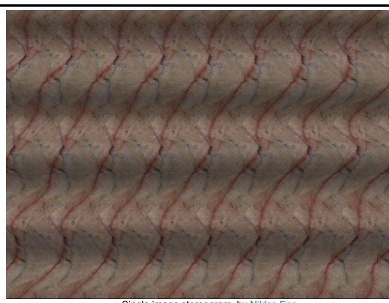
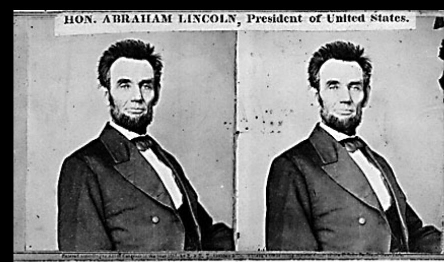
Stereo



Single image stereogram, by Niklas Een

Readings

- Trucco & Verri, Chapter 7
  - Read through 7.1, 7.2.1, 7.2.2, 7.3.1, 7.3.2, 7.3.7 and 7.4, 7.4.1.
  - The rest is optional.



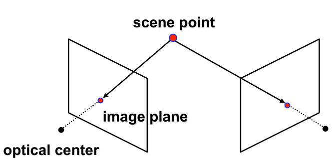
Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923

Teesta suspension bridge-Darjeeling, India

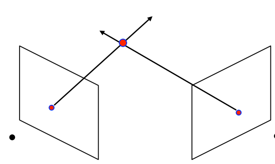
Woman getting eye exam during immigration procedure at Ellis Island, c. 1905 - 1920 , UCR Museum of Phography

Mark Twain at Pool Table", no date, UCR Museum of Photography

Stereo



Stereo



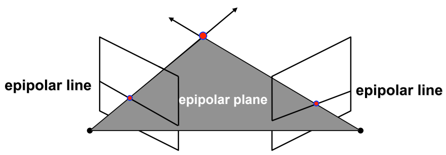
Basic Principle: Triangulation

- Gives reconstruction as intersection of two rays
  - Requires
  - camera pose (calibration)
  - ***point correspondence***

Stereo correspondence

Determine Pixel Correspondence

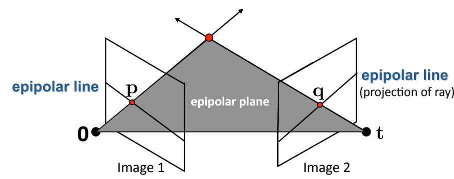
- Pairs of points that correspond to same scene point



Epipolar Constraint

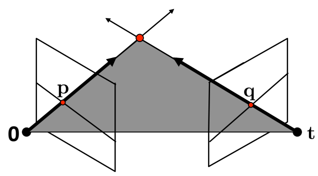
- Reduces correspondence problem to 1D search along *conjugate epipolar lines*
- Java demo: http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html

Fundamental matrix



- This *epipolar geometry* of two views is described by a Very Special 3x3 matrix $\mathbf{F}$, called the *fundamental matrix*
- $\mathbf{F}$ maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point **p** is: $\mathbf{Fp}$
- *Epipolar constraint* on corresponding points: $\mathbf{q}^T\mathbf{F}\mathbf{p} = 0$

Fundamental matrix – uncalibrated case



$\mathbf{K}_1$: intrinsics of camera 1

$\mathbf{K}_2$: intrinsics of camera 2

$\mathbf{R}$: rotation of image 2 w.r.t. camera 1

$$\mathbf{q}^T \underbrace{\mathbf{K}_2^{-T}\mathbf{R}\,[\mathbf{t}]_\times\,\mathbf{K}_1^{-1}}_{\mathbf{F}} \mathbf{p} = 0$$

$\mathbf{F}$ ← the Fundamental matrix

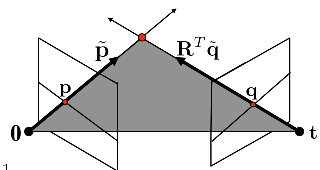
Cross-product as linear operator

Useful fact: Cross product with a vector **t** can be represented as multiplication with a (*skew-symmetric*) 3x3 matrix

