相机模型与运动推断结构

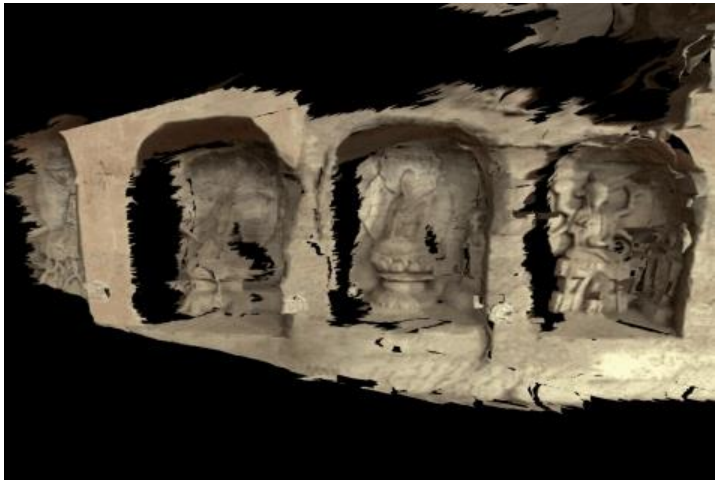
章国锋
浙江大学CAD&CG实验室

视频场景重建的流程

运动推断结构

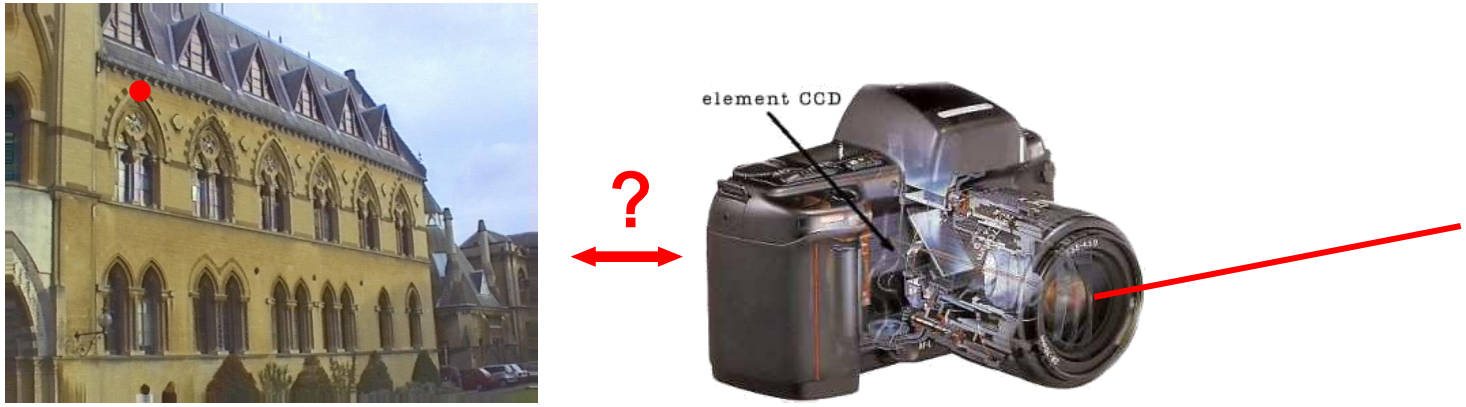
深度恢复

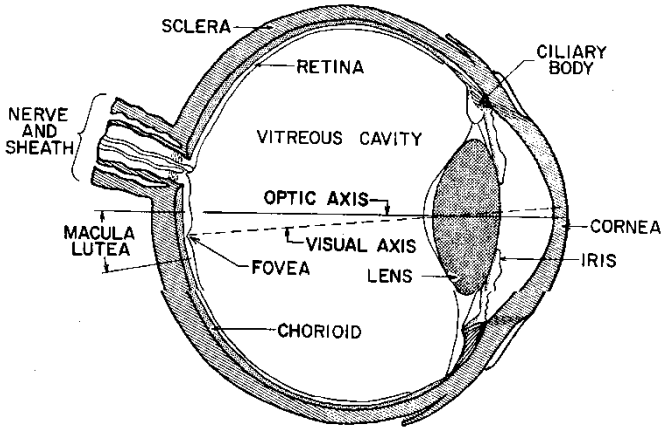
三维重建



Camera model

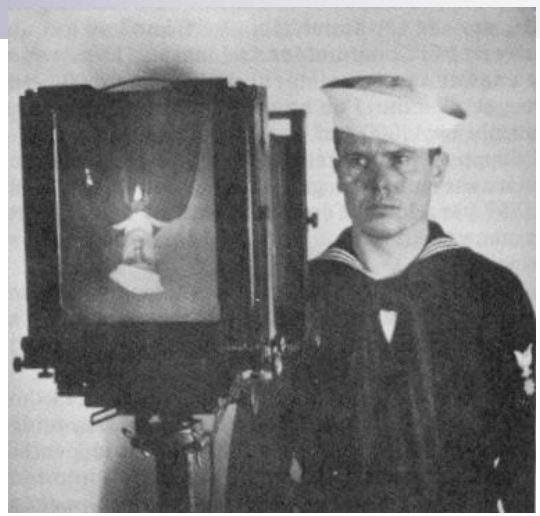
- Relation between pixels and rays in space
  - Pinhole Perspective Projection



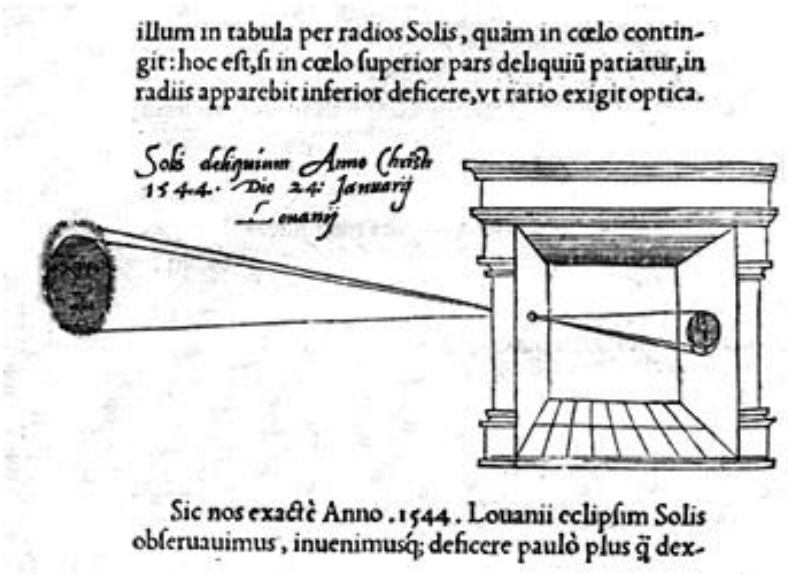


Animal eye:

a looonnng time ago.

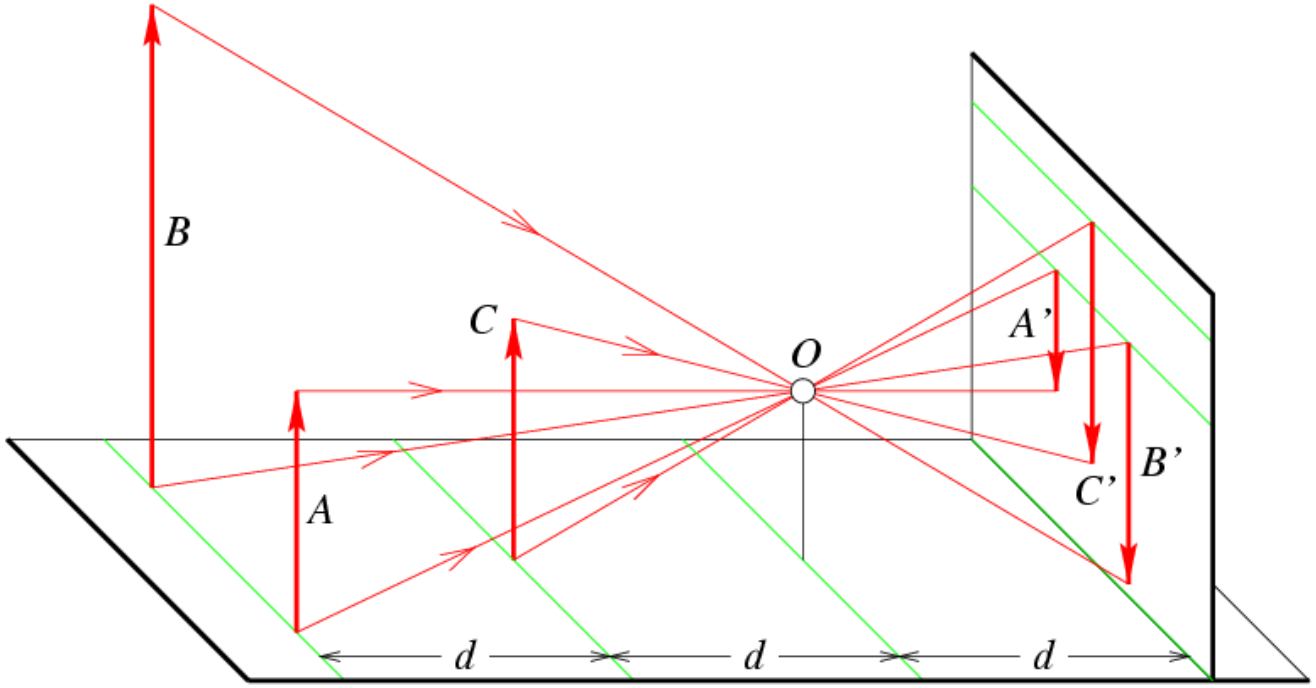


Photographic camera:
Niepce, 1816.



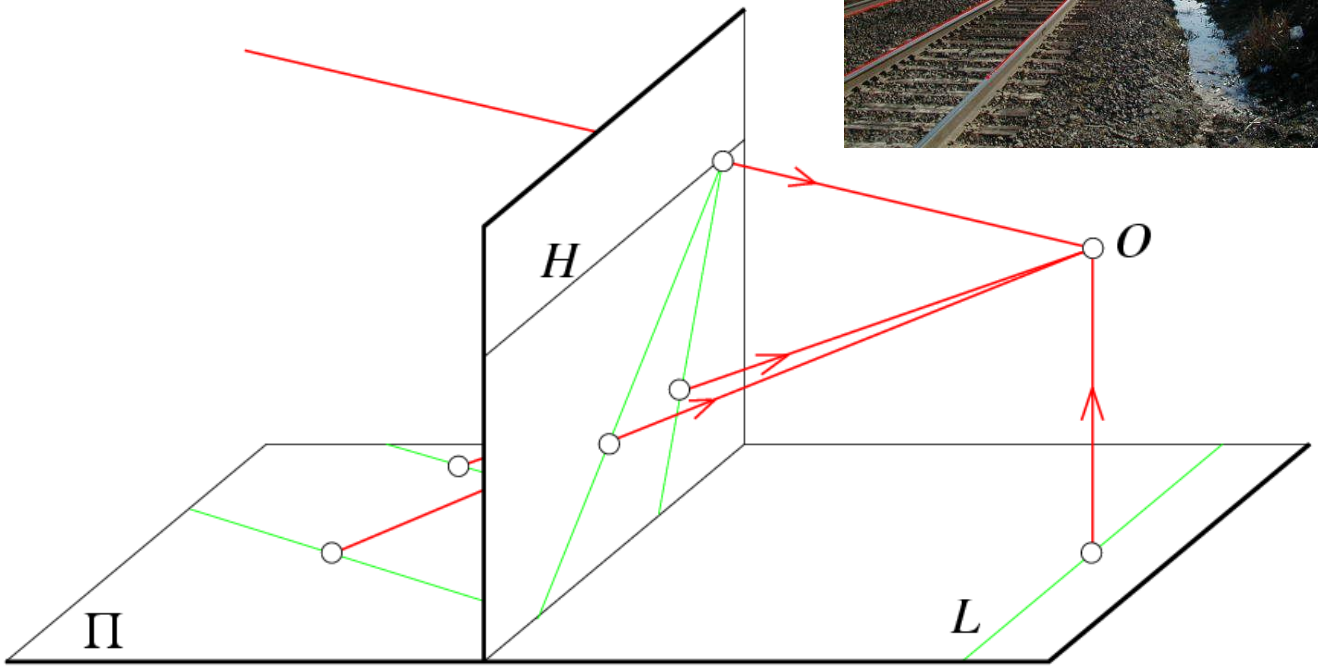
Pinhole perspective projection: Brunelleschi, XVth Century.
Camera obscura: XVIth Century.

Distant objects appear smaller

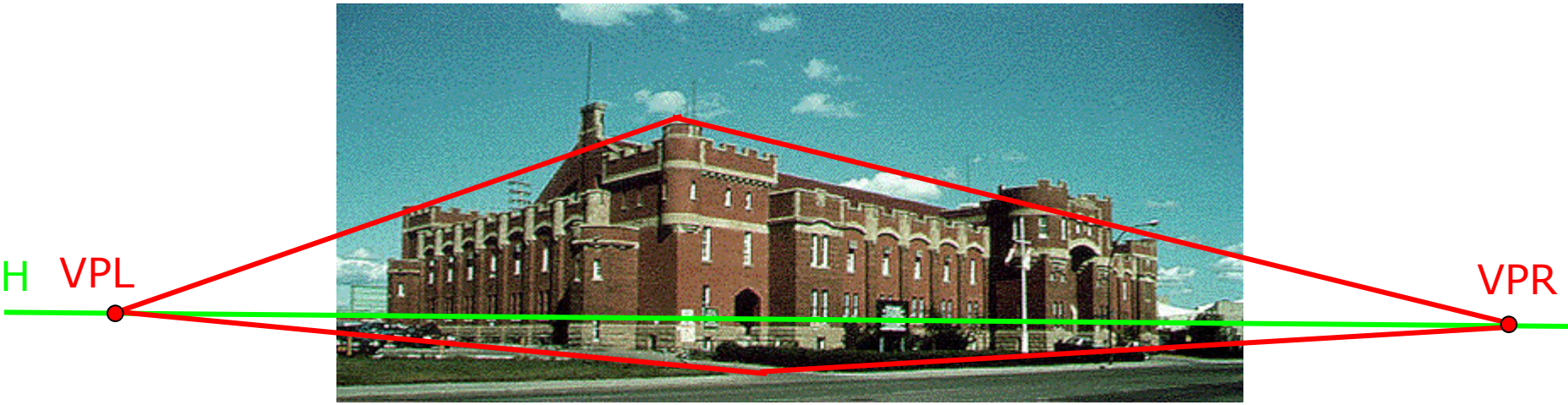


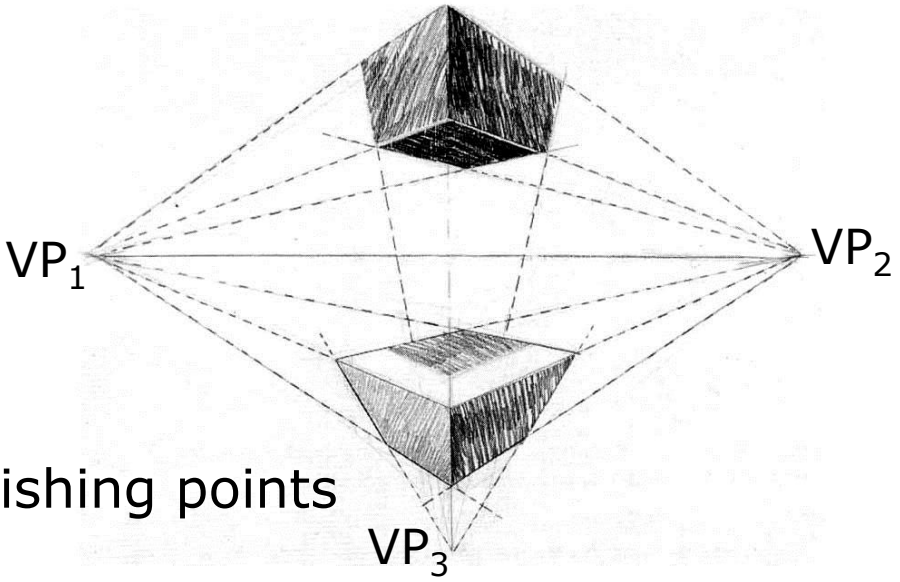
Parallel lines meet

- vanishing point



Vanishing points

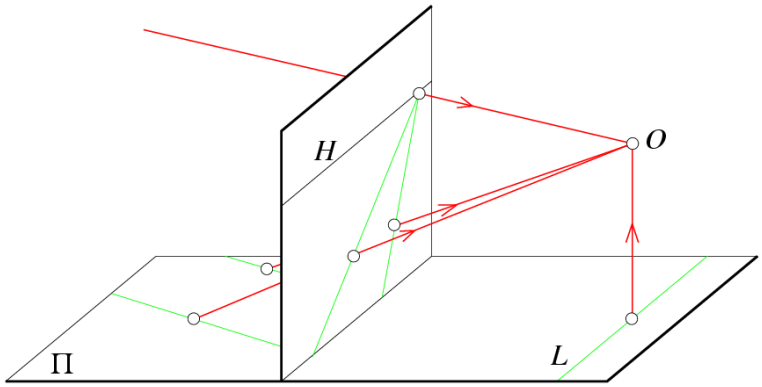




To different directions
correspond different vanishing points

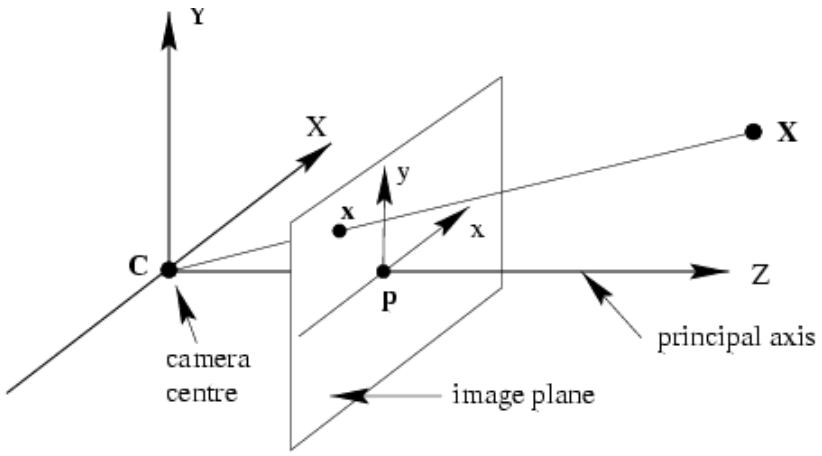
Geometric properties of projection

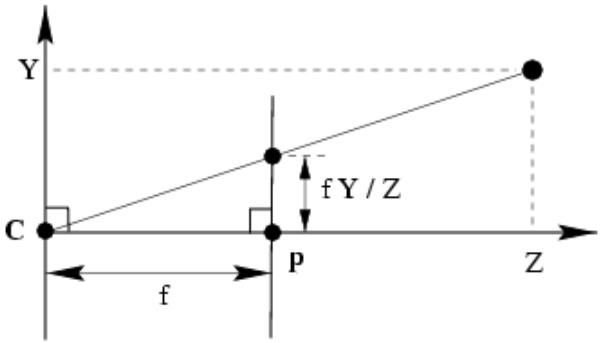
- Points go to points
- Lines go to lines
- Planes go to whole image
  or half-plane
- Polygons go to polygons



- Degenerate cases:
  - line through focal point yields point
  - plane through focal point yields line

针孔相机模型





投影方程：

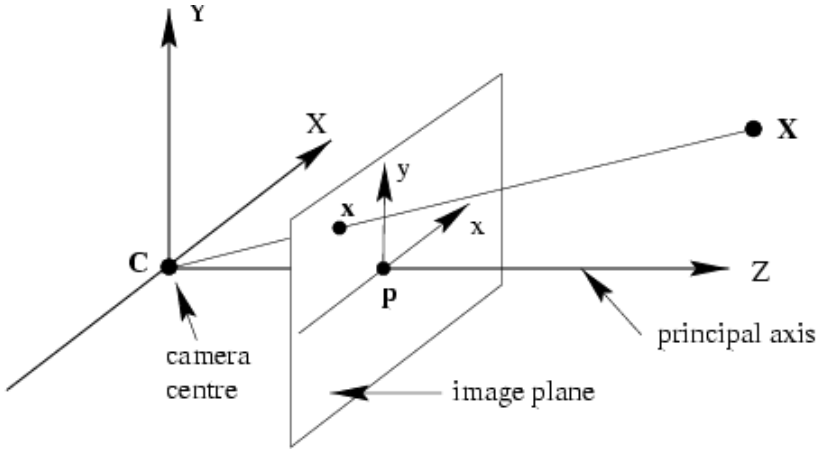
$$x = f\frac{X}{Z}$$

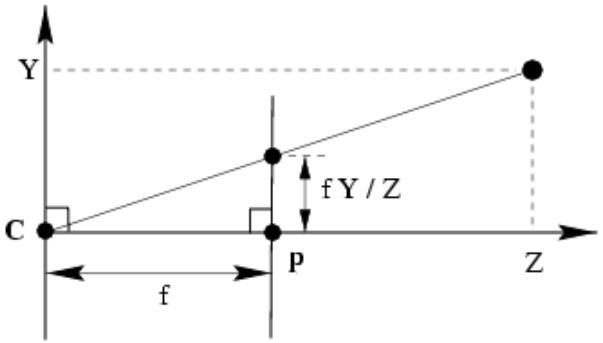
$$y = f\frac{Y}{Z}$$

齐次坐标表示：

$$\begin{pmatrix} x \\ y \\ f \end{pmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

