QDEGSAC), CVPR 2006

[Shi & Tomasi 94] Jianbo Shi and Carlo Tomasi, Good Features to Track**,** IEEE Conference on Computer Vision and Pattern Recognition 1994

[Sinha et al. 06], Sudipta N Sinha, Jan-Michael Frahm, Marc Pollefeys and Yakup Genc, "GPU-Based Video Feature Tracking and Matching", EDGE 2006, workshop on Edge Computing

[Verlani, Goswami, Narayanan 06] Verlani, P., Goswami, A., Narayanan, P.J., "Depth Images: Representation and Real-time Rendering", 3DPVT 2006, Chapel Hill, NC,USA

[WaldSAC 2005], Matas, Chum, Optimal Randomised RANSAC, *International Conference on Computer Vision*, Beijing, 2005

[Woetzel, Koch 04] Woetzel, J. and Koch, R., “Multi-camera realtime depth estimation with discontinuity handling on PC graphics hardware”, ICPR 2004

[Yang, Welch, Bishop, 02] Yang, R., Welch, G. and Bishop, G. , “ Real-Time Consensus-Based Scene Reconstruction using Commodity Graphics Hardware”, Pacific Graphics, 2002, Beijing

[Yang, Pollefeys 03] Yang, R. and Pollefeys, M., “Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware, CVPR 2003, Madison (WI), USA