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

$$\mathbf{t} \times \tilde{\mathbf{p}} = [\mathbf{t}]_\times \tilde{\mathbf{p}}$$

Fundamental matrix – calibrated case



$\tilde{\mathbf{p}} = \mathbf{K}_1^{-1}\mathbf{p}$: ray through **p** in camera 1's (and world) coordinate system

$\tilde{\mathbf{q}} = \mathbf{K}_2^{-1}\mathbf{q}$: ray through **q** in camera 2's coordinate system

$$\tilde{\mathbf{q}}^T \underbrace{\mathbf{R}\,[\mathbf{t}]_\times}_{\mathbf{E}} \tilde{\mathbf{p}} = 0 \qquad \tilde{\mathbf{q}}^T \mathbf{E} \tilde{\mathbf{p}} = 0$$

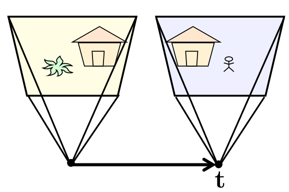
$\mathbf{E}$ ← the Essential matrix

Properties of the Fundamental Matrix

- $\mathbf{F}\mathbf{p}$ is the epipolar line associated with $\mathbf{p}$
- $\mathbf{F}^T\mathbf{q}$ is the epipolar line associated with $\mathbf{q}$
- $\mathbf{F}\mathbf{e}_1 = \mathbf{0}$ and $\mathbf{F}^T\mathbf{e}_2 = \mathbf{0}$
- $\mathbf{F}$ is rank 2
- How many parameters does **F** have?

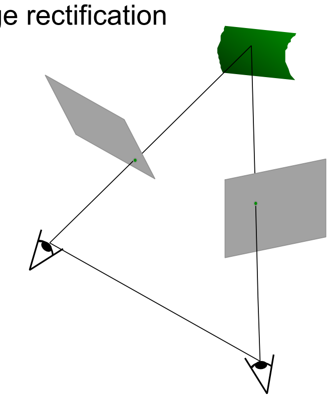
14

Rectified case

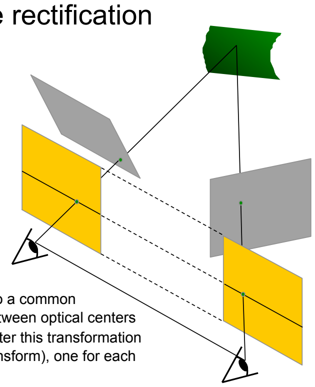


$$\mathbf{R} = \mathbf{I}_{3\times3}$$
$$\mathbf{t} = [\ 1\quad 0\quad 0\ ]^T$$
$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Stereo image rectification



Stereo image rectification

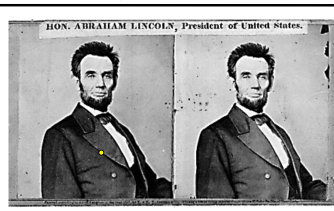


- reproject image planes onto a common plane parallel to the line between optical centers
- pixel motion is horizontal after this transformation
- two homographies (3x3 transform), one for each input image reprojection
- C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. IEEE Conf. Computer Vision and Pattern Recognition, 1999.

Estimating **F**

- If we don't know $\mathbf{K}_1$, $\mathbf{K}_2$, **R**, or **t**, can we estimate **F** for two images?
- Yes, given enough correspondences. We'll see soon…

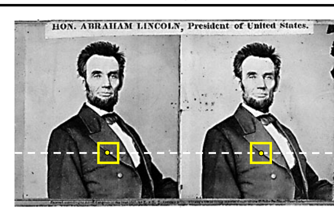
Stereo Matching



Given a pixel in the left image, how to find its match?