针孔相机模型



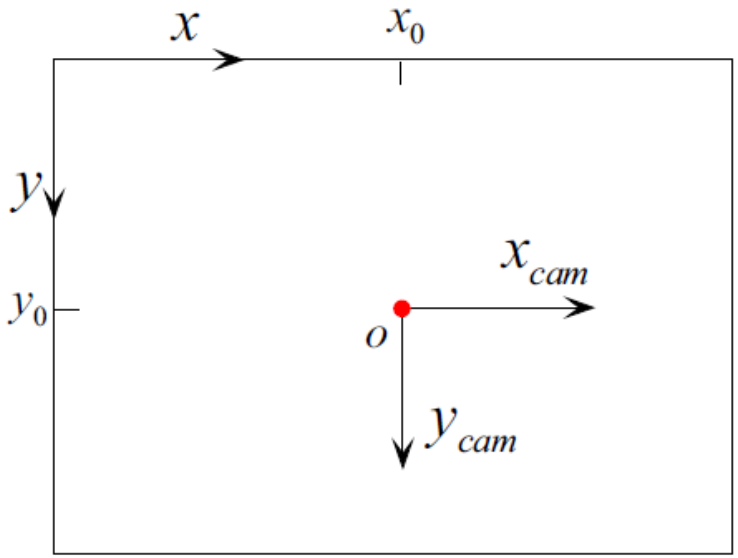


$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\mathrm{K} \qquad\qquad [\mathrm{R}|\mathrm{t}]$$

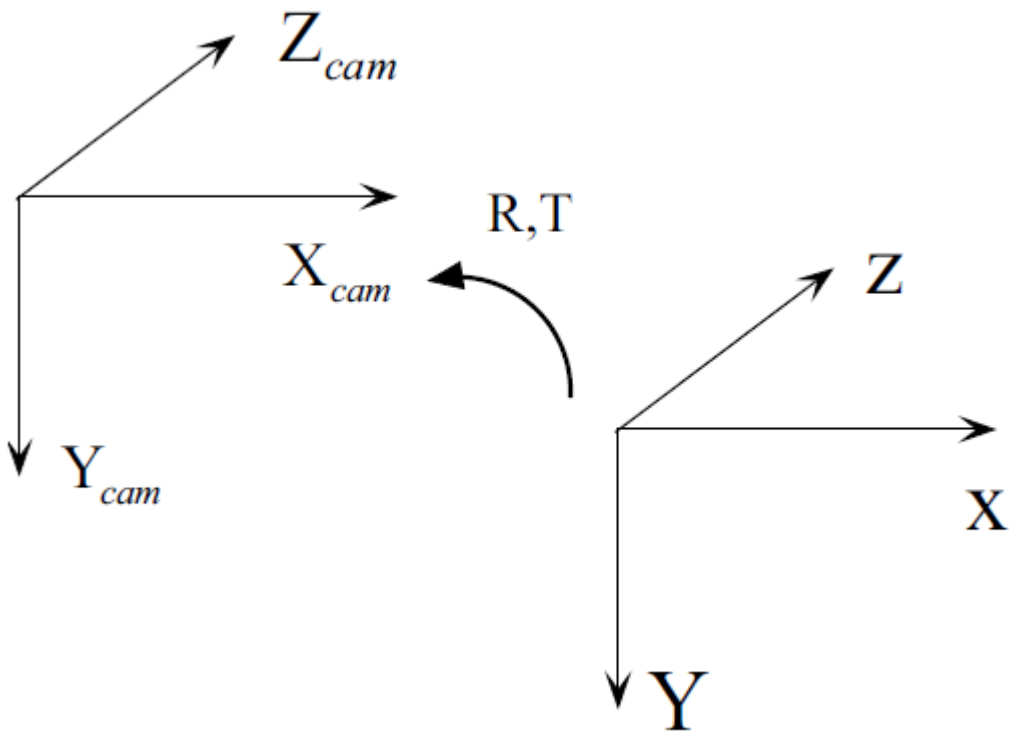
主点的偏移

principal point



$$\begin{pmatrix} fX/Z + x_0 \\ fY/Z + y_0 \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX + Zx_0 \\ fY + Zy_0 \\ Z \end{pmatrix} = \begin{bmatrix} f & & x_0 & 0 \\ & f & y_0 & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

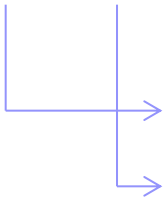
相机的外部参数



透视相机模型

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

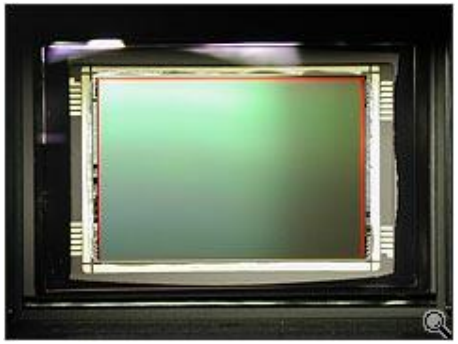
$$\mathrm{P} = \mathrm{K}[\mathrm{R} \mid \mathrm{t}]$$ 11 dof (5+3+3)



intrinsic camera parameters

extrinsic camera parameters

CCD camera



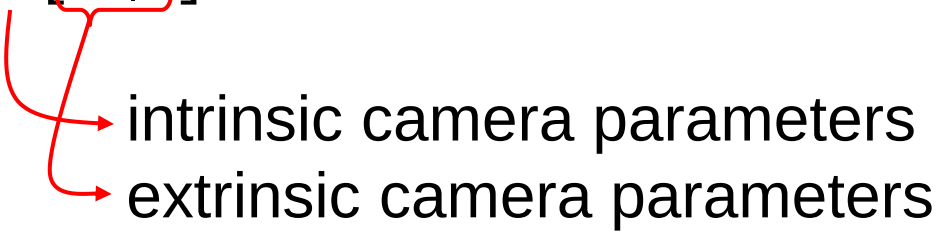
$$K = \begin{bmatrix} \alpha_x & & p_x \\ & \alpha_y & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$

General projective camera

$$K = \begin{bmatrix} \alpha_x & s & p_x \\ & \alpha_x & p_y \\ & & 1 \end{bmatrix}$$

$$\mathrm{P} = \mathrm{KR}[\mathrm{I} \mid \mathrm{t}]$$ 11 dof (5+3+3)

$$\mathrm{P} = \mathrm{K}[\mathrm{R} \mid \mathrm{t}]$$



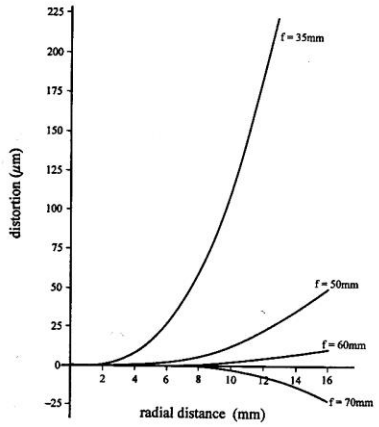
Radial distortion

- Due to spherical lenses (cheap)
- Model:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathtt{t} \\ 0_3^\top & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \right)$$

$$\mathbf{R}\,(x, y) = (1 + K_1(x^2 + y^2) + K_2(x^2 + y^2)^2 + \ldots) \begin{bmatrix} x \\ y \end{bmatrix}$$

straight lines are not straight anymore



barrel dist.

pincushion dist.

http://foto.hut.fi/opetus/260/luennot/11/atkinson_6-11_radial_distortion_zoom_lenses.jpg

Radial distortion example

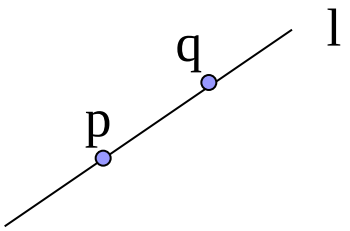
Homogeneous Notation

- A line **l** is represented by the homogeneous 3-vector

$$l = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix}$$

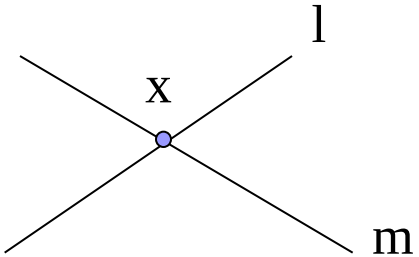
for the line $l_1x+l_2y+l_3 = 0$. Only the ratio of the homogeneous line coordinates is significant.

- point on line: $\mathbf{l}.\mathbf{x} = 0$ or $\mathbf{l}\mathbf{x} = 0$ or $\mathbf{x}\mathbf{l} = 0$



- two points define a line: $\mathbf{l} = \mathbf{p} \times \mathbf{q}$

- two lines define a point: $\mathbf{x} = \mathbf{l} \times \mathbf{m}$



Matrix notation for vector product

The vector product $\mathbf{v} \times \mathbf{x}$ can be represented as a matrix multiplication

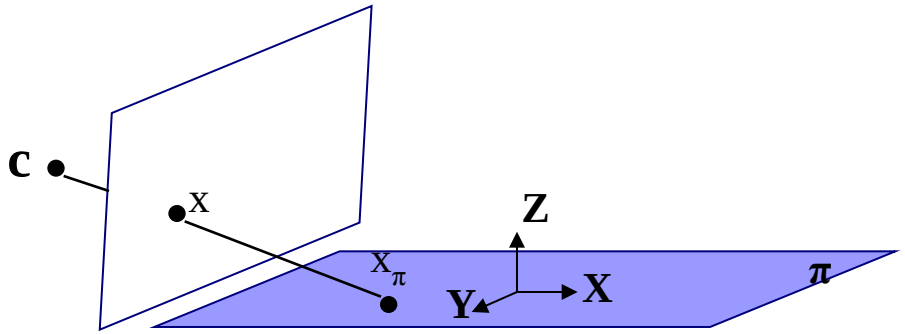
$$\mathbf{v} \times \mathbf{x} = [\mathbf{v}]_{\times} \mathbf{x}$$

where

$$[v]_{\times} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$$

- $[\mathbf{v}]_{\times}$ is a 3×3 skew-symmetric matrix of rank 2.
- $\mathbf{v}$ is the null-vector of $[\mathbf{v}]_{\times}$, since $\mathbf{v} \times \mathbf{v} = [\mathbf{v}]_{\times}\mathbf{v} = \mathbf{0}$.

Plane projective transformations



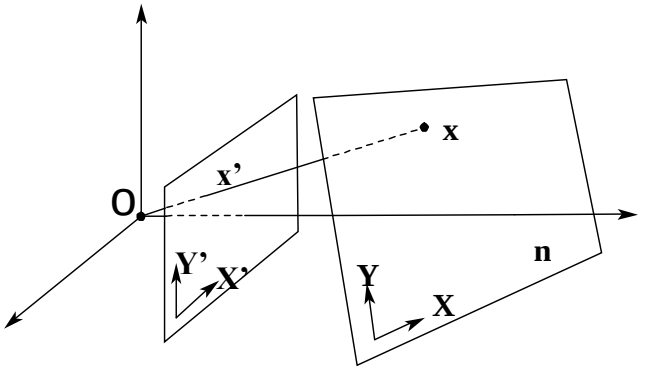
Choose the world coordinate system such that the plane of the points has zero Z coordinate. Then the 3×4 matrix P reduces to

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{21} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

which is a 3×3 matrix representing a general plane to plane projective transformation.

Projective Transformations Continued

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$



or **x** = H**x**, where H is a 3 × 3 non-singular homogeneous matrix.

- This is the most general transformation between the world and image plane under imaging by a perspective camera.
- It is often only the 3 × 3 form of the matrix that is important in establishing properties of this transformation.
- A projective transformation is also called a "homography" and a "collineation".
- H has 8 degrees of freedom.

Four points define a projective transformation

Given n point correspondences $(x, y) \leftrightarrow (x', y')$

Compute H such that $x_i' = \mathrm{H} x_i$

- Each point correspondence gives two constraints

$$x' = \frac{x_1'}{x_3'} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad y' = \frac{x_2'}{x_3'} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

and multiplying out generates two linear equations for the elements of H

$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13}$$

$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23}$$

- If $n \geq 4$ (no three points collinear), then H is determined uniquely.

Removing Perspective Distortion

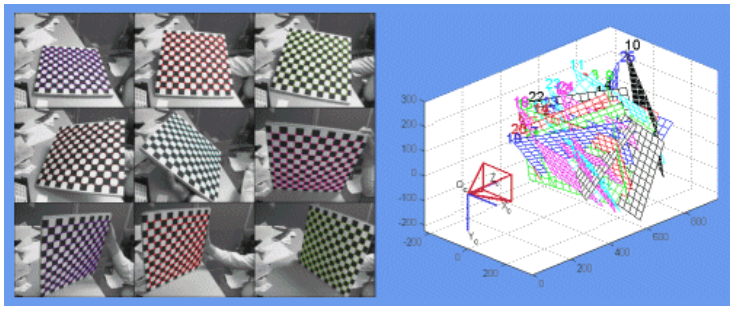
Given: the coordinates of four points on the scene plane

Find: a projective rectification of the plane

- This rectification does not require knowledge of any of the camera's parameters or the pose of the plane.
- It is not always necessary to know coordinates for four points.

Camera Calibration

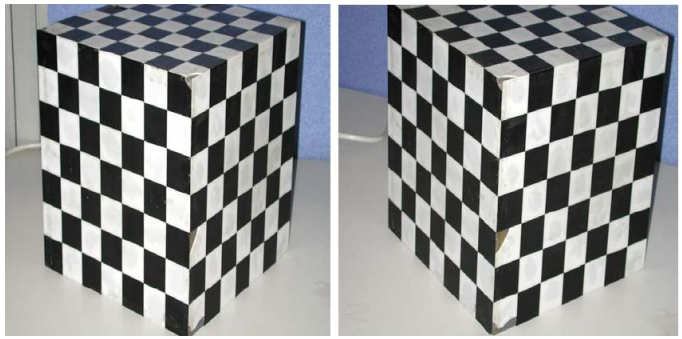
- Camera Calibration is to decide the relationship of the scenes and image
- Camera Parameters
  - Intrinsic parameters: K
  - Extrinsic parameters: R, t

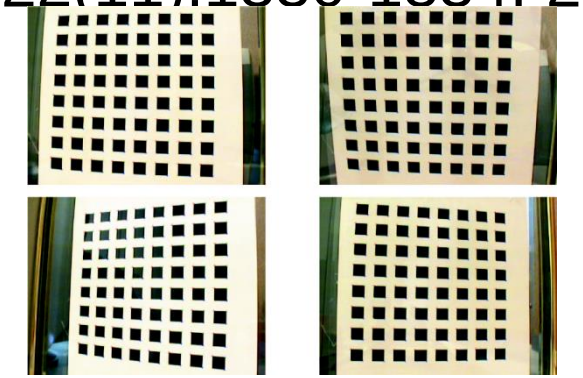


- MATLAB: http://www.vision.caltech.edu/bouguetj/calib_doc/
- OpenCV
  - Introduction: http://www.intel.com/technology/computing/opencv/
  - Source code: http://sourceforge.net/projects/opencvlibrary/

Camera Calibration

- TSAI R Y. A Versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-shelf TV cameral and lenses[J]. IEEE Journal of Robotics and Automation, 1987,RA-3(4):322-344.
- Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 22(11):1330-1334. 2000.





Camera Calibration

- Problem Statement
  - **Given:** *n* correspondences $x_i \leftrightarrow X_i$, where X_i is a scene point and x_i its image:
  - **Compute:** $P = K\,[R|\,\mathbf{t}]$ such that $x_i = PX_i$.
- The algorithm for camera calibration has two parts:
  - Compute the matrix P from a set of point correspondences.
  - Decompose P into K, R and **t** via the QR decomposition.

Camera Calibration – Compute Matrix P

Each correspondence generates two equations

$$\mathrm{x}_i = \mathrm{P}\mathrm{X}_i$$

$$x_i = \frac{p_{11}\mathrm{X}_i + p_{12}\mathrm{Y}_i + p_{13}\mathrm{Z}_i + p_{14}}{p_{31}\mathrm{X}_i + p_{32}\mathrm{Y}_i + p_{33}\mathrm{Z}_i + p_{34}} \qquad y_i = \frac{p_{21}\mathrm{X}_i + p_{22}\mathrm{Y}_i + p_{23}\mathrm{Z}_i + p_{24}}{p_{31}\mathrm{X}_i + p_{32}\mathrm{Y}_i + p_{33}\mathrm{Z}_i + p_{34}}$$

Multiplying out gives equations linear in the matrix elements of P

$$x_i(p_{31}\mathrm{X}_i + p_{32}\mathrm{Y}_i + p_{33}\mathrm{Z}_i + p_{34}) = p_{11}\mathrm{X}_i + p_{12}\mathrm{Y}_i + p_{13}\mathrm{Z}_i + p_{14}$$

$$y_i(p_{31}\mathrm{X}_i + p_{32}\mathrm{Y}_i + p_{33}\mathrm{Z}_i + p_{34}) = p_{21}\mathrm{X}_i + p_{22}\mathrm{Y}_i + p_{23}\mathrm{Z}_i + p_{24}$$

These equations can be rearranged as

$$\begin{pmatrix} \mathrm{X} & \mathrm{Y} & \mathrm{Z} & 1 & 0 & 0 & 0 & 0 & -x\mathrm{X} & -x\mathrm{Y} & -x\mathrm{Z} & -x \\ 0 & 0 & 0 & 0 & \mathrm{X} & \mathrm{Y} & \mathrm{Z} & 1 & -y\mathrm{X} & -y\mathrm{Y} & -y\mathrm{Z} & -y \end{pmatrix} \mathrm{p} = 0$$

With $\mathrm{p} = (p_{11}, p_{12}, p_{13}, p_{14}, p_{21}, p_{22}, p_{23}, p_{24}, p_{31}, p_{32}, p_{33}, p_{34})^{\tau}$ a 12-vector

Camera Calibration – Compute Matrix P

Solve **P**

- Concatenate the equations from ($n \geq 6$) correspondences to generate $2n$ simultaneous equations, which can be written:

  A**p** = **0**, where A is a $2n \times 12$ matrix.
- In general this will not have an exact solution, but a (linear) solution which minimizes |A**p**|, subject to |**p**| = 1 is obtained from the eigenvector with least eigenvalue of AA. Or equivalently from the vector corresponding to the smallest singular value of the SVD of A.
- This linear solution is then used as the starting point for a non-linear minimization of the difference between the measured and projected point:

$$\min_{\mathrm{p}} \sum_{i} ((x_i, y_i) - P(X_i, Y_i, Z_i, 1))^2$$

Decompose P into K, R and t

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R}\,|\,\mathbf{t}\right] = \left[\mathbf{KR}\,|\,\mathbf{K}\,\mathbf{t}\right]$$

The first 3 × 3 submatrix, M, of P is the product (M = KR) of an upper triangular and rotation matrix.

(1) Factor M into KR using the QR matrix decomposition. This determines K and R.

(2) Then

$$\mathrm{t} = \mathbf{K}^{-1}(p_{14}, p_{24}, p_{34})^{\tau}$$

Note, this produces a matrix with an extra skew parameter *s*

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

with $s = \tan\theta$, and θ the angle between the image axes.

