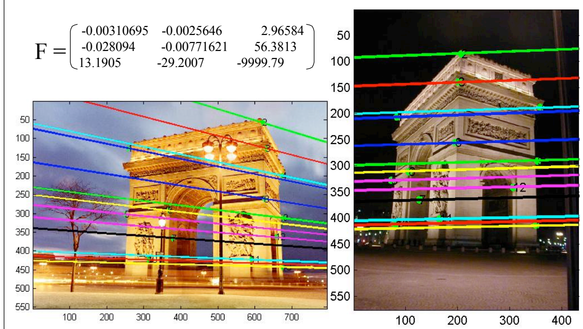
Lecture 20:
The Eight-Point Algorithm

Readings T&V 7.3 and 7.4

Reminder:

$$F = \begin{bmatrix} -0.00310695 & -0.0025646 & 2.96584 \\ -0.028094 & -0.00771621 & 56.3813 \\ 13.1905 & -29.2007 & -9999.79 \end{bmatrix}$$



Essential/Fundamental Matrix

The essential and fundamental matrices are 3x3 matrices that “encode” the epipolar geometry of two views.

Motivation: Given a point in one image, multiplying by the essential/fundamental matrix will tell us which epipolar line to search along in the second view.

E/F Matrix Summary

Longuet-Higgins equation $$p_r^T E p_l = 0$$

Epipolar lines: $$\tilde{p_r}^T \tilde{l_r} = 0 \qquad \tilde{p_l}^T \tilde{l_l} = 0$$
$$\tilde{l_r} = E p_l \qquad \tilde{l_l} = E^T p_r$$

Epipoles: $$e_r^T E = 0 \qquad E e_l = 0$$

E vs F: E works in film coords (calibrated cameras)
F works in pixel coords (uncalibrated cameras)

Computing F from Point Matches

- Assume that you have m correspondences
- Each correspondence satisfies:

$$\bar{p_r}_i^T F \bar{p_l}_i = 0 \quad i = 1, \ldots, m$$

- F is a 3x3 matrix (9 entries)
- Set up a **HOMOGENEOUS** linear system with 9 unknowns

Computing F

$$\bar{p_l}_i = (x_i \; y_i \; 1)^T \quad \bar{p_r}_i = (x'_i \; y'_i \; 1)^T$$

$$\bar{p_r}_i^T F \bar{p_l}_i = 0 \quad i = 1, \ldots, m$$

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

Computing F

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

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

Computing F

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

Given m point correspondences…

$$\begin{bmatrix} x_1 x'_1 & x_1 y'_1 & x_1 & y_1 x'_1 & y_1 y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_m x'_m & x_m y'_m & x_m & y_m x'_m & y_m y'_m & y_m & x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

Think: how many points do we need?

How Many Points?

Unlike a homography, where each point correspondence contributes two constraints (rows in the linear system of equations), for estimating the essential/fundamental matrix, each point only contributes one constraint (row). [because the Longuet-Higgins / Epipolar constraint is a scalar eqn.]

Thus need at least 8 points.

Hence: The Eight Point algorithm!

Solving Homogeneous Systems

Self-study

Assume that we need the non trivial solution of:

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

with m equations and n unknowns, m >= n – 1 and rank(A) = n-1

Since the norm of x is arbitrary, we will look for a solution with norm ||x|| = 1

Least Square solution

Self-study

We want Ax as close to 0 as possible and ||x|| =1:

$$\min_{\mathbf{x}} ||A\mathbf{x}||^2 \text{ s.t. } ||x||^2 = 1$$

$$||A\mathbf{x}||^2 = (A\mathbf{x})^T(A\mathbf{x}) = \mathbf{x}^T A^T A\mathbf{x}$$

$$||\mathbf{x}||^2 = \mathbf{x}^T\mathbf{x} = 1$$

Optimization with constraints

Self-study

Define the following cost:

$$\mathcal{L}(\mathbf{x}) = \mathbf{x}^T A^T A\mathbf{x} - \lambda(\mathbf{x}^T\mathbf{x} - 1)$$

This cost is called the LAGRANGIAN *cost* and λ is called the LAGRANGIAN *multiplier*

The Lagrangian incorporates the constraints into the cost function by introducing extra variables.