- Assume the photos have been rectified

Your basic stereo algorithm



For each epipolar line

For each pixel in the left image

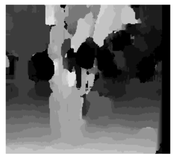
- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

- This should look familar…

Window size



W = 3

W = 20

Effect of window size

- Smaller window
  - +
  - –
- Larger window
  - +
  - –

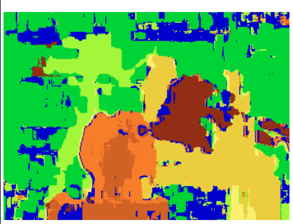
Stereo results

- Data from University of Tsukuba
- Similar results on other images without ground truth


Scene


Ground truth

Results with window search


Window-based matching
(best window size)


Ground truth

Better methods exist...

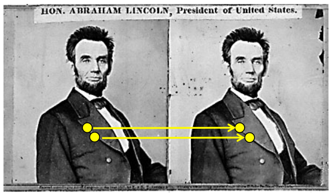

State of the art method


Ground truth

Boykov et al., Fast Approximate Energy Minimization via Graph Cuts, International Conference on Computer Vision, September 1999.

For the latest and greatest: http://www.middlebury.edu/stereo/

Stereo as energy minimization



What defines a good stereo correspondence?

1. Match quality
   – Want each pixel to find a good match in the other image
2. Smoothness
   – If two pixels are adjacent, they should (usually) move about the same amount

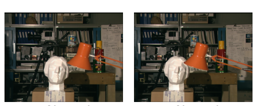
Stereo as energy minimization

- Find disparity map *d* that minimizes an energy function $E(d)$
- Simple pixel / window matching

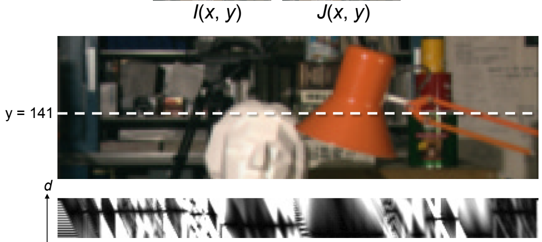
$$E(d) = \sum_{(x,y)\in I} C(x, y, d(x, y))$$

$C(x, y, d(x, y))$ = SSD distance between windows *I*(*x*, *y*) and *J*(*x* + *d*(*x*,*y*), *y*)

Stereo as energy minimization



I(*x*, *y*) *J*(*x*, *y*)



C(*x*, *y*, *d*); the *disparity space image* (DSI)

Stereo as energy minimization

Simple pixel / window matching: choose the minimum of each column in the DSI independently:

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

Stereo as energy minimization

Better objective function

$$E(d) = \underbrace{E_d(d)}_{\text{match cost}} + \lambda \underbrace{E_s(d)}_{\text{smoothness cost}}$$

match cost: Want each pixel to find a good match in the other image

smoothness cost: Adjacent pixels should (usually) move about the same amount

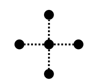
Stereo as energy minimization

$$E(d) = E_d(d) + \lambda E_s(d)$$

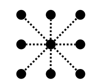
match cost: $E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y))$

smoothness cost: $E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q)$

$\mathcal{E}$: set of neighboring pixels



4-connected neighborhood



8-connected neighborhood

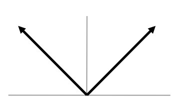
Smoothness cost

$$E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q)$$

How do we choose *V*?

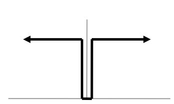
$$V(d_p, d_q) = |d_p - d_q|$$

L_1 distance



$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$

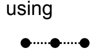
"Potts model"



Dynamic programming

$$E(d) = E_d(d) + \lambda E_s(d)$$

Can minimize this independently per scanline using dynamic programming (DP)



$D(x, y, d)$: minimum cost of solution such that *d*(*x*,*y*) = *d*

$$D(x, y, d) = C(x, y, d) + \min_{d'} \{D(x-1, y, d') + \lambda |d - d'|\}$$

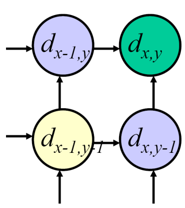
Dynamic programming

Finds "smooth" path through DPI from left to right

Dynamic Programming

Dynamic programming

Can we apply this trick in 2D as well?