Decompose P into K, R and t

- QR decomposition
  - Any matrix X could be decomposed to X=QR or RQ, where R is an upper triangular matrix, and Q is orthogonal matrix
  - Rewrite P as $P=K\begin{bmatrix}R & t\end{bmatrix}=\begin{bmatrix}KR & Kt\end{bmatrix}=\begin{bmatrix}M & Kt\end{bmatrix}$
  - M is decomposed to KR
- Calculate directly
  - Calculate K, R by

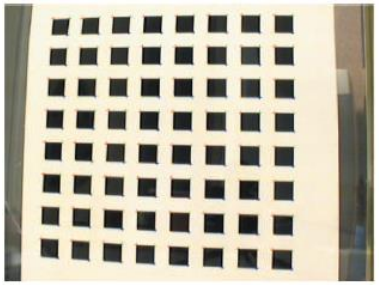
    $M=KR \Rightarrow MM^T=KRR^TK^T=KK^T \Rightarrow K$

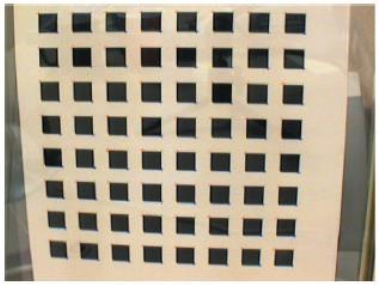
    $R=K^{-1}M$
  - Set R to be orthogonal matrix

Zhengyou Zhang's method

Basic idea

- Observe a **planar** pattern shown at a few (at least two) different orientations to calibrate a camera
- From homography to obtain constraints





Features

- All the features are on a plane
- No knowledge of 3D geometry
- No knowledge of camera motion
- Radial lens distortion is modeled
- Flexible

Camera Model

$$s\tilde{m} = A\begin{bmatrix} R & \mathrm{t} \end{bmatrix}\tilde{M} \qquad (1)$$

$\tilde{m} = [u, \mathrm{v}, 1]^T$ is 2D point in an image

$\tilde{M} = [x, y, z, 1]^T$ is 3D point in space

s is an arbitrary scale factor

(R, t) is rotation and translation, extrinsic parameters,

$$A = \begin{pmatrix} \alpha & \lambda & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$ intrinsic matrix

Homography

Suppose the Z-axis is the normal of a plane, thus we have

$$\tilde{M} = [X, Y, 1]^T$$

Therefore, a model point M and its image m is related by a homography H:

$$s\tilde{m} = H\tilde{M} \text{ with } H = A[r_1 \quad r_2 \quad t] \qquad (2)$$

the 3x3 matrix H is defined up to a scale factor.

Constraints on the intrinsic parameters

From (2), we have $[h_1 \quad h_2 \quad h_3] = \lambda A [r_1 \quad r_2 \quad t]$

Where λ is an arbitrary scalar.

Using the knowledge that r_1 and r_2 are orthonormal, we have

$$h_1^T A^{-T} A^{-1} h_2 = 0 \qquad (3)$$

$$h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 \qquad (4)$$

These are the **two basic constraints** on the intrinsic parameters, given one homography. Because a homography has 8 degrees of freedom and there are 6 extrinsic parameters (3 for rotation and 3 for translation), we can only obtain 2 constraints on the intrinsic parameters.

Solving Camera Calibration

Let $B = A^{-T}A^{-1} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{pmatrix}$

$$= \begin{pmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta} - \frac{v_0}{\beta} \\ \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta} - \frac{v_0}{\beta} & \frac{(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} + \frac{v_0^{\,0}}{\beta_0} + 1 \end{pmatrix}$$

Note that B is symmetric, defined by a 6D vector

$$b = [B_{11} \quad B_{12} \quad B_{22} \quad B_{13} \quad B_{23} \quad B_{33}]^T$$

We then have $h_i^T B h_j = v_{ij}^T b$ with

$$v_{ij} = [h_{i1}h_{j1} \quad h_{i1}h_{j2}h_{i2}h_{j1} \quad h_{i2}h_{j2} \quad h_{i3}h_{j1} + h_{i1}h_{j3} \quad h_{i3}h_{j2} + h_{i2}h_{j3} \quad h_{i3}h_{j3}]^T$$

Solving Camera Calibration

Therefore, the two fundamental constraints (3) and (4), from a given homography, can be rewritten as 2 homogeneous equations in b:

$$\begin{bmatrix} v_{12}^{T} \\ (v_{11} - v_{12})^{T} \end{bmatrix} b = 0 \qquad (8)$$

If *n* images of the model plane are observed, by stacking *n* such equations as (8) we have

$$Vb = 0 \qquad (9)$$

Steps of Solving Camera Calibration

- Estimate H from feature matching
- Calculate V by H
- Estimate b by Vb=0
- Calculate A by $B = \lambda A^{-T}A^{-1}$
- Calculate R and t by A

$$v_0 = \left(B_{12}B_{13} - B_{11}B_{23}\right) \Big/ \left(B_{11}B_{22} - B_{12}^{\ 2}\right)$$

$$\lambda = B_{33} - \left[B_{13}^{\ 2} + v_0\left(\left(B_{12}B_{13} - B_{11}B_{23}\right)\right)\right] \Big/ B_{11}$$

$$\alpha = \sqrt{\lambda / B_{11}}$$

$$\beta = \sqrt{\lambda B_{11} \Big/ \left(B_{11}B_{22} - B_{12}^{\ 2}\right)}$$

$$\gamma = -B_{12}\alpha^2\beta/\lambda$$

$$u_0 = \gamma v_0/\beta - \beta_{13}\alpha^2/\lambda$$

$r_1 = \lambda A^{-1}h_1$

$r_2 = \lambda A^{-1}h_2$

$r_3 = r_1 \times r_2$

$t = \lambda A^{-1}h_3$

with

$$\lambda = \left\| A^{-1}\boldsymbol{h}_1 \right\|^{-1} = \left\| A^{-1}\boldsymbol{h}_2 \right\|^{-1}$$

Maximum likelihood estimation

The maximum likelihood estimate can be obtained by minimizing the following functional:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\mathbf{m}_{ij}-\tilde{\mathbf{m}}(\mathbf{A}_i,\mathbf{R}_i,\mathbf{t}_i,\mathbf{M}_j)\right\|^2 \quad (10)$$

- $\hat{m}(A;R_i;t_i;M_j)$ is the projection of point M_j in image i, according to equation (2).
- R is parameterized by a vector of 3 parameters, denoted by r, which is parallel to the rotation axis and whose magnitude is equal to the rotation angle
- Minimizing (10) is a nonlinear minimization problem, which is solved with the Levenberg-Marquardt Algorithm as implemented in Minpack [18].
- It requires an initial guess of $A; fR_i; t_i \mid i = 1, \ldots, n,$ which can be obtained using the technique described in the previous subsection.

Dealing with radial distortion

Note

- (*u,v*) be the ideal (nonobservable distortion-free) pixel image coordinates
- (˘*u; ˘v*) the corresponding real observed image coordinates.
- (*x, y*) are the ideal (distortion-free) normalized image coordinates
- (˘*x, ˘y*) real (distorted) normalized image coordinates.

$$\tilde{x} = x + x\left[k_1(x^2+y^2) + k_2\left(x^2+y^2\right)^2\right]$$
$$\tilde{y} = y + y\left[k_1(x^2+y^2) + k_2\left(x^2+y^2\right)^2\right] \quad (11)$$

$$\tilde{u} = u + (u-u_0)\left[k_1(x^2+y^2) + k_2\left(x^2+y^2\right)^2\right]$$
$$\tilde{v} = v + (v-v_0)\left[k_1(x^2+y^2) + k_2\left(x^2+y^2\right)^2\right] \quad (12)$$

Coefficients k_1 and k_2 are unknown

Dealing with radial distortion

From (11) and (12), we have two equations for each point in each image:

$$\begin{pmatrix} (u-u_0)(x^2+y^2) & (u-u_0)\left(x^2+y^2\right)^2 \\ (v-v_0)\left(x^2+y^2\right) & (v-v_0)\left(x^2+y^2\right)^2 \end{pmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \tilde{u}-u \\ \tilde{v}-v \end{bmatrix}$$

Given *m* points in *n* images, we can stack all equations together to obtain in total $2mn$ equations, or in matrix form as $\mathrm{D}\mathbf{k} = \mathbf{d}$, where $\mathbf{k} = [k_1; k_2]T$. The linear least-squares solution is given by

$$\mathbf{k} = \left(\mathrm{D}^{\mathrm{T}}\mathrm{D}\right)^{-1}\mathrm{D}^{\mathrm{T}}\mathbf{d} \qquad (13)$$

Once k_1 and k_2 are estimated, one can refine the estimate of the other parameters by solving (10) with $\hat{}\mathrm{m}(\mathrm{A}; \mathrm{R}_i; \mathrm{t}_i; \mathrm{M}_j)$ replaced by (11) and (12). We can alternate these two procedures until convergence.

Complete Maximum Likelihood Estimation

Estimate the complete set of parameters by minimizing the following functional:

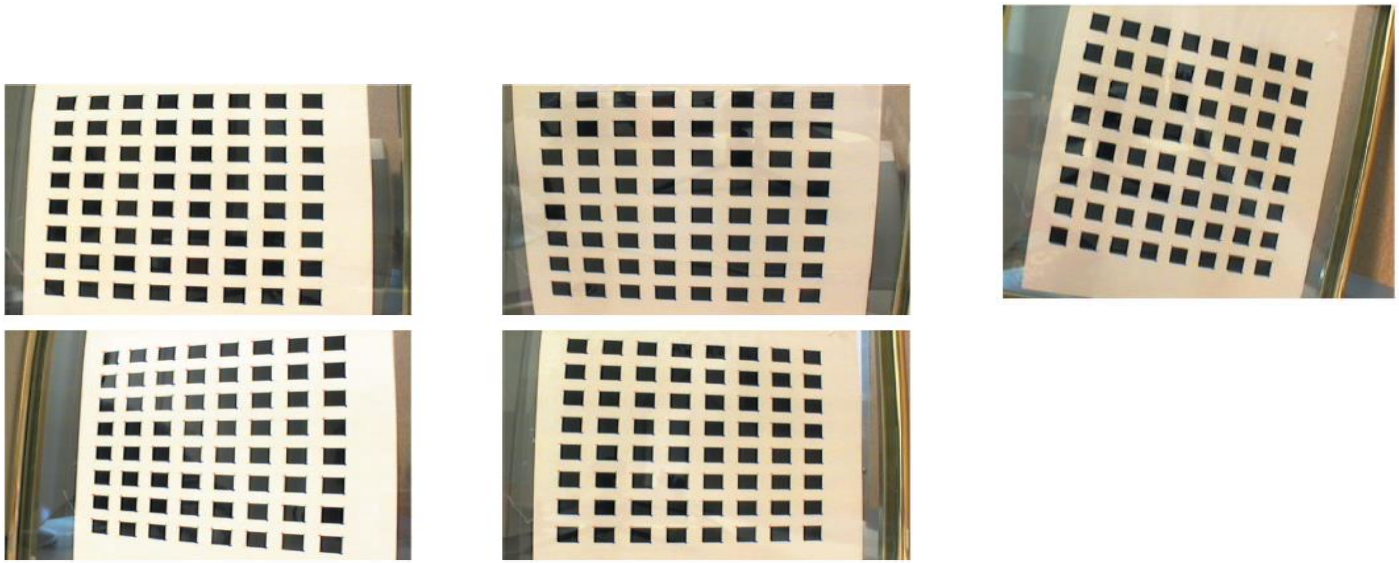
$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\mathbf{m}_{ij}-\breve{\mathbf{m}}(\mathbf{A}_i,\mathbf{k}_1,\mathbf{k}_2,\mathbf{R}_i,\mathbf{t}_i,\mathbf{M}_j)\right\|^2 \qquad (14)$$

- This is a nonlinear minimization problem, LM
- The initial values of (A, R_i, t_i) and (k_1,k_2) are obtained in two steps

Summery

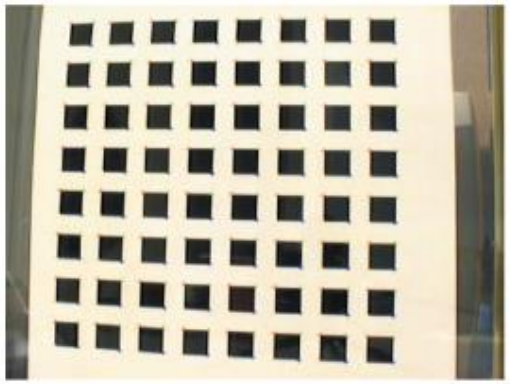
- Print a pattern and attach it to a planar surface;
- Take a few images of the model plane under different orientations by moving either the plane or the camera;
- Detect the feature points in the images;
- Estimate the five intrinsic parameters and all the extrinsic parameters using the closed-form solution;
- Estimate the coefficients of the radial distortion by solving the linear least-squares (13);
- Refine all parameters by minimizing (14).

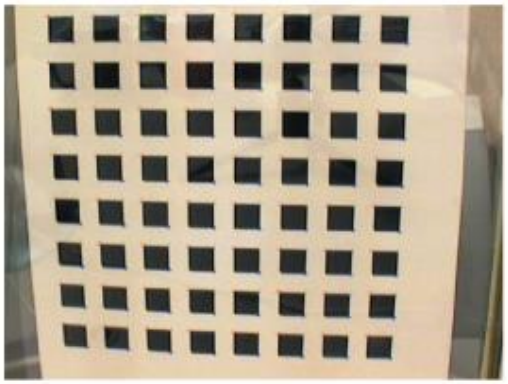
Example



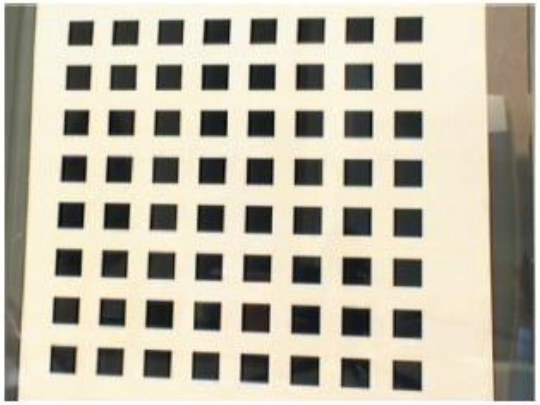
Five images of a model plane, together with the extracted corners (indicated by cross)

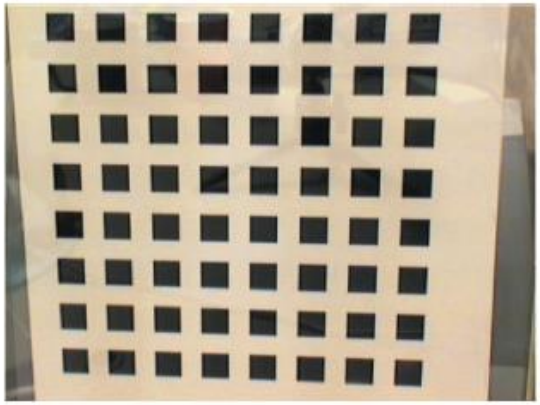
Example





Original images





after having corrected radial distortion

Implementation

- All the real data and results are available from the following Web page: http://research.microsoft.com/~zhang/Calib/
- Optimization:
  - Levenberg-Marquardt Algorithm in C++
    - http://www.ics.forth.gr/~lourakis/levmar/
  - Matlab
- OpenCV includes this function
- Camera Calibration Toolbox for Matlab
  - http://www.vision.caltech.edu/bouguetj/calib_doc/

Multi-view Geometry

- **Structure-from-Motion**
  - **Automatically recover the camera parameters and 3D structure from multiple images or video sequences.**

Two-View Geometry

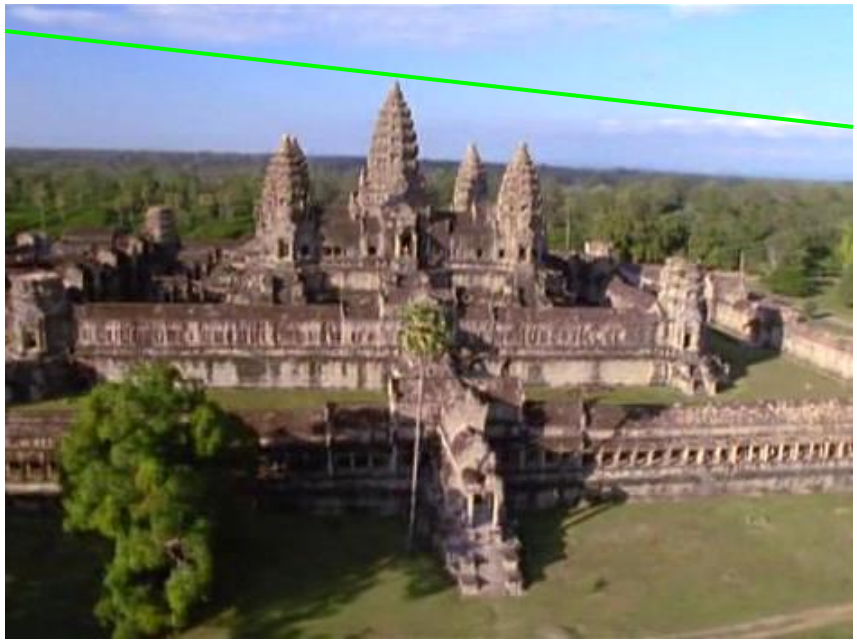
3D???

Two-View Geometry

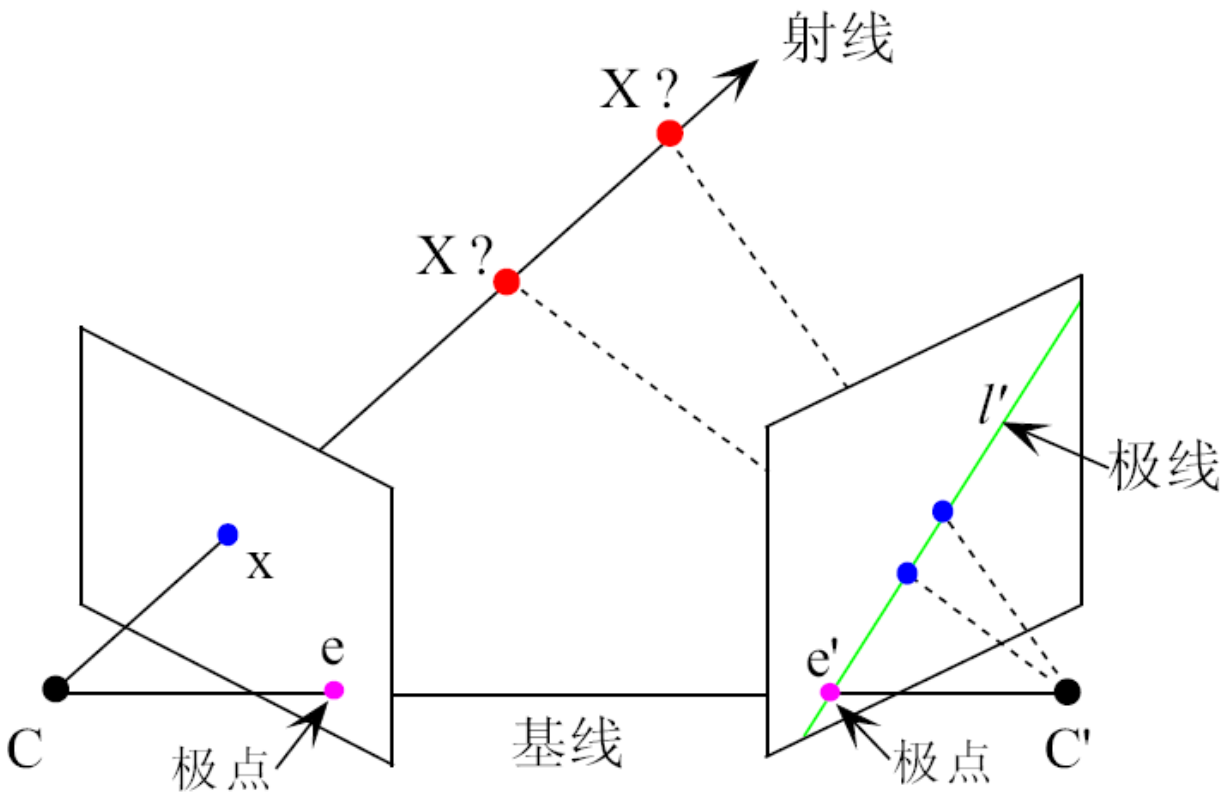
3D???