Optimization with constraints

Self-study

$$\min_{\mathbf{x}} \left\{ \mathcal{L}(\mathbf{x}) = \mathbf{x}^T A^T A \mathbf{x} - \lambda(\mathbf{x}^T \mathbf{x} - 1) \right\}$$

Taking derivatives wrt to x and λ:

$$A^T A \mathbf{x} - \lambda \mathbf{x} = 0$$
$$\mathbf{x}^T \mathbf{x} - 1 = 0$$

- The first equation is an eigenvector problem
- The second equation is the original constraint

Optimization with constraints

Self-study

$$A^T A \mathbf{x} - \lambda \mathbf{x} = 0$$
$$A^T A \mathbf{x} = \lambda \mathbf{x}$$

- x is an eigenvector of $A^T A$ with eigenvalue λ: e_λ

$$\mathcal{L}(\mathbf{e}_\lambda) = \mathbf{e}_\lambda{}^T A^T A \mathbf{e}_\lambda - \lambda(\mathbf{e}_\lambda{}^T \mathbf{e}_\lambda - 1)$$
$$\mathcal{L}(\mathbf{e}_\lambda) = \lambda \mathbf{e}_\lambda{}^T \mathbf{e}_\lambda = \lambda$$

- We want the eigenvector with <u>smallest</u> eigenvalue

Self-study

We can find the eigenvectors and eigenvalues of $A^T A$ by finding the Singular Value Decomposition of A

Singular Value Decomposition (SVD)

Self-study

Any m x n matrix A can be written as the product of 3 matrices:

$$A = UDV^T$$

Where:
- U is m x m and its columns are orthonormal vectors
- V is n x n and its columns are orthonormal vectors
- D is m x n diagonal and its diagonal elements are called the singular values of A, and are such that:

$\sigma_1 , \sigma_2 , \ldots \sigma_n , 0$

SVD Properties

Self-study

$$A = UDV^T$$

- The columns of U are the eigenvectors of AA^T
- The columns of V are the eigenvectors of $A^T A$
- The squares of the diagonal elements of D are the eigenvalues of AA^T and $A^T A$

Computing F: The 8 pt Algorithm

$$\begin{bmatrix} x_1 x'_1 & x_1 y'_1 & x_1 & y_1 x'_1 & y_1 y'_1 & y_1 & x'_1 & y'_1 & 1 \\ & & & & \vdots & & & & \\ x_8 x'_8 & x_8 y'_8 & x_8 & y_8 x'_8 & y_8 y'_8 & y_8 & x'_8 & y'_8 & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

$A\mathbf{x} = 0$ <u>A has rank 8</u>

$$\min_{\mathbf{x}} ||A\mathbf{x}||^2 \text{ s.t. } ||x||^2 = 1$$

- Find eigenvector of $A^T A$ with <u>smallest</u> eigenvalue!

Algorithm EIGHT_POINT

The input is formed by m point correspondences, m >= 8

- Construct the m x 9 matrix A
- Find the SVD of A: $A = UDV^T$
- The entries of F are the components of the column of V corresponding to the least s.v.

Algorithm EIGHT_POINT

F must be singular (remember, it is rank 2, since it is important for it to have a left and right nullspace, i.e. the epipoles). To enforce rank 2 constraint:

- Find the SVD of F: $F = U_f D_f V_f^T$
- Set smallest s.v. of F to 0 to create D'_f
- Recompute F: $F = U_f D'_f V_f^T$

Numerical Details

Self-study

- The coordinates of corresponding points can have a wide range leading to numerical instabilities.
- It is better to first normalize them so they have average 0 and stddev 1 and denormalize F at the end:

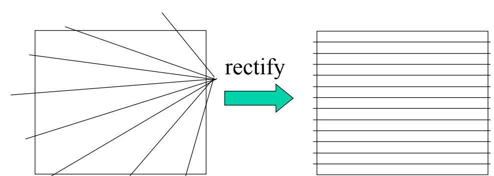
$$\hat{x}_i = T x_i \qquad \hat{x}'_i = T' x'_i \qquad T = \begin{bmatrix} \frac{1}{\sigma^2} & 0 & -\mu_x \\ 0 & \frac{1}{\sigma^2} & -\mu_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$F = (T')^{-1} F_n T$$

Hartley preconditioning algorithm: this was an "optional" topic in Lecture 16. Go back and look if you want to know more.