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

Slide credit: D. Huttenloche

Stereo as a minimization problem

$$E(d) = E_d(d) + \lambda E_s(d)$$

The 2D problem has many local minima

- Gradient descent doesn't work well

And a large search space

- n x m image w/ k disparities has k^{nm} possible solutions
- Finding the global minimum is NP-hard in general

Stereo as global optimization

Expressing this mathematically

1. Match quality
   – Want each pixel to find a good match in the other image

$$matchCost = \sum_{x,y} \|I(x,y) - J(x + d_{xy}, y)\|$$

2. Smoothness
   – If two pixels are adjacent, they should (usually) move about the same amount

$$smoothnessCost = \sum_{neighbor\ pixels\ p,q} |d_p - d_q|$$

We want to minimize sum of these two cost terms

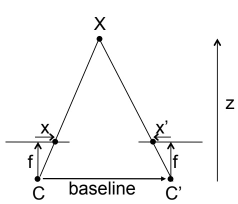
- This is a special type of cost function known as an MRF (Markov Random Field)
  – Effective and fast algorithms have been recently developed:
    » Graph cuts, belief propagation….
    » for more details (and code): http://vision.middlebury.edu/MRF/

Middlebury Stereo Evaluation

http://vision.middlebury.edu/stereo/

38

Depth from disparity



$$disparity = x - x' = \frac{baseline * f}{z}$$

Real-time stereo

Nomad robot searches for meteorites in Antartica
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

Used for robot navigation (and other tasks)

- Several software-based real-time stereo techniques have been developed (most based on simple discrete search)

Stereo reconstruction pipeline

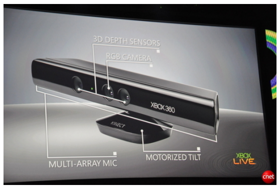
Steps

- Calibrate cameras
- Rectify images
- Compute disparity
- Estimate depth

What will cause errors?

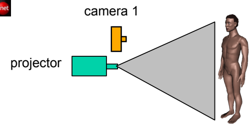
- Camera calibration errors
- Poor image resolution
- Occlusions
- Violations of brightness constancy (specular reflections)
- Large motions
- Low-contrast image regions

Active stereo with structured light



Microsoft's Kinect

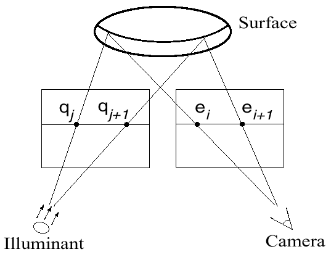
http://www.youtube.com/watch?v=7QrnwoO1-8A



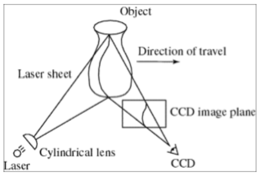
Project "structured" light patterns onto the object

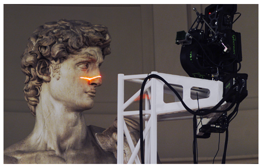
- simplifies the correspondence problem
- can remove one of the cameras (replace with projector)

Active stereo with structured light



Laser scanning





Digital Michelangelo Project
http://graphics.stanford.edu/projects/mich/

Optical triangulation

- Project a single stripe of laser light
- Scan it across the surface of the object
- This is a very precise version of structured light scanning

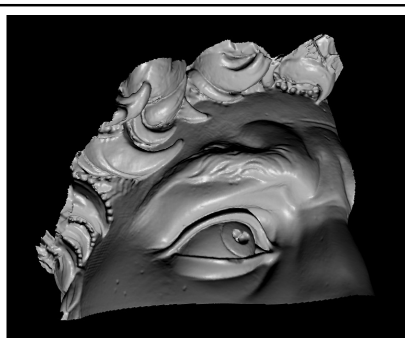
Laser scanned models

The Digital Michelangelo Project, Levoy et al.