Two-View Geometry

3D: Epipolar Geometry



Epipolar Geometry



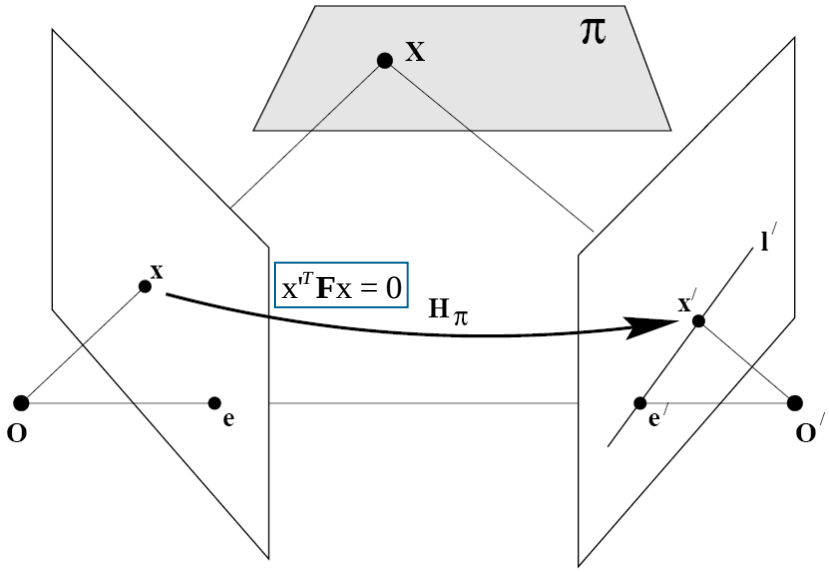
$$\hat{\mathbf{x}}'^{\top} F \hat{\mathbf{x}} = 0$$

Features of Fundamental Matrix

- Depends only on the relative pose and internal parameters

$$F = K_2^{-T}[t]_\times R K_1^{-1}$$

- F is a 3×3 rank 2 homogeneous matrix
- F**e** = **0**
- It has 7 degrees of freedom
- Compute from 7 image point correspondences



Basic equations

Given a correspondence

$$\mathbf{x} \leftrightarrow \mathbf{x}'$$

The basic incidence relation is

$$\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$$

May be written

$$x'x f_{11} + x'y f_{12} + x' f_{13} + y'x f_{21} + y'y f_{22} + y' f_{23} + x f_{31} + y f_{32} + f_{33} = 0 \ .$$

Single point equation - Fundamental matrix

Gives an equation :

$$(x'x, x'y, x', y'x, y'y, y', x, y, 1) \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

where

$$\mathbf{f} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})^\top$$

holds the entries of the Fundamental matrix

Total set of equations

$$\mathtt{Af} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = \mathbf{0}$$

Random Sample Consensus (RANSAC)

[Fischler and Bolles, 1981]

Objective Robust fit of a model to a data set S which contains outliers.

Step 1. Compute a set of potential matches

Step 2. While T(#inliers, #samples) < 95% do

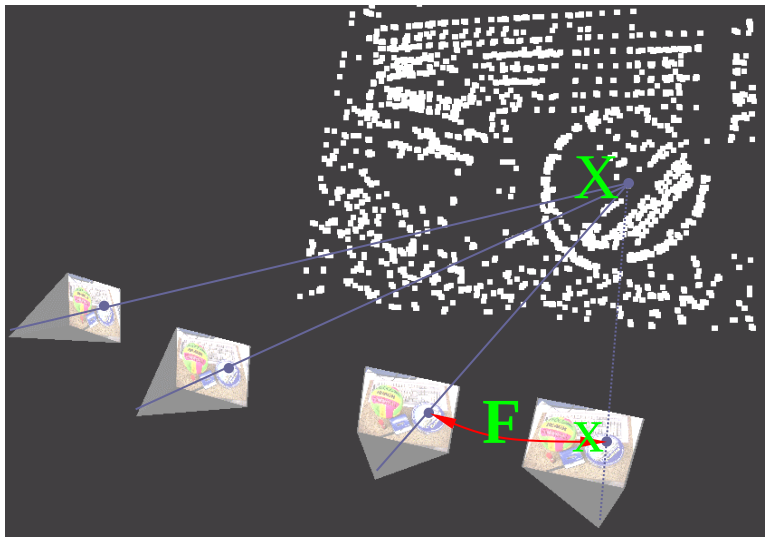
step 2.1 select minimal sample (8 matches)

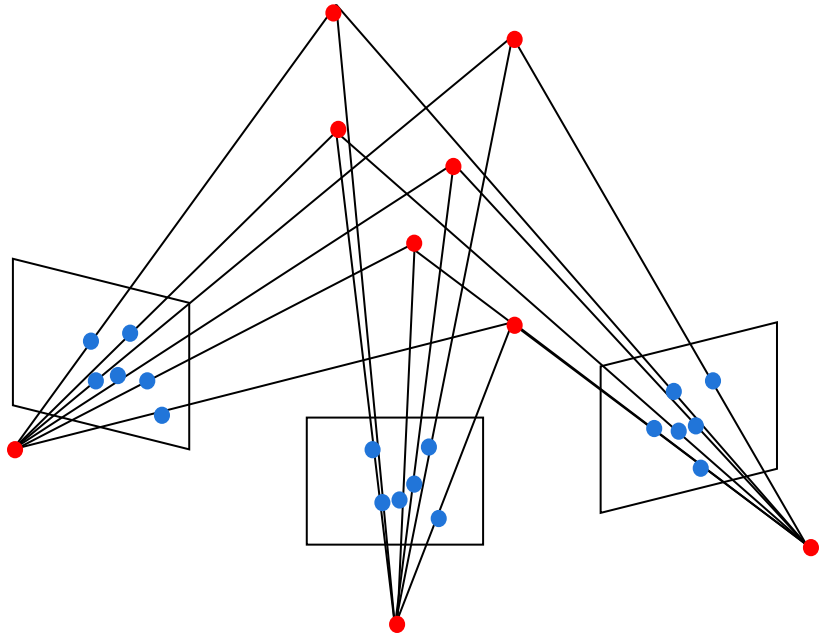
step 2.2 compute solutions for *F*

step 2.3 determine inliers

Step 3. Refine *F* based on all inliers

Multi-view Geometry





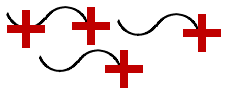
$$\mathbf{x}_{ij} = \pi(\mathbf{P}_i X_j)$$

Projection Function $\pi(x, y, z) = (x/z, y/z) \quad \mathbf{P}_i = \mathbf{K}_i[\mathbf{R}_i|\mathbf{T}_i]$

Automatic Camera Tracking

- Pipeline
  - Feature Tracking
    - Obtain a set of feature tracks

$$\mathcal{X} = \{\mathbf{x}_i|_{i=1,\ldots,m}\}$$



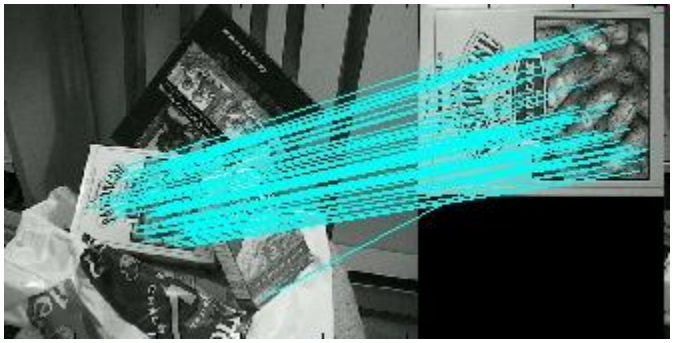
  - Structure from Motion
    - Solve the camera parameters and 3D points of tracks

$$\mathbf{x}_{ij} = \pi(\mathbf{P}_i X_j) \qquad \mathbf{P}_i = \mathbf{K}_i[\mathbf{R}_i|\mathbf{T}_i]$$

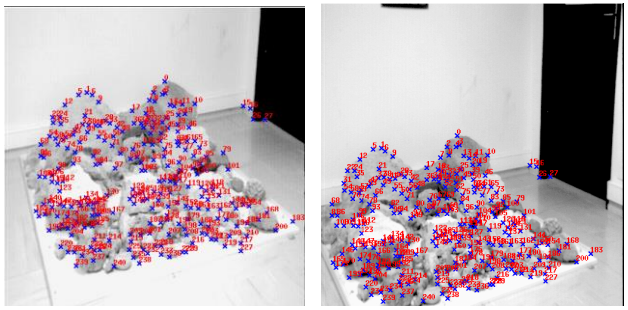
$$E(\mathbf{P}_1, \ldots, \mathbf{P}_m, X_1, \ldots, X_n) = \sum_{i=1}^{m}\sum_{j}^{n} w_{ij}||\pi(\mathbf{P}_i X_j) - \mathbf{x}_{ij}||^2$$

Feature Tracking

- SIFT:
  http://www.cs.ubc.ca/~lowe/keypoints/
  - Scale-Invariant Features Transform
  - Feature describer

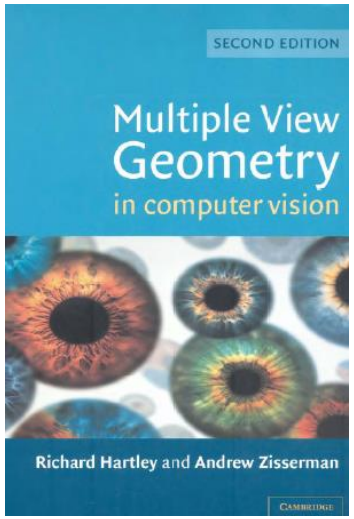


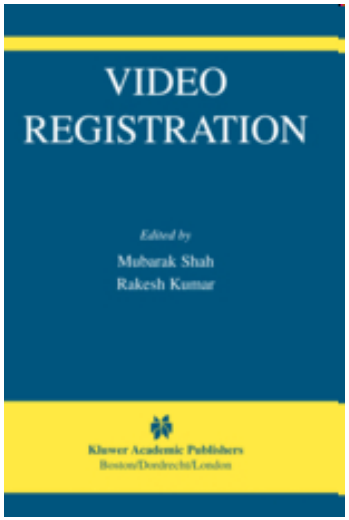
- KLT:
  http://www.ces.clemson.edu/~stb/klt/
  - Kanade-Lucas-Tomasi
  - Feature Tracker



Some Literatures

- Books
  - Multiple View Geometry in Computer Vision. Second Edition. Richard Hartley and Andrew Zisserman, Cambridge University Press, March 2004.
  - A. Fitzgibbon and A. Zisserman. Automatic camera tracking. In M. Shah and R. Kumar, editors, Video Registration, chapter 2, pages 18-35. Kluwer, 2003.





Some Literatures

- **Papers**
  - M. Pollefeys, L. J. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. International Journal of Computer Vision, 59(3):207-232, 2004.
  - Noah Snavely, Steven M. Seitz, Richard Szeliski. Modeling the World from Internet Photo Collections. International Journal of Computer Vision, 2007.
  - Guofeng Zhang, Xueying Qin, Wei Hua, Tien-Tsin Wong, Pheng-Ann Heng, Hujun Bao. Robust Metric Reconstruction from Challenging Video Sequences. CVPR 2007.

Some State-of-the-Art Softwares

- Commercial Softwares
  - Boujou
  - SynthEyes Camera tracker
  - REALVIZ MatchMover
- Free Softwares
  - VisualSFM
    - http://ccwu.me/vsfm/
  - ACTS
    - http://www.zjucvg.net/acts/acts.html
  - Bundler
    - http://phototour.cs.washington.edu/bundler/

VisualSFM : A Visual Structure from Motion System

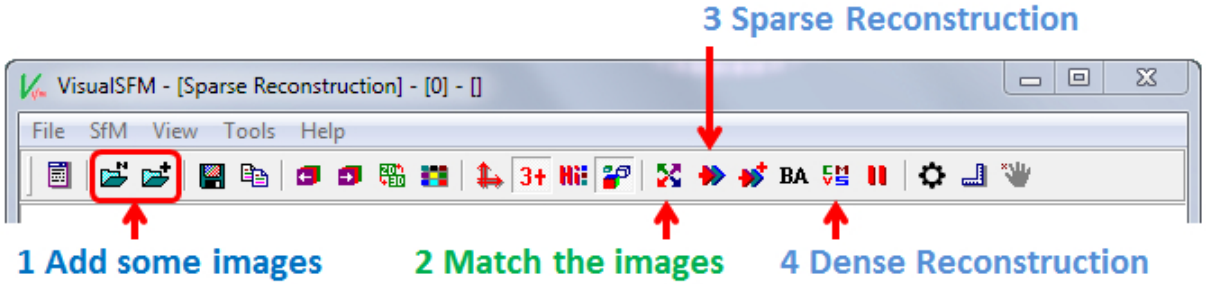
Changchang Wu

VisualSFM is a GUI application for 3D reconstruction using structure from motion (SFM). The reconstruction system integrates several of my previous projects: SIFT on GPU(SiftGPU), Multicore Bundle Adjustment, and Towards Linear-time Incremental Structure from Motion. VisualSFM runs fast by exploiting multicore parallelism for feature detection, feature matching, and bundle adjustment.

For dense reconstruction, this program integrates the execution of Yasutaka Furukawa's PMVS/CMVS tool chain. The SfM output of VisualSFM works with several additional tools, including CMP-MVS by Michal Jancosek, MVE by Michael Goesele's research group, SURE by Mathias Rothermel and Konrad Wenzel, and MeshRecon by Zhuoliang Kang.

Structure from Motion - A Visual Interface

Reconstruct 3D with a few button clicks, and watch the dynamic reconstruction process!



You still have the option to run from command line without a GUI!

```
>VisualSFM sfm+pmvs ./images ./result.nvm
```

Download V0.5.26 (changelog with new feature documentation)

Windows* (64-bit, 32-bit, installation guide), *for nVidia CUDA or CUDA Simulation.
Windows (64-bit, 32-bit, installation guide)
Linux (64-bit, 32-bit, installation guide), see the tutorials for Ubuntu or Fedora.
Mac OSX (64-bit, 32-bit, installation guide), see the installer by Dan Monaghan.

* VisualSFM is free for personal, non-profit or academic use. See README for more details.
* Please cite VisualSFM according to README in your publication.

ACTS

ACTS: Automatic Camera Tracking System

Home

Description

ACTS is an automatic camera tracking system which can efficiently and stably recover camera motion as well as 3D scene strucutre from videos and film sequences. **It is for non-commercial research and educational use ONLY. Not for reproduction, distribution or commercial use.** If you use this executable for your academic publication, please acknowledge our work. This program is tested on Windows XP, Server 2003, but is still not guaranteed to be bug-free and work properly with all versions of Windows. You are welcome to report any suggestions or bugs. We will actively update the program. Please email Guofeng Zhang if you have any questions.

Structure-from-Motion is a cornerstone for many other computer vision tasks, such as 3D reconstruction, video-based rendering and video editing. In the future, we will actively update ACTS and **add more and more advanced or extra functions along with our published papers**, such as non-consecutive feature tracking, video stabilization, stereoscopic video synthesis, and depth video recovery.

Demo of ACTS

Application：Augmented Reality

Bundler

http://phototour.cs.washington.edu/bundler/

- Can recover 3D points as well as camera position and orientation from a set of unordered image set.

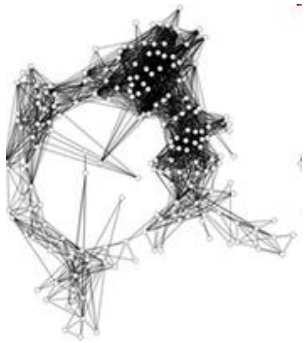
Photo Tourism

Noah Snavely, Steven M. Seitz, Richard Szeliski. Photo Tourism: Exploring image collections in 3D. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006), 2006.

Skeletal Sets for Efficient Structure from Motion

- How to solve large, unordered, highly redundant, and irregularly sampled photo collections?

A few sample photos from a collection of Flickr images of Stonehenge.



An image graph for this photo collection

The computed skeletal graph.

A view of the complete reconstruction.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. Skeletal Sets for Efficient Structure from Motion. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2008.

Building Rome in a Day

Optimize the SFM pipeline by Parallel Technique

150,000	496	13 Hours	8 Hours	2,106

Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, Richard Szeliski. Building Rome in a day. ICCV 2009: 72-79.

Automatic Camera Tracking Framework

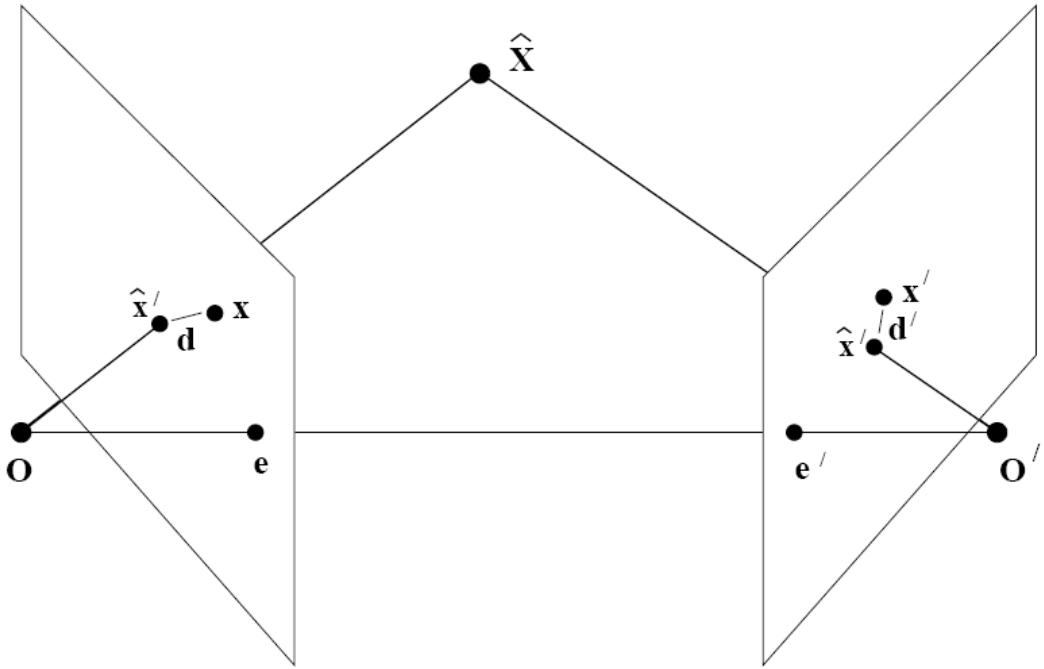
- Feature tracking over whole sequence
- Structure & motion initialization
  - Compute F between two initial images
  - Compute P_1 and P_2
  - Triangulate 3D points of the matched features
- For each additional view
  - Compute the camera pose
  - Refine and extend 3D points
- Self-Calibration
  - Upgrade the projective reconstruction to metric one.
- Refine structure and motion
  - Bundle adjustment

Triangulation

- Knowing *F*, Compute *P* and *P'*

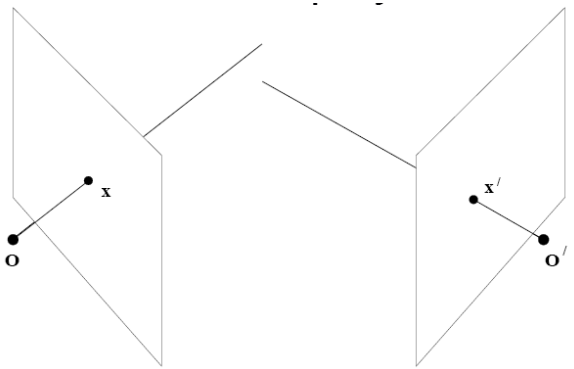
$$\mathrm{P} = [\mathrm{I} \mid 0] \quad ; \quad \mathrm{P}' = [[\mathrm{e}']_{\times}\mathrm{F} \mid \mathrm{e}'] = [\mathrm{M} \mid \mathrm{e}']$$

- Knowing **x** and **x'**
- Compute **X** such that $\mathrm{x}=\mathbf{P}\mathrm{X}$ $\mathrm{x}'=\mathbf{P}'\mathrm{X}$

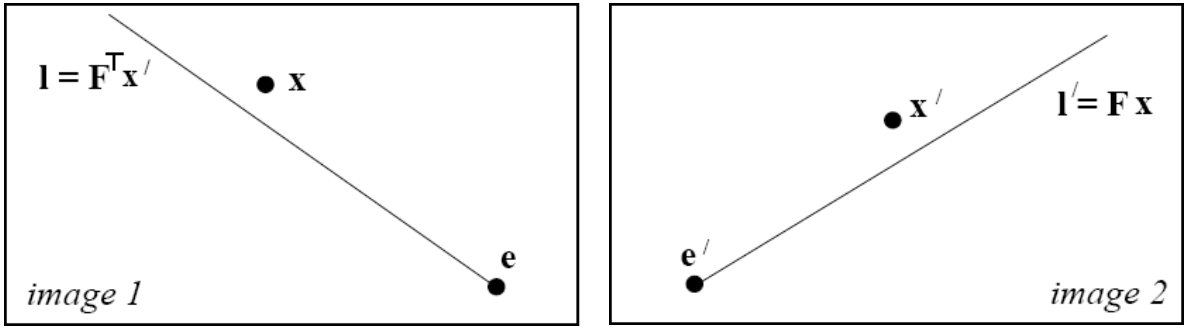


Triangulation in presence of noise

- In the presence of noise, back-projected lines do not intersect.