A Practical Issue

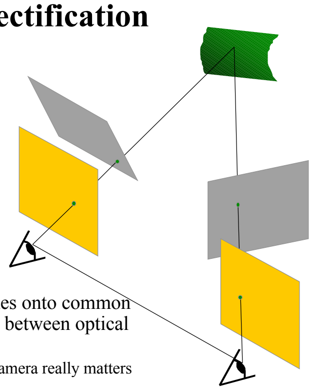
How to "rectify" the images so that any scan-line stereo alorithm that works for simple stereo can be used to find dense matches (i.e. compute a disparity image for every pixel).



general epipolar lines

parallel epipolar lines

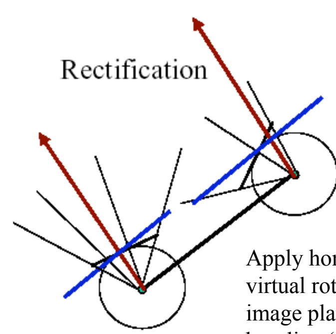
Stereo Rectification



- Image Reprojection
  - reproject image planes onto common plane parallel to line between optical centers
- Notice, only focal point of camera really matters

Seitz, UW

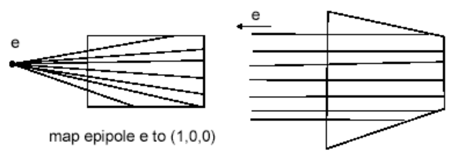
General Idea



Apply homography representing a virtual rotation to bring the image planes parallel with the baseline (epipoles go to infinity).

General Idea, continued

Apply projective transformation so that epipolar lines correspond to horizontal scanlines



$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = He$$

map epipole e to (1,0,0)

try to minimize image distortion

Note that rectified images usually not rectangular

Image Rectification

Method from T&V 7.3.7

Assuming extrinsic parameters R & T are known, compute a 3D rotation that makes conjugate epipolar lines collinear and parallel to the horizontal image axis

Remember: a rotation around focal point of camera is just a 2D homography in the image!

Note: this method from the book assumes calibrated cameras (we can recover R,T from the E matrix). In a moment, we will see a more general approch that uses F matrix.

Image Rectification

- Rectification involves two rotations:
  - First rotation sends epipoles to infinity
  - Second rotation makes epipolar lines parallel
- Rotate the left and right cameras with first R_1 (constructed from translation T)
- Rotate the right camera with the R matrix
- Adjust scales in both camera references

Image Rectification

Build the rotation: $$R_{rect} = \begin{bmatrix} \mathbf{e_1}^T \\ \mathbf{e_2}^T \\ \mathbf{e_3}^T \end{bmatrix}$$

with: $$\mathbf{e_1} = \frac{\mathbf{T}}{||\mathbf{T}||}$$

where T is just a unit vector representing the epipole in the left image. We know how to compute this from E, from last class.

Image Rectification

Build the rotation: $$R_{rect} = \begin{bmatrix} \mathbf{e_1}^T \\ \mathbf{e_2}^T \\ \mathbf{e_3}^T \end{bmatrix}$$

with: $$\mathbf{e_1} = \frac{\mathbf{T}}{||\mathbf{T}||}$$

$$\mathbf{e_2} = \frac{1}{\sqrt{T_x^2 + T_y^2}} \begin{bmatrix} -T_y \\ T_x \\ 0 \end{bmatrix} \qquad \mathbf{e_3} = \mathbf{e_1} \times \mathbf{e_2}$$

Verify that this homography maps e1 to [1 0 0]'