Laser scanned models

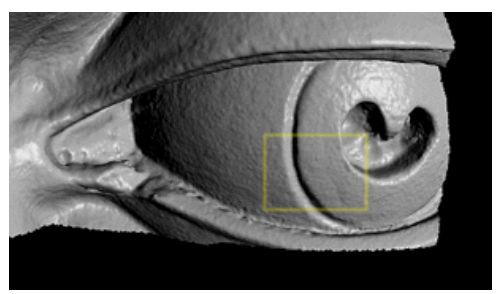
The Digital Michelangelo Project, Levoy et al.

Laser scanned models



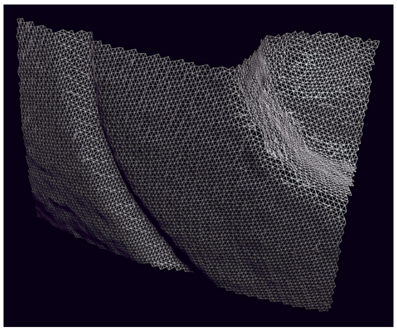
The Digital Michelangelo Project, Levoy et al.

Laser scanned models



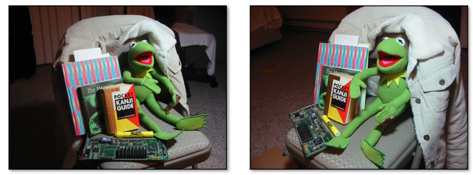
The Digital Michelangelo Project, Levoy et al.

Laser scanned models



The Digital Michelangelo Project, Levoy et al.

Estimating **F**



- If we don't know $\mathbf{K}_1$, $\mathbf{K}_2$, $\mathbf{R}$, or $\mathbf{t}$, can we estimate **F** for two images?
- Yes, given enough correspondences

Estimating F – 8-point algorithm

- The fundamental matrix F is defined by

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$$

for any pair of matches x and x' in two images.

- Let x=$(u,v,1)^{\mathrm{T}}$ and x'=$(u',v',1)^{\mathrm{T}}$, $\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$

each match gives a linear equation

$$uu'f_{11} + vu'f_{12} + u'f_{13} + uv'f_{21} + vv'f_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0$$

8-point algorithm

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

- In reality, instead of solving $\mathbf{Af} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$, least eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

8-point algorithm – Problem?

- **F** should have rank 2
- To enforce that **F** is of rank 2, F is replaced by F' that minimizes $\|\mathbf{F}-\mathbf{F}'\|$ subject to the rank constraint.
- This is achieved by SVD. Let $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$, where

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}, \text{ let } \Sigma' = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

then $\mathbf{F}' = \mathbf{U}\Sigma'\mathbf{V}^{\mathrm{T}}$ is the solution.

8-point algorithm

```
% Build the constraint matrix
   A = [x2(1,:)'.*x1(1,:)'   x2(1,:)'.*x1(2,:)'  x2(1,:)' ...
        x2(2,:)'.*x1(1,:)'   x2(2,:)'.*x1(2,:)'  x2(2,:)' ...
        x1(1,:)'             x1(2,:)'            ones(npts,1) ];

   [U,D,V] = svd(A);

% Extract fundamental matrix from the column of V
% corresponding to the smallest singular value.
   F = reshape(V(:,9),3,3)';

% Enforce rank2 constraint
   [U,D,V] = svd(F);
   F = U*diag([D(1,1) D(2,2) 0])*V';
```

8-point algorithm

- Pros: it is linear, easy to implement and fast
- Cons: susceptible to noise