Rays do not intersect in space



Measured points do not lie on corresponding epipolar lines

Linear triangulation methods

- Given equations

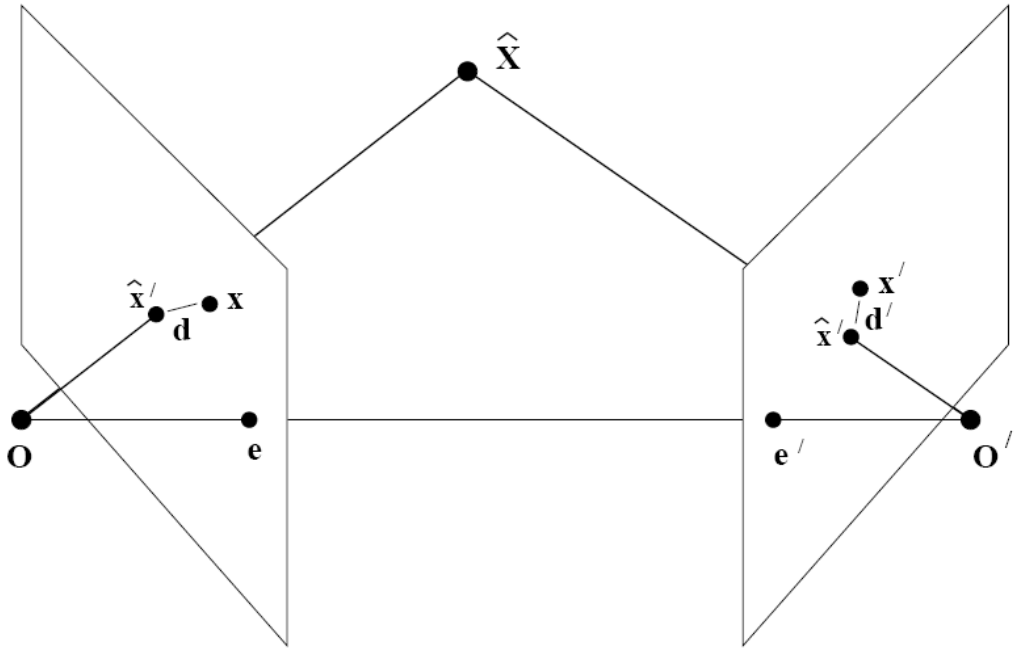
$$\mathbf{x} = \mathrm{P}\mathbf{x}$$
$$\mathbf{x}' = \mathrm{P}'\mathbf{x}$$

  - $\mathbf{p}^{iT}$ are the rows of P.
  - Write as linear equations in **X**

$$\begin{bmatrix} x\mathbf{p}^{3\mathrm{T}} - \mathbf{p}^{1\mathrm{T}} \\ y\mathbf{p}^{3\mathrm{T}} - \mathbf{p}^{2\mathrm{T}} \\ x'\mathbf{p}'^{3\mathrm{T}} - \mathbf{p}'^{1\mathrm{T}} \\ y\mathbf{p}'^{3\mathrm{T}} - \mathbf{p}'^{2\mathrm{T}} \end{bmatrix} \mathbf{x} = 0$$

  - Solve for **X**.
  - Generalizes to point match in several images.
  - Minimizes no meaningful quantity – not optimal.

Geometric error . . .



- Cost function

$$X = \arg\min_{X} \sum_{i} ||\pi(\mathbf{P}_i X) - \mathbf{x}_i||^2$$

Knowing 3D points, Compute Camera Motion

- Compute Projection Matrix

$$\mathbf{P}_i = \arg\min_{\mathbf{P}_i} \sum_j ||\pi(\mathbf{P}_i X_j) - \mathbf{x}_{ij}||^2$$

- Decomposition for Metric Projection Matrix

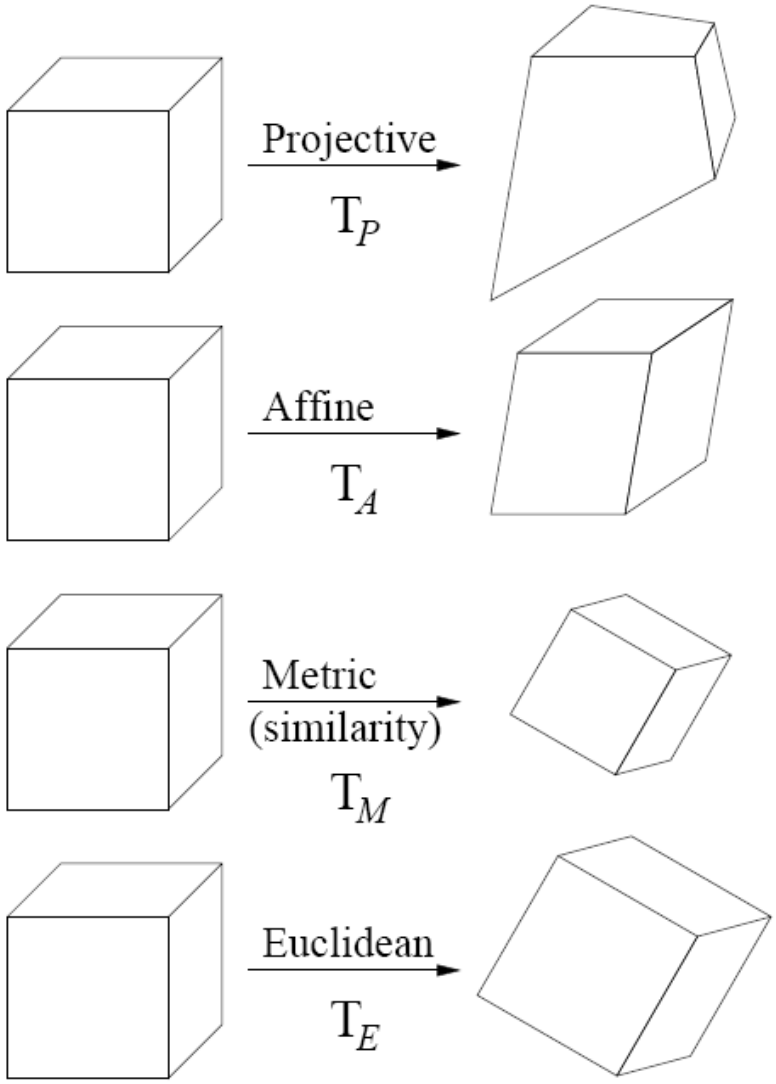
$$P = K[R\,|\,t] = [KR\,|\,Kt] = [M\,|\,Kt]$$

Decompose *M* into *K*, *R* by QR decomposition

$$t = K^{-1}(p_{14}, p_{24}, p_{34})^T$$

Geometric Ambiguities

ambiguity	DOF	transformation	invariants
projective	15	$\mathbf{T}_P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix}$	cross-ratio
affine	12	$\mathbf{T}_A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$	relative distances along direction parallelism *plane at infinity*
metric	7	$\mathbf{T}_M = \begin{bmatrix} \sigma r_{11} & \sigma r_{12} & \sigma r_{13} & t_x \\ \sigma r_{21} & \sigma r_{22} & \sigma r_{23} & t_y \\ \sigma r_{31} & \sigma r_{32} & \sigma r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$	relative distances angles *absolute conic*
Euclidean	6	$\mathbf{T}_E = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$	absolute distances



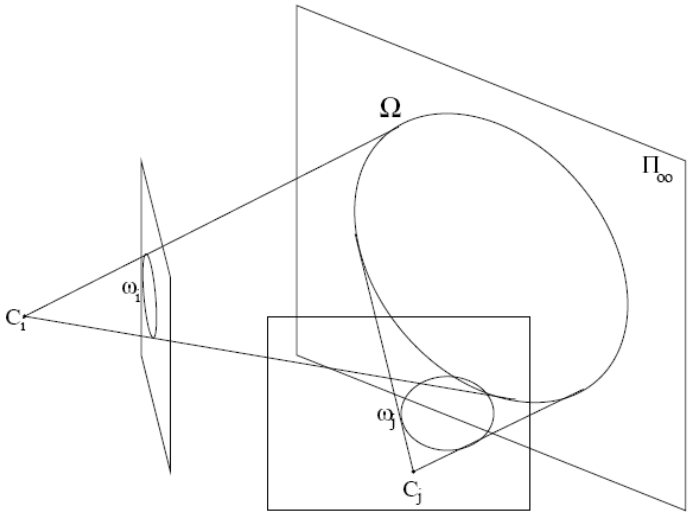
Projective Reconstruction → Self-Calibration → Metric Reconstruction

Self-Calibration Theory

- Absolute Conic (绝对二次曲线)
  - 无穷远平面 π_∞ (w=0)上的一个曲线，满足

    $$(x, y, z) I (x, y, z)^T = 0$$

  - 其上的点是无穷远平面上的虚点



Self-Calibration Theory

- 双绝对二次曲面 Ω^*
  - 在度量坐标框架下 $\Omega^* = \mathrm{diag}(1,1,1,0)$
    - 对于相似变换具有不变性
  - 在射影重建下
    - 可以表示成一个4*4秩为3的对称半正定矩阵
    - 射影空间->度量空间
      - 求解一个变换矩阵U，使得

$$U\Omega^* U^T = diag(1,1,1,0)$$

$$\mathbf{P}_M = \mathbf{P}\mathbf{U}^{-1},\ X_M = \mathbf{U}X$$

Self-Calibration Theory

- 双绝对二次曲面在图像上的投影 w^*

$$\omega^* = \mathbf{P}\Omega^*\mathbf{P}^\top$$

- 在度量坐标框架下,w^*只与摄像机的内参有关系

$$\omega^* = \mathbf{K}\mathbf{K}^\top$$

- 自定标约束方程：

$$\mathbf{K}\mathbf{K}^\top \sim \mathbf{P}\Omega^*\mathbf{P}^\top$$

Self-Calibration

- Assuming skew = 0, principal point and aspect ratio is known.
- Normalization

$$\mathbf{P}_k^N = (\mathbf{K}^N)^{-1}\mathbf{P}_k, \mathbf{K}^N = \begin{bmatrix} w+h & & w/2 \\ & w+h & h/2 \\ & & 1 \end{bmatrix}$$

- Self-calibration Equations:

$$\lambda_k \begin{bmatrix} f_k^2 & 0 & 0 \\ 0 & f_k^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{P}_k \begin{bmatrix} f_1^2 & 0 & 0 & a_1 \\ 0 & f_1^2 & 0 & a_2 \\ 0 & 0 & 1 & a_3 \\ a_1 & a_2 & a_3 & ||a||^2 \end{bmatrix} \mathbf{P}_k^\top$$

Self-Calibration

- Incorporating a prior knowledge：

$$\omega^* \sim \mathbf{K}\mathbf{K}^\top = \begin{bmatrix} f^2+s^2+u^2 & srf+uv & u \\ srf+uv & r^2f^2+v^2 & v \\ u & v & 1 \end{bmatrix} \approx \begin{bmatrix} 1\pm 9 & \pm 0.01 & \pm 0.1 \\ \pm 0.01 & 1\pm 9 & \pm 0.1 \\ \pm 0.1 & \pm 0.1 & 1 \end{bmatrix}$$

$$\frac{1}{9\nu}(P_k[1]\Omega^* P_k[1]^\top - P_k[3]\Omega^* P_k[3]^\top) = 0$$
$$\frac{1}{9\nu}(P_k[2]\Omega^* P_k[2]^\top - P_k[3]\Omega^* P_k[3]^\top) = 0$$
$$\frac{1}{0.2\nu}(P_k[1]\Omega^* P_k[1]^\top - P_k[2]\Omega^* P_k[2]^\top) = 0$$
$$\frac{1}{0.1\nu}(P_k[1]\Omega^* P_k[3]^\top) = 0$$
$$\frac{1}{0.1\nu}(P_k[2]\Omega^* P_k[3]^\top) = 0$$
$$\frac{1}{0.01\nu}(P_k[1]\Omega^* P_k[2]^\top) = 0$$

ν a scale factor that is initially set to 1 and later on to $P_3\tilde{\Omega}^* {P_3}^\top$

$\tilde{\Omega}^*$ the result of the previous iteration

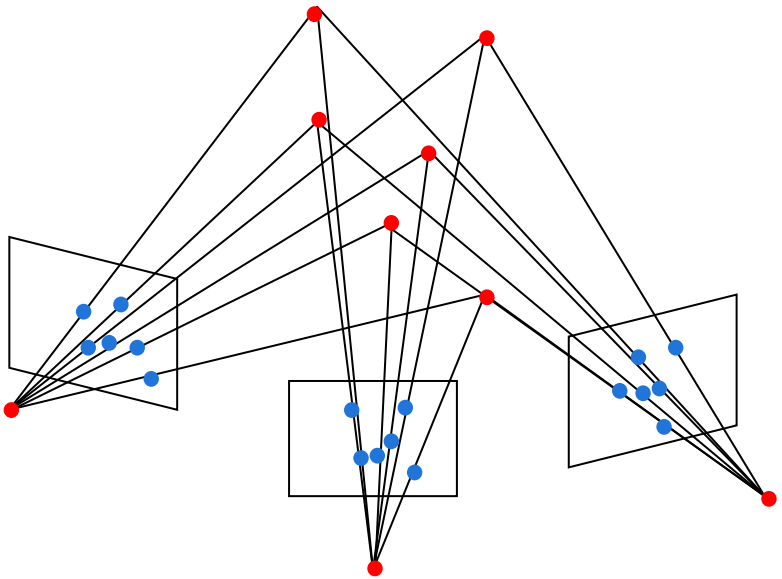
More Details about Self-Calibration

- State-of-the-Art References
  - R.I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, second ed. Cambridge Univ. Press, 2004.
  - M. Pollefeys, L.J. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, Visual Modeling with a Hand-Held Camera, Int'l J. Computer Vision, vol. 59, no. 3, pp. 207-232, 2004.
  - G. Zhang, X. Qin, W. Hua, T.-T. Wong, P.-A. Heng, and H. Bao, Robust Metric Reconstruction from Challenging Video Sequences, Proc. IEEE CS Conf. Computer Vision and Pattern Recognition, 2007.

Bundle Adjustment

- Definition
  - Refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates.

$$\arg\min_{\mathbf{P}_k, \mathrm{X}_i} \sum_{k=1}^{m} \sum_{i=1}^{n} D\left(\mathrm{x}_{ki}, \mathbf{P}_k\left(\mathrm{X}_i\right)\right)^2$$



Non-Linear Least Squares

- The sum squared error:

$$E(\mathrm{P}_1, ..., \mathrm{P}_m, X_1, ..., X_n) = \sum_{i=1}^{m}\sum_{j}^{n} w_{ij}||\pi(\mathrm{P}_i X_j) - \mathrm{x}_{ij}||^2$$

$$\text{投影函数}\pi(x, y, z) = (x/z, y/z)$$

$$f_{ij}(\Phi) = \pi(\mathrm{P}_i X_j)$$

- Linear approximation of residual error

$$f_{ij}(\Phi + \Delta) \approx f_{ij}(\Phi) + J_{ij}\Delta$$

$$r_{ij} = \pi(\mathrm{P}_i X_j) - \mathrm{x}_{ij} \approx (f_{ij}(\Phi) - \mathrm{x}_{ij}) + J_{ij}\Delta$$

$$J_{ij} = \frac{\partial f_{ij}(\Phi)}{\partial \Phi}$$

- Allows quadratic approximation of sum-of-squares: $(r - J\Delta)^T (r - J\Delta)$

Non-Linear Least Squares

- Minimization corresponds to finding zeros of derivative

$$2\mathrm{J}^{\mathrm{T}}\mathrm{J}\Delta - 2\mathrm{J}^{\mathrm{T}}r = 0$$

$$\Rightarrow \Delta = \left(\mathrm{J}^{\mathrm{T}}\mathrm{J}\right)^{-1}\mathrm{J}^{\mathrm{T}}r$$

$$\Delta = (\mathrm{J}^{\mathrm{T}}\mathrm{J} + \lambda \mathrm{diag}(\mathrm{J}^{\mathrm{T}}\mathrm{J}))^{-1}\mathrm{J}^{\mathrm{T}}r$$

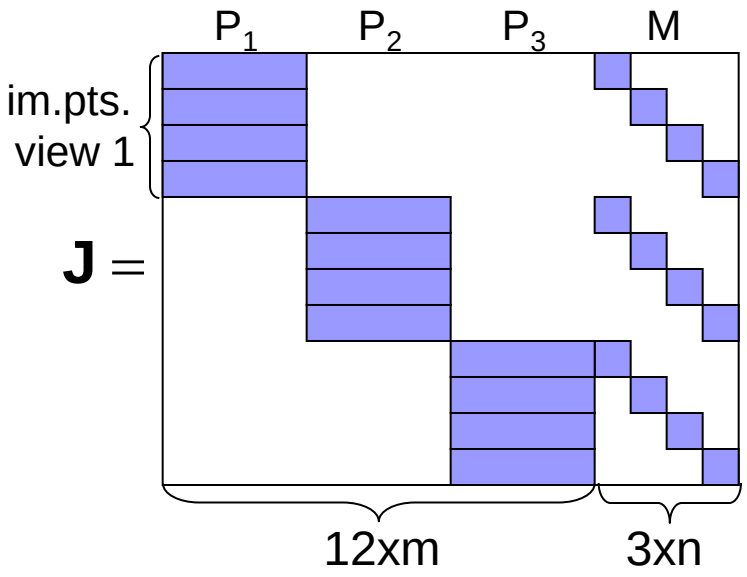
- Levenberg–Marquardt algorithm

(decrease/increase λ if success/failure to descent)

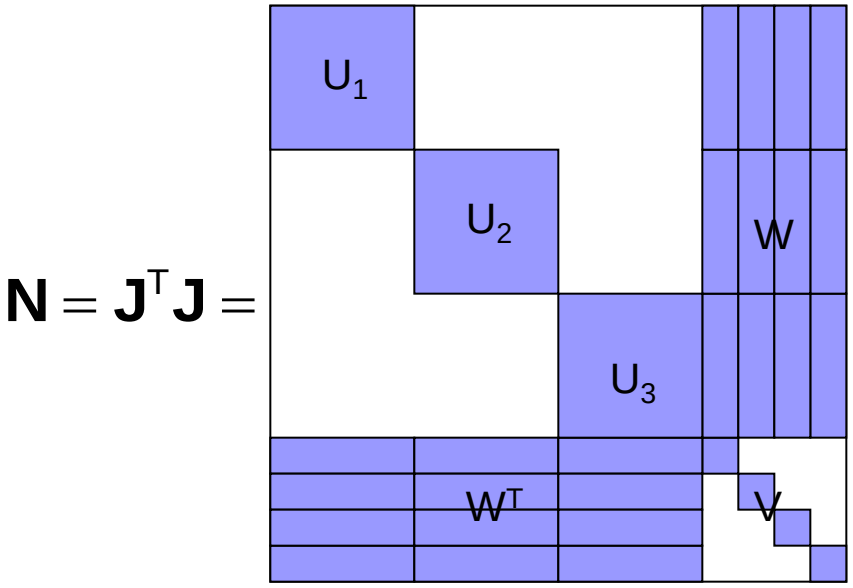
http://www.ics.forth.gr/~lourakis/levmar/