Algorithm Rectification

- Build the matrix R_{rect}
- Set $R_l = R_{rect}$ and $R_r = R.R_{rect}$
- For each left point $p_l = (x,y,f)^T$
  - compute $R_l\, p_l = (x', y', z')^T$
  - Compute $p'_l = f/z'\, (x', y'\, z')^T$
- Repeat above for the right camera with R_r

Example

Example

Rectified Pair

A Better Approach

The traditional method does not work when epipoles are in the image (for example, camera translating forward)

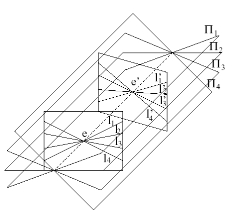


Figure 1. Epipolar geometry with the epipoles in the images. Note that the matching ambiguity is reduced to half epipolar lines.

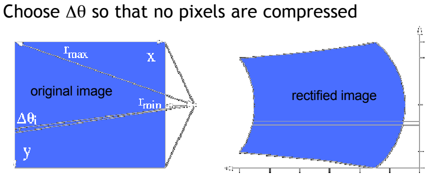
Good paper: "Simple and efficient rectification methods for general motion", Marc Pollefeys, R.Koch, L.VanGool, ICCV99.

General idea: polar rectification around the epipoles.

Polar rectification

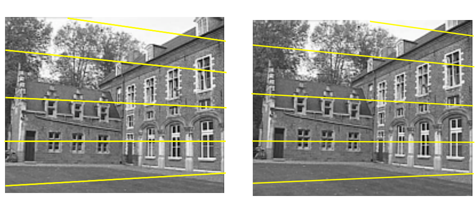
(Pollefeys et al. ICCV'99)

Polar re-parameterization around epipoles
Requires only (oriented) epipolar geometry
Preserve length of epipolar lines
Choose $\Delta\theta$ so that no pixels are compressed

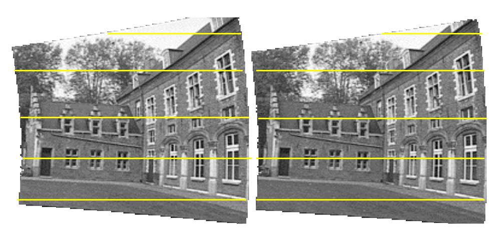


Works for all relative motions
Guarantees minimal image size

polar rectification: example

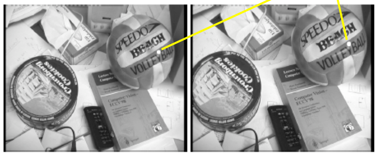


polar rectification: example



Example

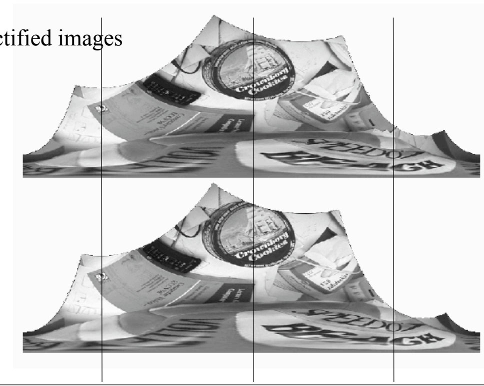
epipoles



Input images

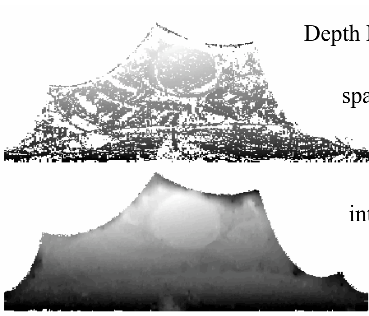
Example (cont)

Rectified images



Example (cont)

Depth Maps

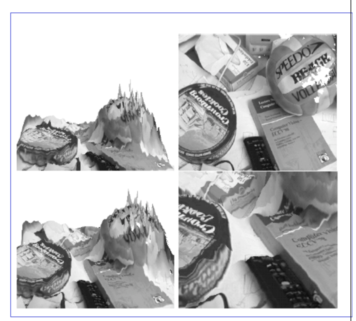


sparse

interpolated

Example (cont)

Depth map in
pixel coords



Views of texture mapped
depth surface