Bundle Adjustment

- Jacobian of $\sum_{i=1}^{m}\sum_{j=1}^{n} D\left(\mathrm{m}_{ij}, \hat{\mathrm{P}}_i\left(\hat{\mathrm{M}}_j\right)\right)^2$ has sparse block structure
  - Cameras independent of other cameras
  - Points independent of other points



in general much larger



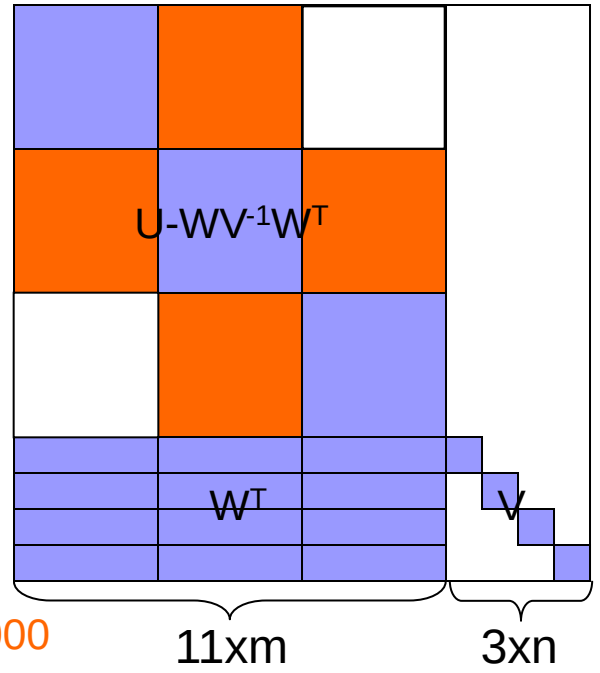
Bundle Adjustment

Eliminate dependence of camera/motion parameters on structure parameters

Note in general 3n >> 11m

http://www.ics.forth.gr/~lourakis/sba/

$$\begin{bmatrix} \mathbf{I} & -\mathbf{W}\mathbf{V}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \times \mathbf{N} =$$



Allows much more efficient computations

e.g. 100 views,10000 points,
solve ±1000x1000, not ±30000x30000

Often still band diagonal

use sparse linear algebra algorithms

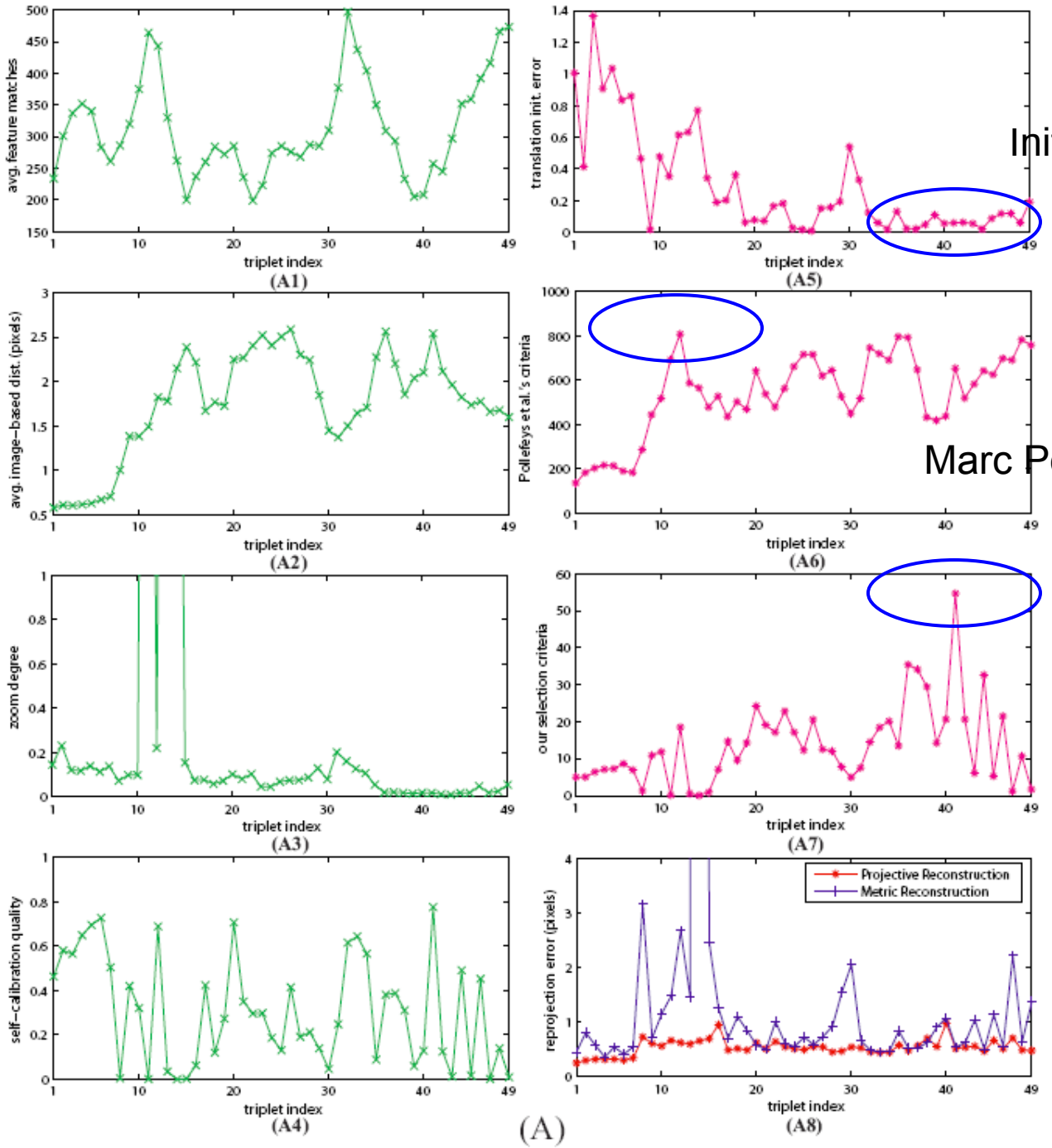
变焦序列的问题

- 自定标的不确定性
  - 摄像机的前后移动与镜头缩放容易混淆
  - 镜头缩放过程中，聚焦的场景往往接近平面，导致自定标不确定性程度更加严重
- 特征匹配误差偏大
  - 在镜头缩放下，匹配更难
  - 镜头的快速缩放，容易产生运动模糊

初始帧选取标准

- 退化程度 $b = \mathrm{median}(d(H\mathbf{u}, \mathbf{u}'))$
- 变焦程度 $\Delta f_{ij} = \frac{|f_i/f_j - 1| + |f_j/f_i - 1|}{2b_{ij}}$
- 自定标质量 $C(E_{calib}) = \frac{\varepsilon}{\varepsilon + \sqrt{E_{calib}}} e^{-\frac{E_{calib}}{2\sigma^2}}$
- 综合

$$\left.\begin{aligned} S_i &= C(E_{calib})(B_{i,i+1} + B_{i+1,i+2} + B_{i,i+2}) \\ \tilde{S}_i &= \Big(\sum_{k=i-3w}^{i+3w} e^{-\frac{(k-i)^2}{2w^2}} S_k\Big) \Big/ \sum_{k=i-3w}^{i+3w} e^{-\frac{(k-i)^2}{2w^2}} \end{aligned}\right\} \quad S_i^b = \sqrt{\tilde{S}_i S_i}$$

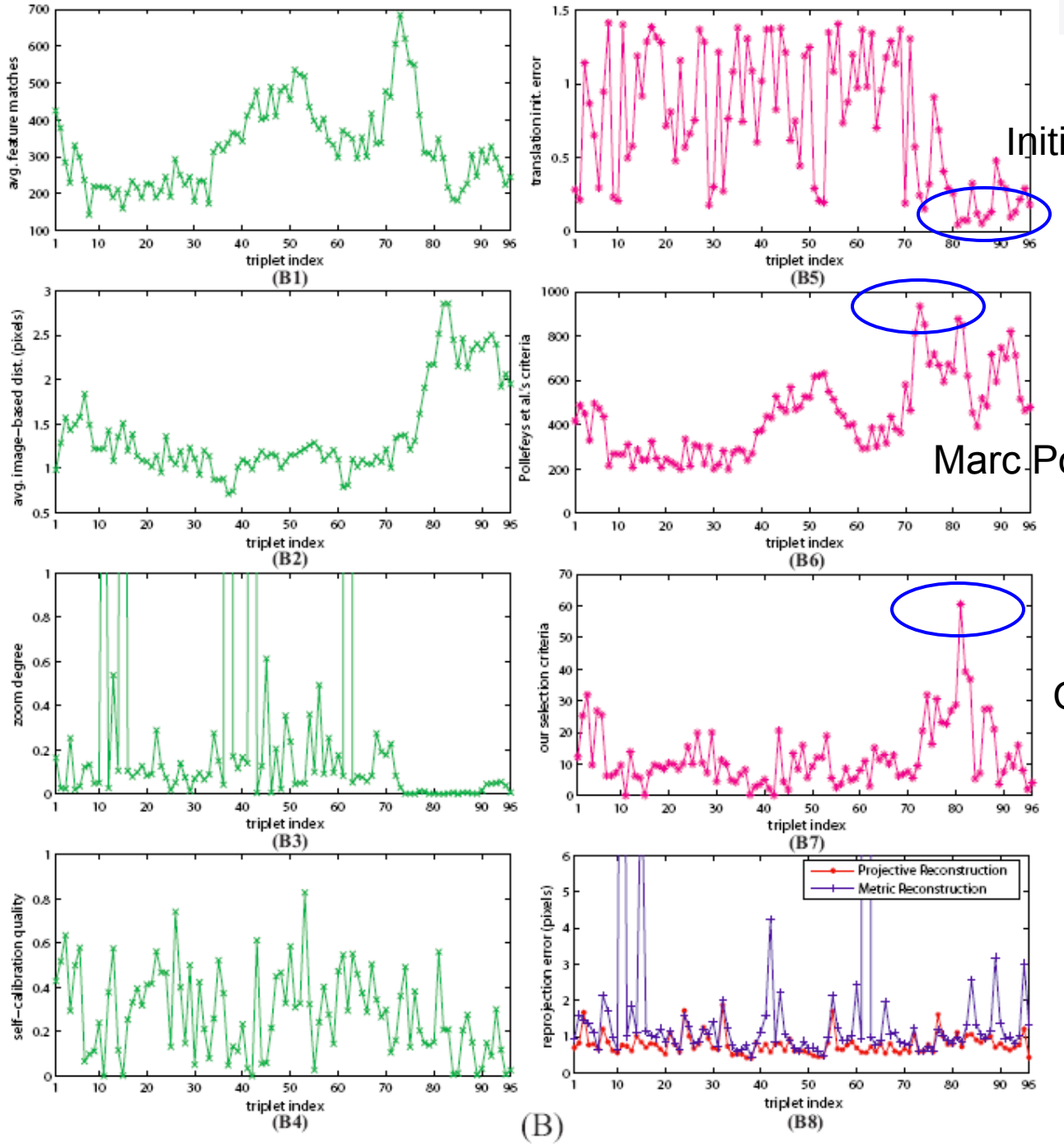


Initialization Error

Marc Polleys' method

Our method

(A)



Initialization Error

Marc Polleys' method

Our method

(B)

长序列的求解效率

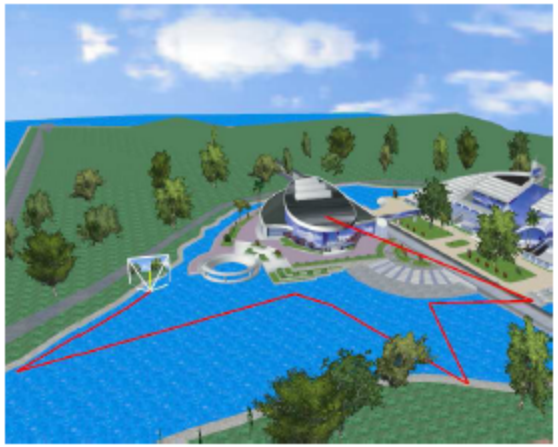
- Bundle Adjustment（BA）
  - 对精度优化而言是不可或缺的！
  - 缺点是极其耗时，占用了绝大部分计算时间！
- Local On-Demand策略
  - 将BA局部化，限制在最需要优化的部分；
  - Incremental方式求解的过程中，绝大多数只要局部的BA就可以了。

长序列的求解效率

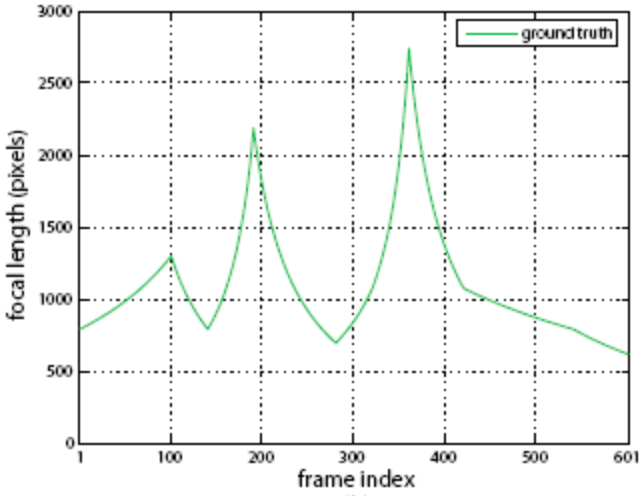
我们方法的结果以及与商业软件“Boujou three”的对比

sequence	Syn. Campus	Building	Garden
frames	601	2410	1608
key frames	51	98	55
3D points	2825	2908	4283
image projections	171,342	462,621	803,750
RMSE. (pixels)	0.586	1.327	1.065
matching time (min.)	7	28	21
solving time (min.)	6	16	12
performance list of boujou three			
matching time (min.)	6	21	14
solving time (min.)	55	51	34

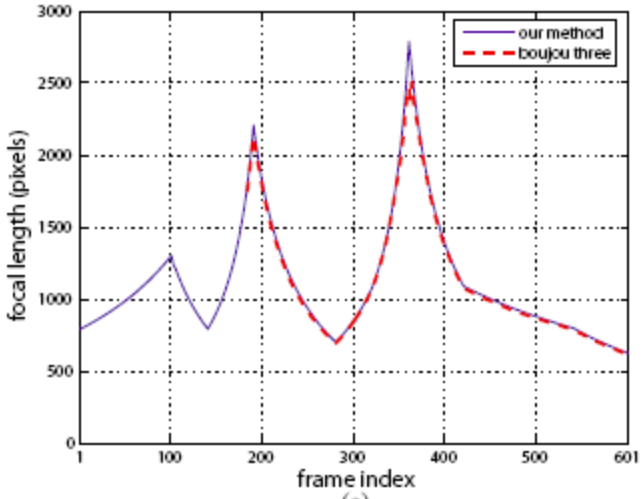
计算机生成的序列测试



(a)

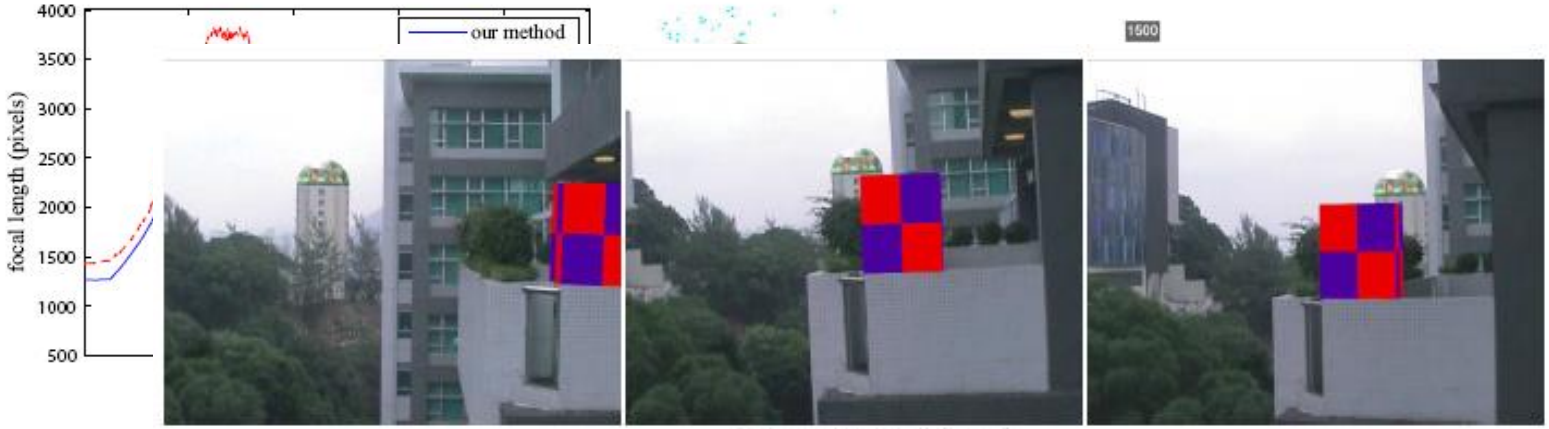


(b)



(c)

真实拍摄的序列测试

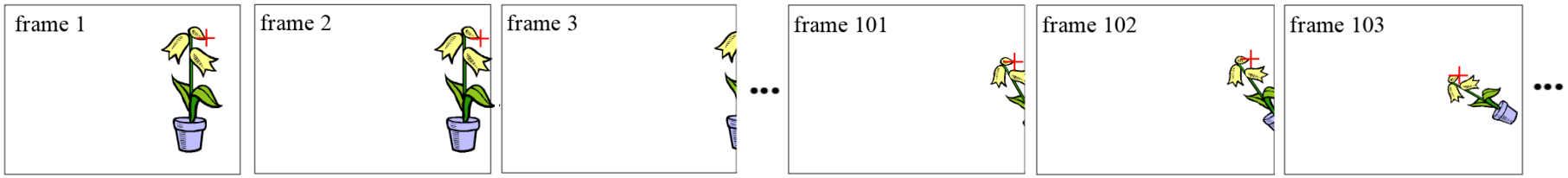


(a) our method

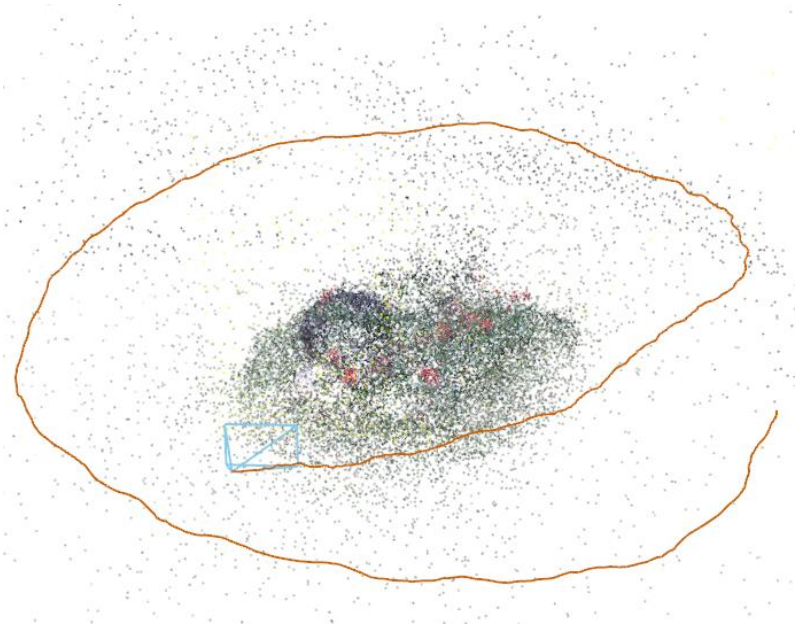
(b) captured from boujou three

The Key Issues of Feature Tracking

- How to obtain long and accurate feature tracks?
- How to handle loopback sequences?

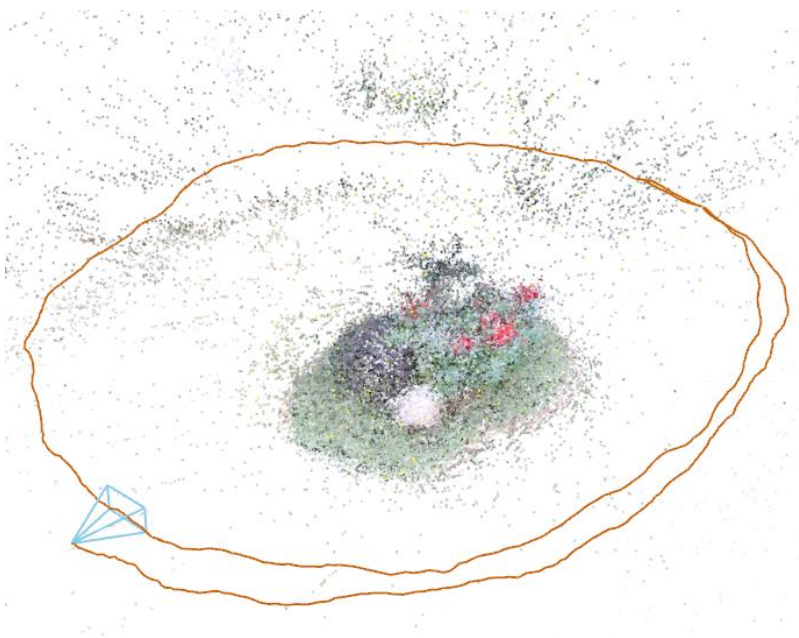


Camera Tracking for Loopback Sequences



KLT tracking

Drift!



Accurate

Feature tracking review

- Traditional Sequential Tracker
  - KLT, …
- Invariant Feature based method
  - SIFT, SURF, …
  - MSER
  - Affine-SIFT

Traditional sequential tracker

- Effective for sequential tracking
- Can not handle wide-baseline image matching

KLT matching

Invariant feature based methods

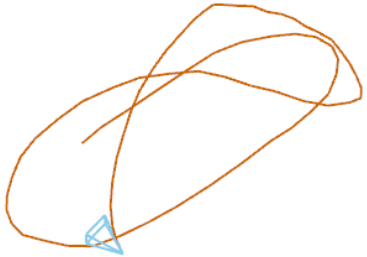
- Effective for wide-baseline matching
- Not easy to be used in consecutive point tracking (feature dropout)
  - Global indistinctiveness
  - Image noise

    Only short tracks can be obtained!

Our Approach

Consecutive Feature Tracking

Non-Consecutive Track Matching



Structure from Motion

Framework Overview

1. Detect invariant features over the entire sequence.
2. **Consecutive point tracking**:
    2.1 Match features between consecutive frames with descriptor comparison.
    2.2 Perform the second-pass matching to extend track lifetime.
3. **Non-consecutive track matching**:
    3.1 Use hierachical k-means to cluster the constructed tracks.
    3.2 Estimate the matching matrix with the grouped tracks.
    3.3 Detect overlapping subsequences and join the matched tracks.

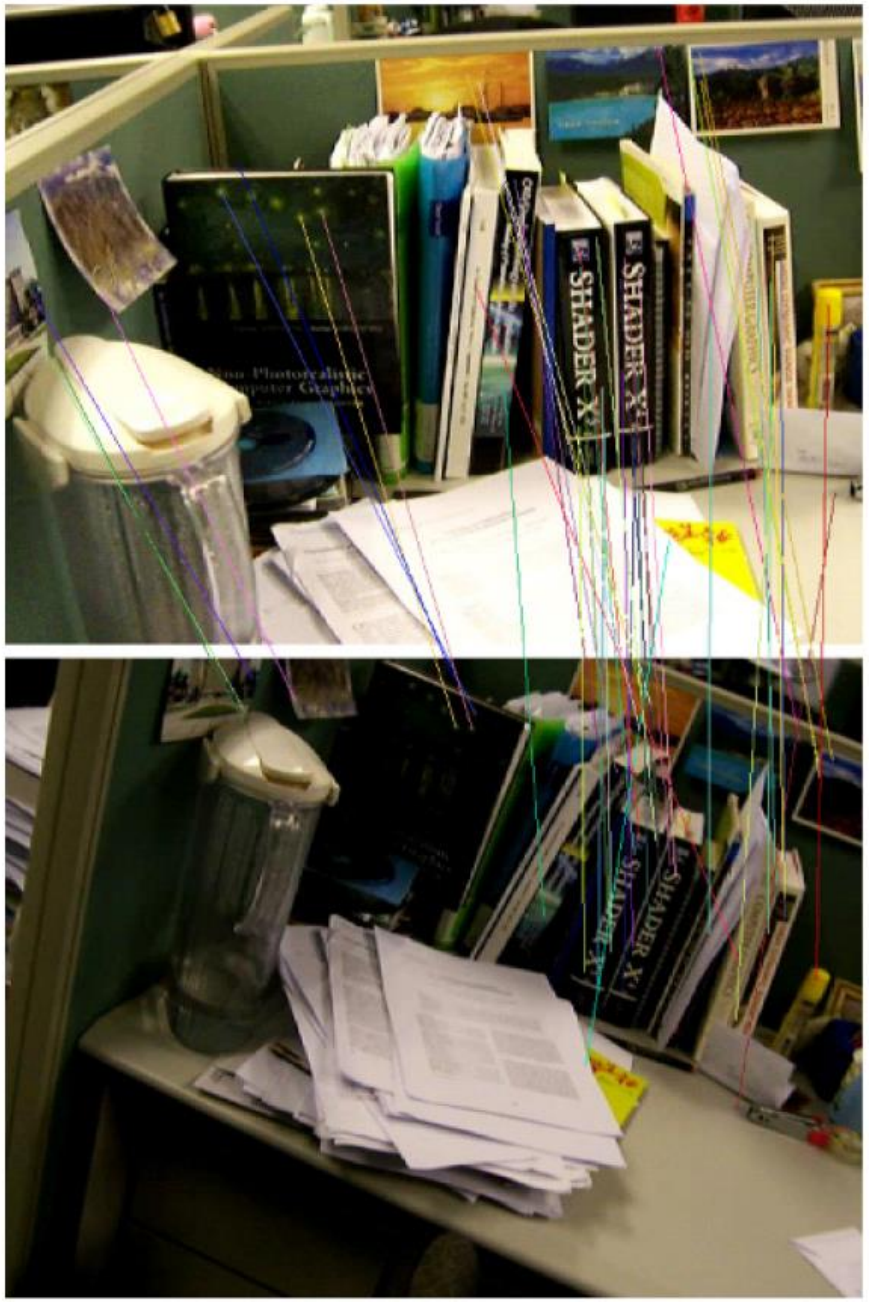
Two-Pass Matching for Consecutive Tracking

- SIFT Feature Extraction

- First-Pass Matching by Descriptor Comparison

$$c = \frac{||\mathbf{p}(\mathcal{N}_1^{t+1}(\mathbf{x}_t)) - \mathbf{p}(\mathbf{x}_t)||}{||\mathbf{p}(\mathcal{N}_2^{t+1}(\mathbf{x}_t)) - \mathbf{p}(\mathbf{x}_t)||}$$

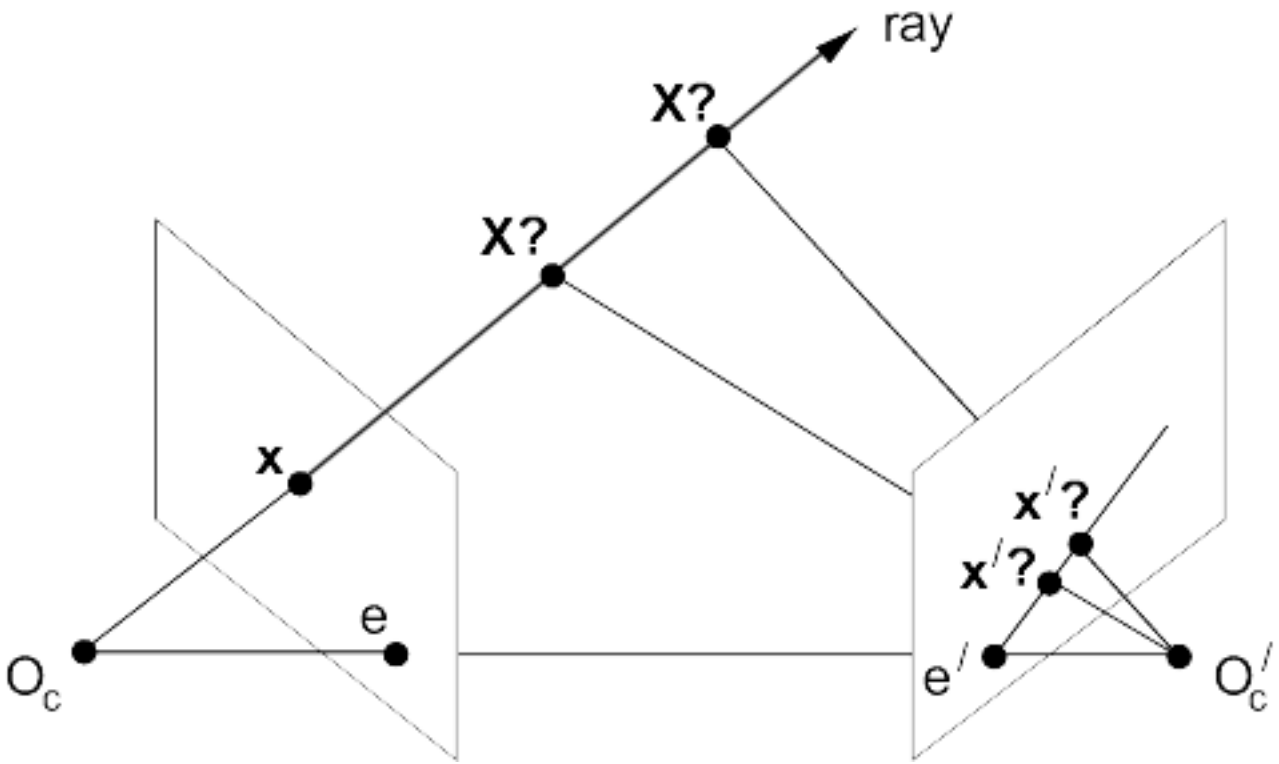
$c < \varepsilon$ Global distinctive


SHADER X
SHADER X

3D: ???

3D: Epipolar Geometry

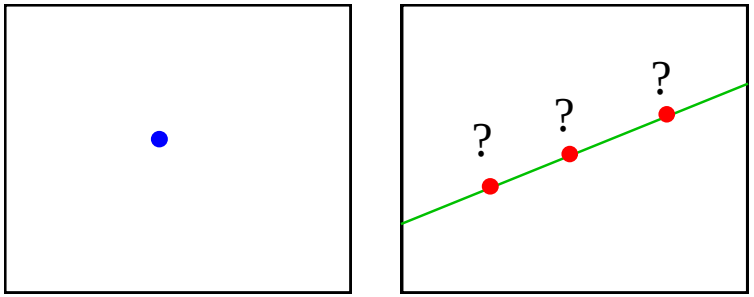
3D: Epipolar Geometry



Figures: Andrew Gee

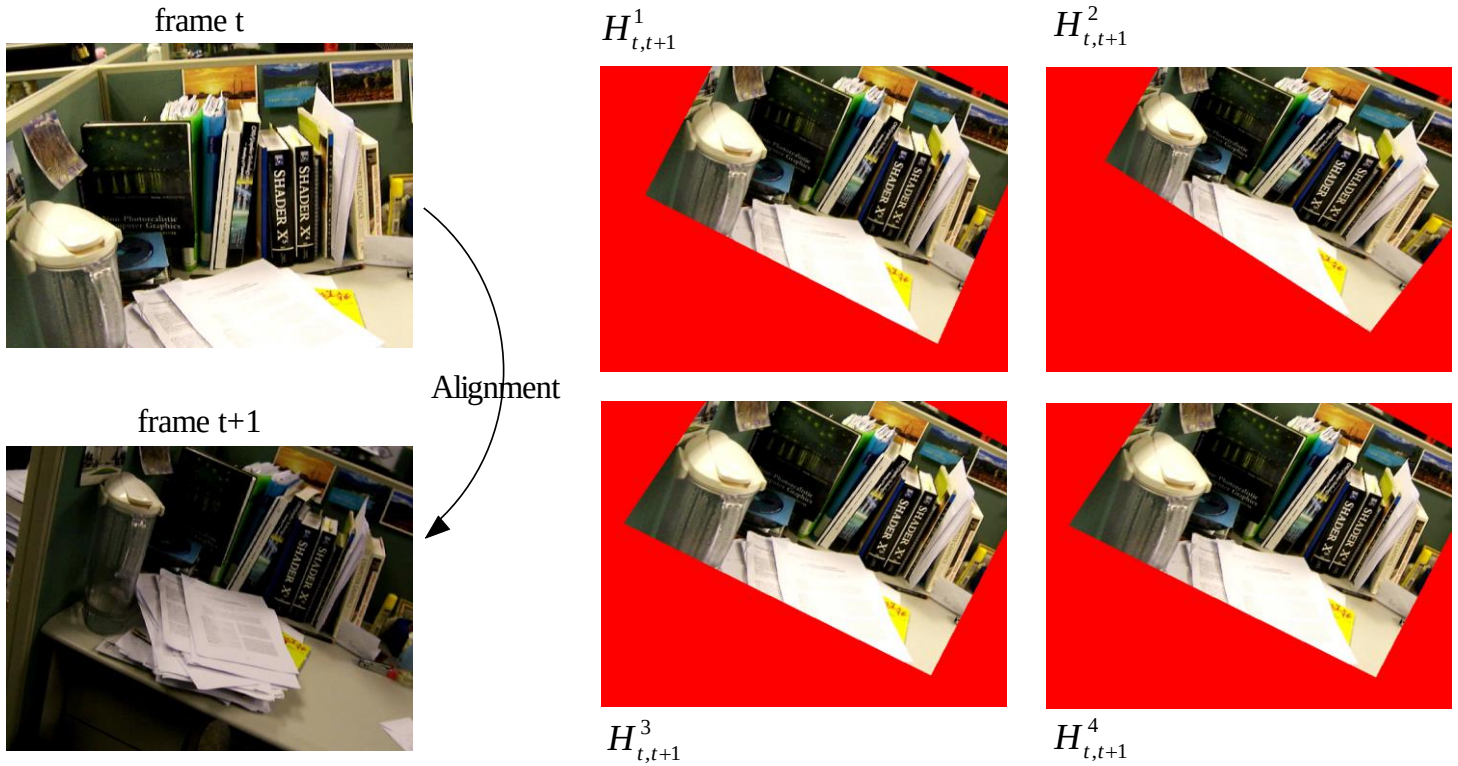
Not enough!

- How to handle image distortion?
  - Naïve window-based matching becomes unreliable!
- How to give a good position initializaton?
  - Whole line searching is still time-consuming and ambiguous with many potential correspondences.



Second-Pass Matching by Planar Motion Segmentation

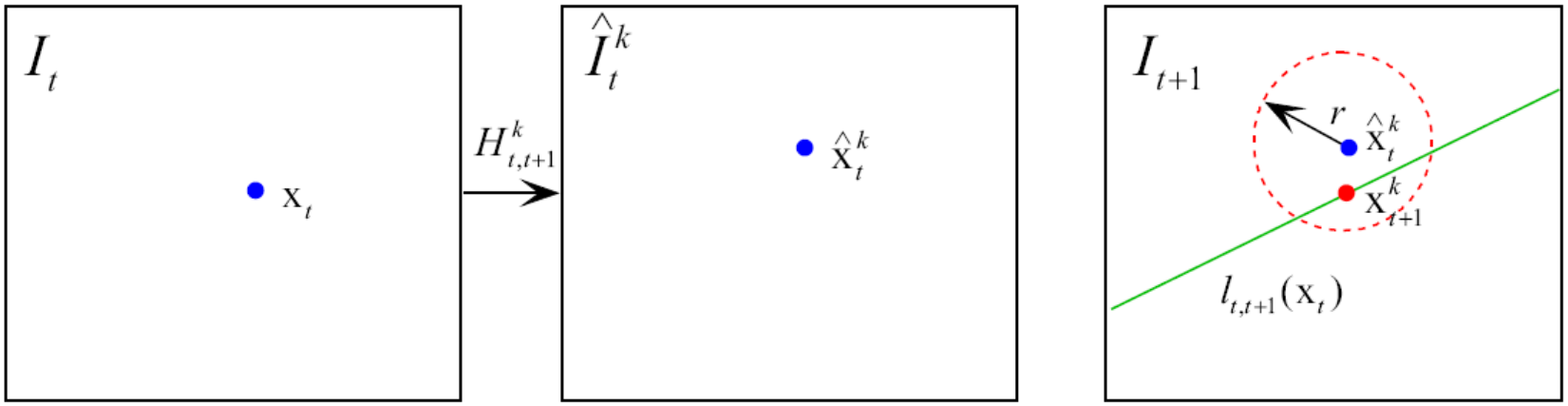
- Estimate a set of homographies $\{H_{t,t+1}^{k} | k = 1, ..., N\}$
  - Using inlier matches in first-pass matching



Second-Pass Matching by Planar Motion Segmentation

- Guided matching

$$S^k_{t,t+1}(\mathbf{x}_t) = \min_{\mathbf{x}' \in l_{t,t+1}(\mathbf{x}_t)} \sum_{\mathbf{y} \in W} ||\hat{I}^k_t(\hat{\mathbf{x}}^k_t + \mathbf{y}) - \hat{I}_{t+1}(\mathbf{x}' + \mathbf{y})||^2$$



Second-Pass Matching by Planar Motion Segmentation

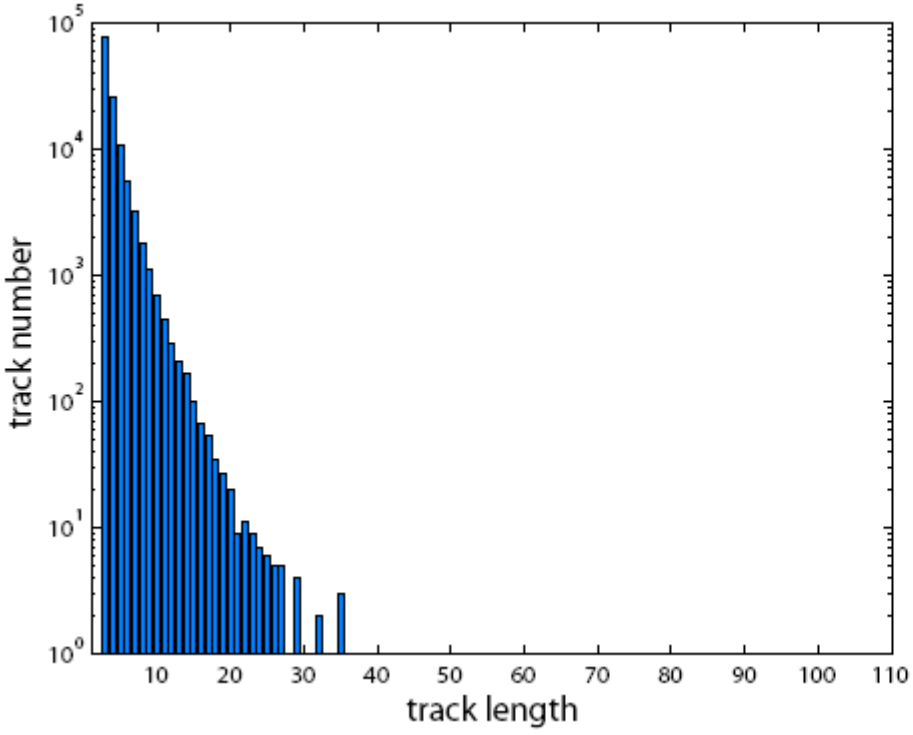
(a) First-pass matching

(b) Second-pass matching

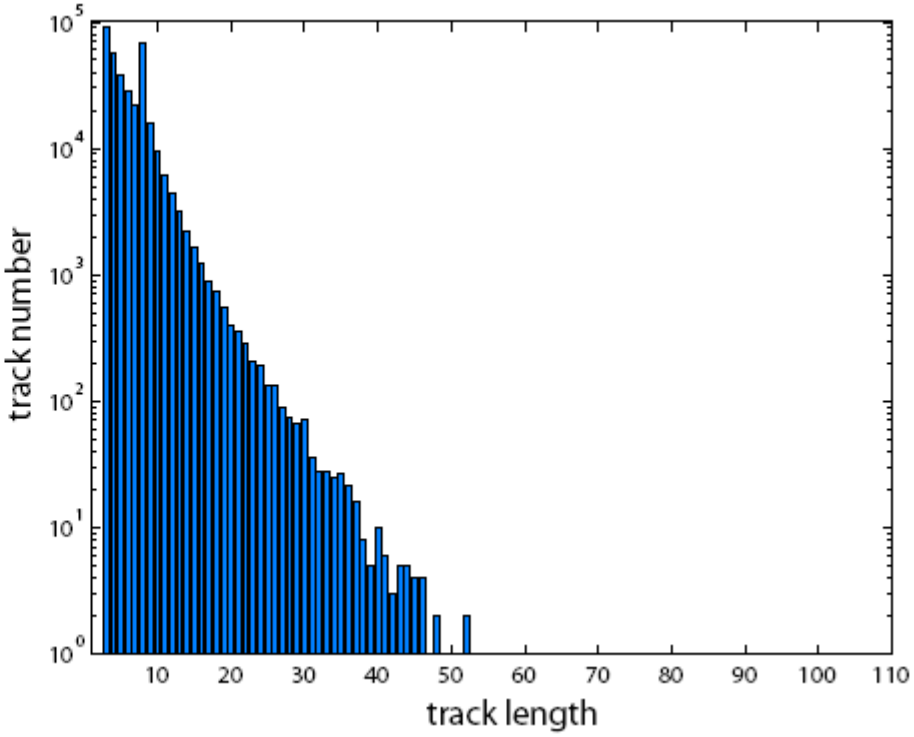
(c) Outlier rejection

SIFT Matching vs. Our Two-pass Matching in Consecutive Tracking

Track length histograms



SIFT matching



Our two-pass matching

Non-consecutive track matching

- Fast Matching Matrix Estimation
- Detect overlapping subsequences and join the matched tracks.

Non-consecutive track matching

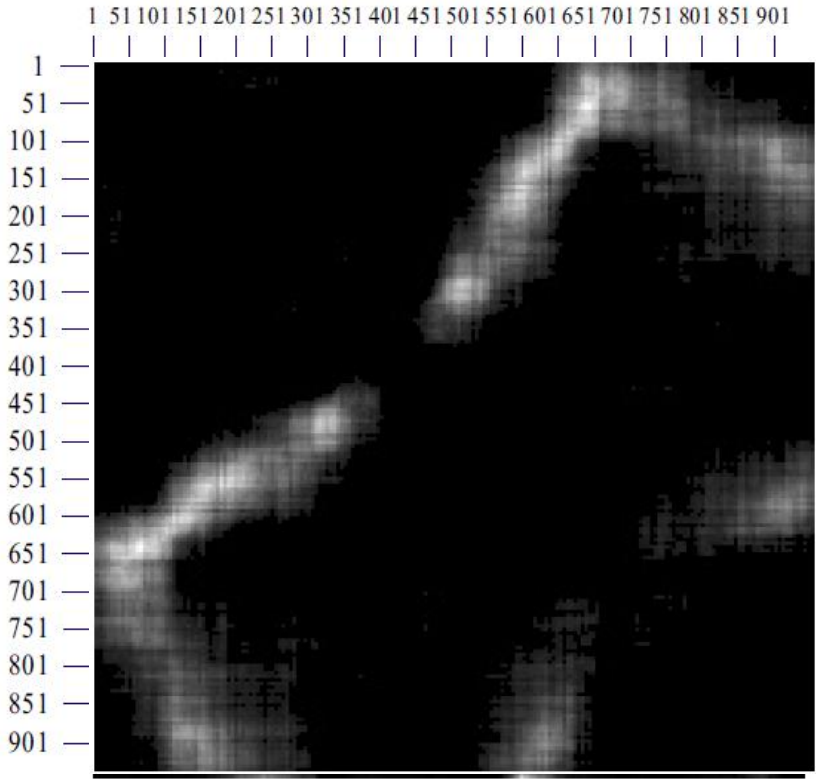
- Fast Matching Matrix Estimation

- Detect overlapping subsequences and join the matched tracks.

Fast Matching Matrix Estimation

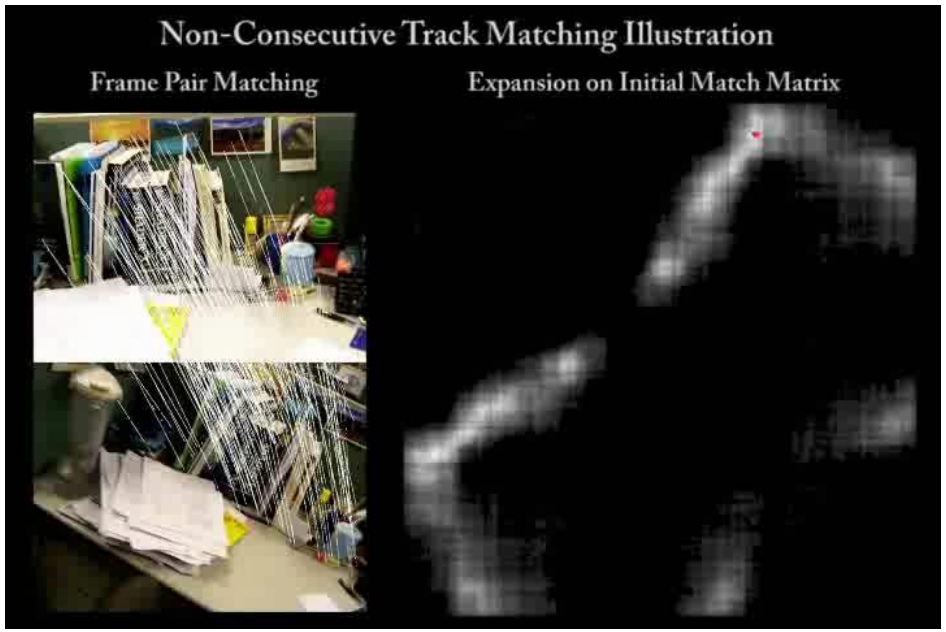
- Each track has a group of description vectors $\mathcal{P}_{\mathcal{X}} = \{\mathbf{p}(\mathbf{x}_t) | t \in f(\mathcal{X})\}$
- Track descriptor $\mathbf{p}(\mathcal{X})$
- Use a hierarchical K-means approach to cluster the track descriptors

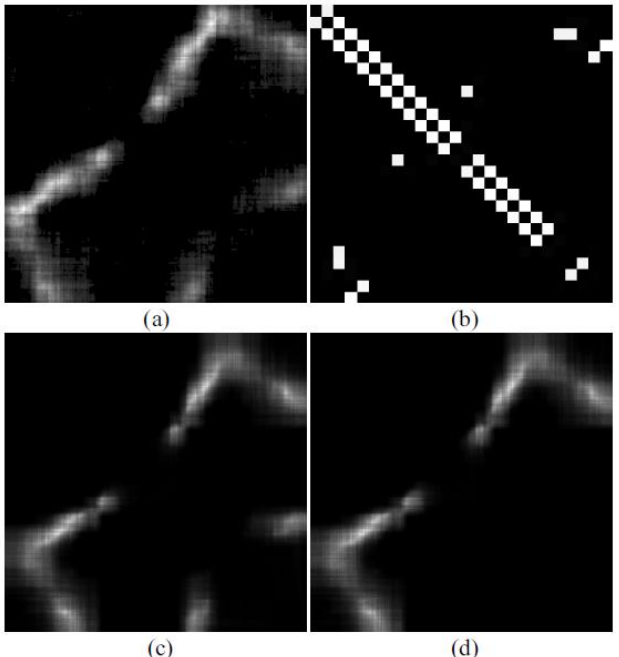
Fast Matching Matrix Estimation



Non-Consecutive Track Matching

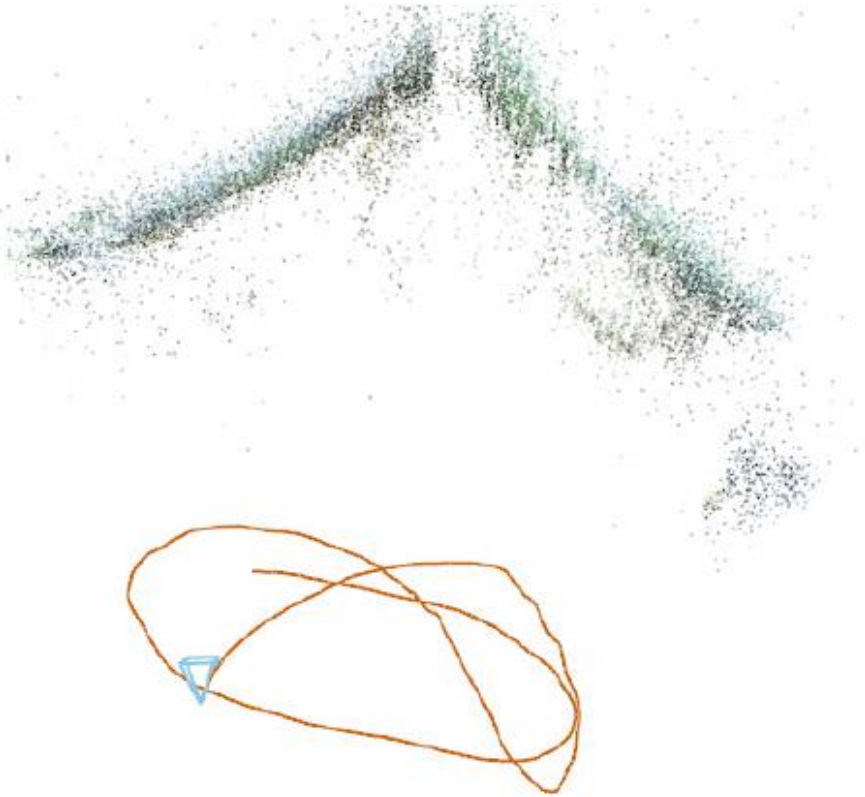
- Simultaneously Match Images and Refine Matching Matrix
  - Refine the matching matrix after matching the common features of the selected image pairs.
  - More reliably find the best matching images with the updated matching matrix.





Comparison

Without non-consecutive track matching



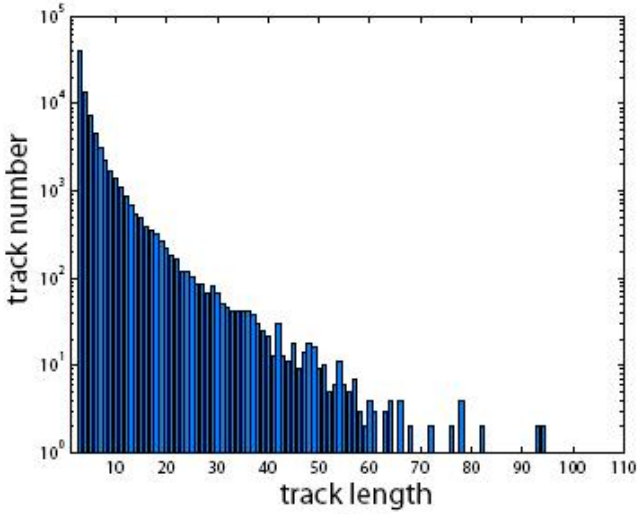
With non-consecutive track matching

More comparisons

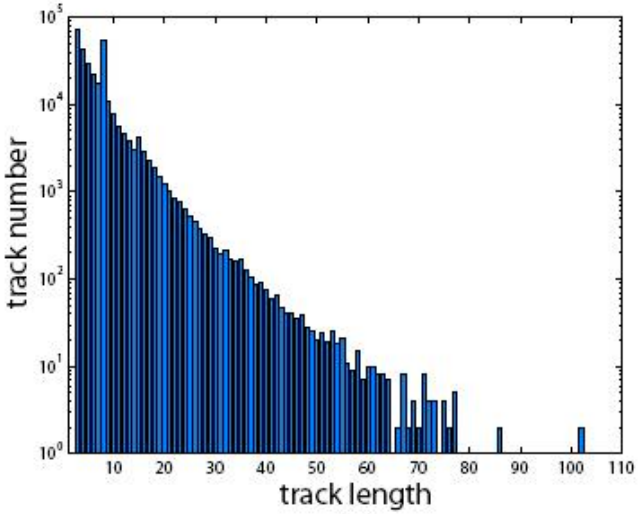
- Four methods
  - Consecutive SIFT matching (C-SIFT)
  - Our consecutive point tracking (CPT)
  - Brute-force SIFT matching (BF-SIFT)
  - Our consecutive point tracking with non-consecutive track matching (CPT+NCTM)

Algorithms	Running Time	Average Track Length	Reprojection error
C-SIFT	8 minutes	4.25	0.76 pixels
CPT	15 minutes	6.07	0.98 pixels
BF-SIFT	187 minutes	6.66	1.05 pixels
CPT+NCTM	25 minutes	7.26	1.09 pixels

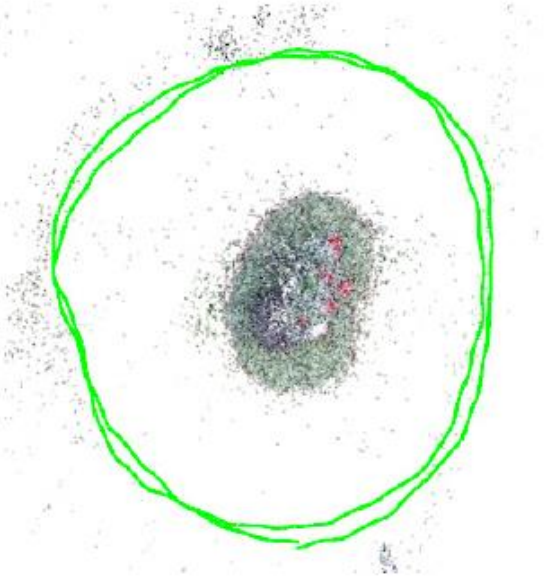
(a)



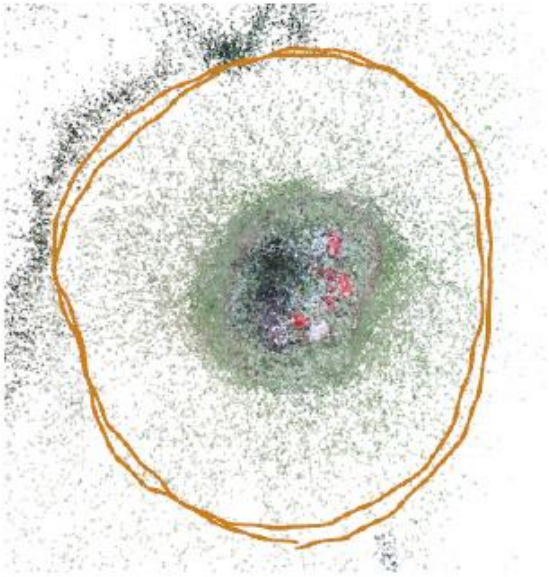
(b)



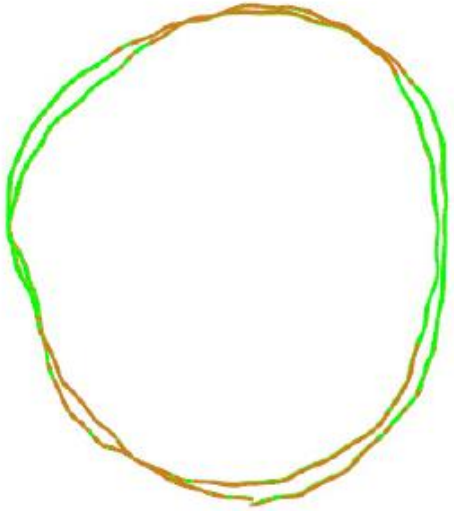
(c)



(d)



(e)



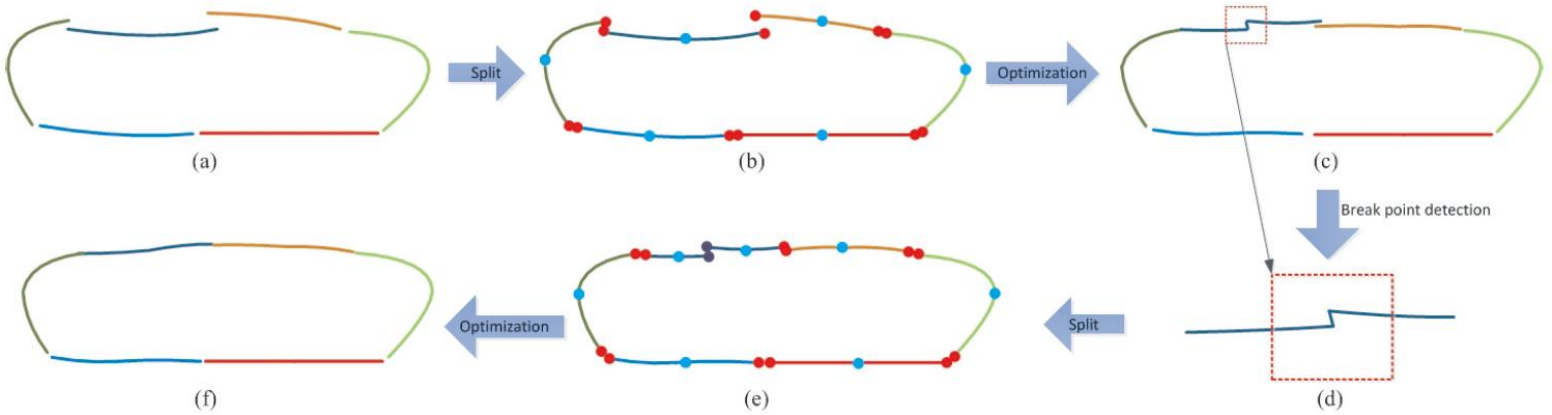
(f)

The Difficulties for Large-Scale SfM

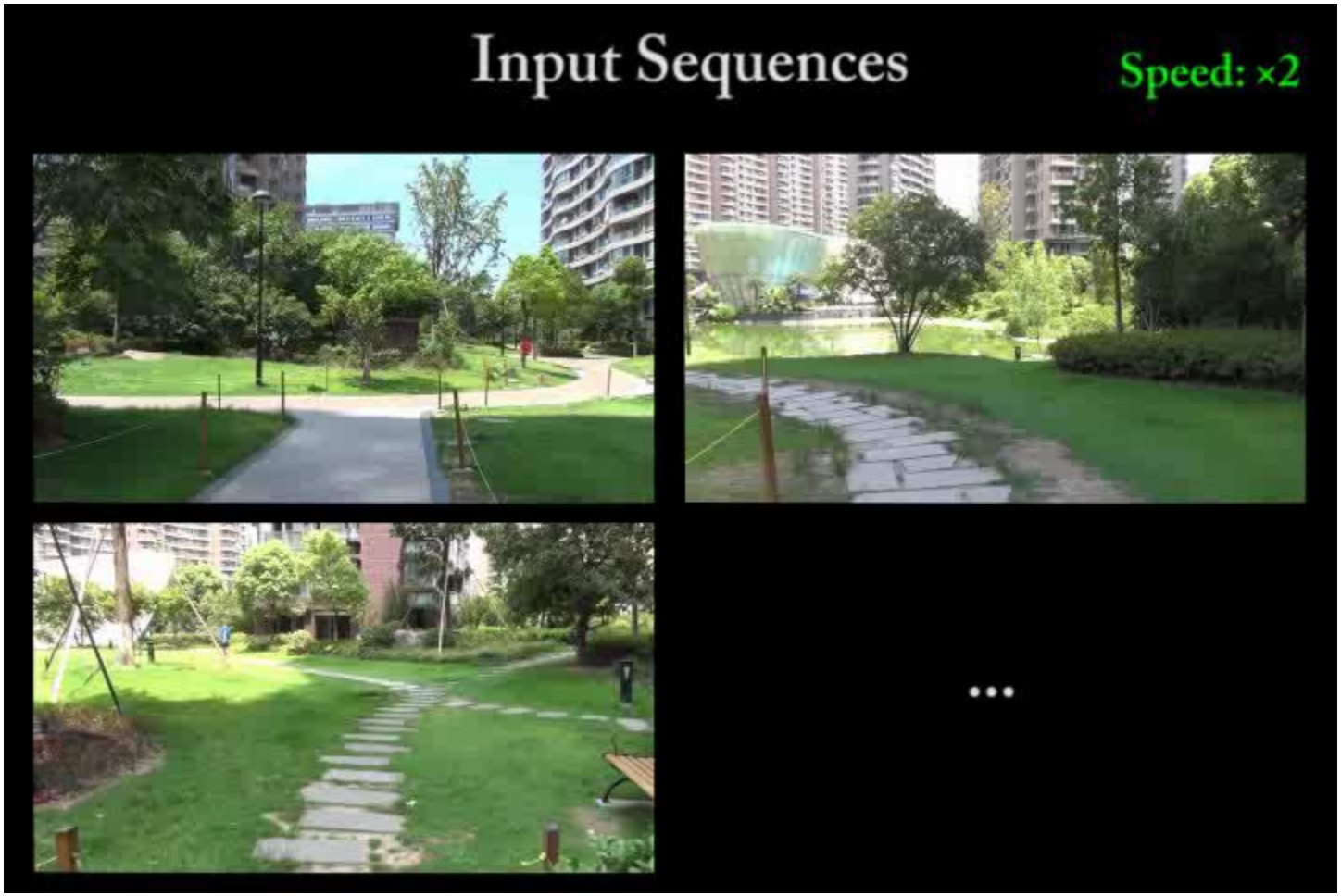
- Global Bundle Adjustment
  - Huge variables
  - Memory limit
  - Time-consuming
- Iterative Local Bundle Adjustment
  - The large error is difficult to be propagated to the whole sequence.
  - Easily stuck in a local optimum.

Segment-based Progressive SfM

- Split a long sequence to multiple short sequences.
- Perform SfM for each sequence and align them together.
- Detect the ``split point'' and further split the sequence if the reprojection error is large.
- The above procedure is repeated until the error is less than a threshold.



Large-Scale Structure-from-Motion



6 long sequences with 95K+ frames, feature tracking 74 minutes, SfM estimation 16 minutes (single thread), 17.7fps averagely

VisualSFM: 57 minutes with GPU acceleration for SfM estimation

Thank you!