CS 532: 3D Computer Vision
3rd Set of Notes

Instructor: Philippos Mordohai
Webpage: www.cs.stevens.edu/~mordohai
E-mail: Philippos.Mordohai@stevens.edu
Office: Lieb 215

Lecture Outline

- Fundamental Matrix estimation
- Binocular Stereo
  - Matching criteria

Based on slides by R. Hartley, A. Zisserman, M. Pollefeys, R. Szeliski and P. Fua

Projective Transformation and Invariance

$$\hat{x} = Hx,\ \hat{x}' = H'x' \Rightarrow \hat{F} = H'^{-T} F H^{-1}$$

F invariant to transformations of projective 3-space

$$x = PX = (PH)(H^{-1}X) = \hat{P}\hat{X}$$
$$x' = P'X = (P'H)(H^{-1}X) = \hat{P}'\hat{X}$$

$(P, P') \mapsto F$ unique

$F \mapsto (P, P')$ not unique

canonical form

$$P = [I \mid 0] \qquad F = [m]_\times M \qquad \left(F = [e']_\times P'P^{+}\right)$$
$$P' = [M \mid m]$$

The Projective Reconstruction Theorem

If a set of point correspondences in two views determine the fundamental matrix uniquely, then the scene and cameras may be reconstructed from these correspondences alone, and any two such reconstructions from these correspondences are projectively equivalent

$$\mathrm{x}_i \leftrightarrow \mathrm{x}'_i \qquad (\mathrm{P}_1, \mathrm{P}_1', \{\mathrm{X}_{1i}\}) \quad (\mathrm{P}_2, \mathrm{P}_2', \{\mathrm{X}_{2i}\})$$

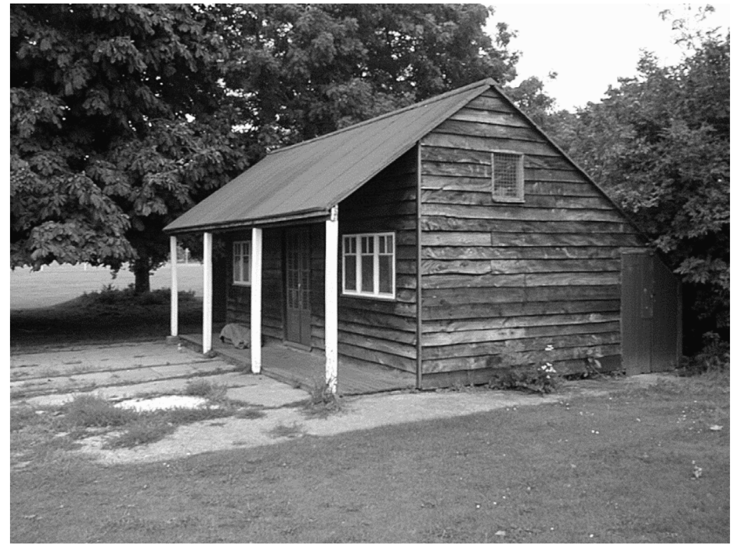
$$\mathrm{P}_2 = \mathrm{P}_1\mathrm{H}^{-1} \quad \mathrm{P}'_2 = \mathrm{P}'_1\mathrm{H}^{-1} \quad \mathrm{X}_{2i} = \mathrm{H}\mathrm{X}_{1i} \qquad (\text{except}: \mathrm{F}\mathrm{x}_i = \mathrm{x}'_i\mathrm{F} = 0)$$

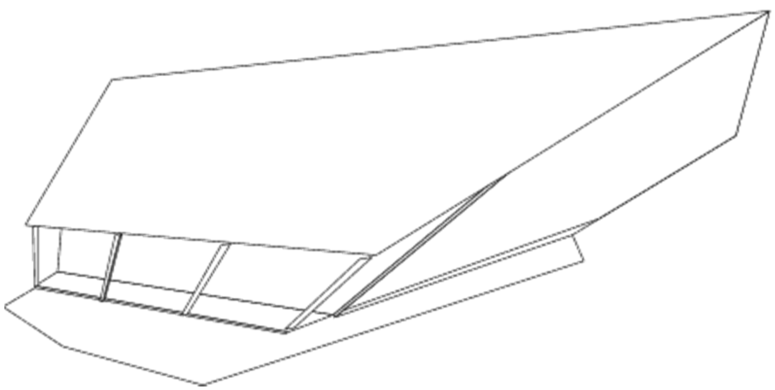
$$\mathrm{P}_2(\mathrm{H}\mathrm{X}_{1i}) = \mathrm{P}_1\mathrm{H}^{-1}\mathrm{H}\mathrm{X}_{1i} = \mathrm{P}_1\mathrm{X}_{1i} = \mathrm{x}_i = \mathrm{P}_2\mathrm{X}_{2i}$$

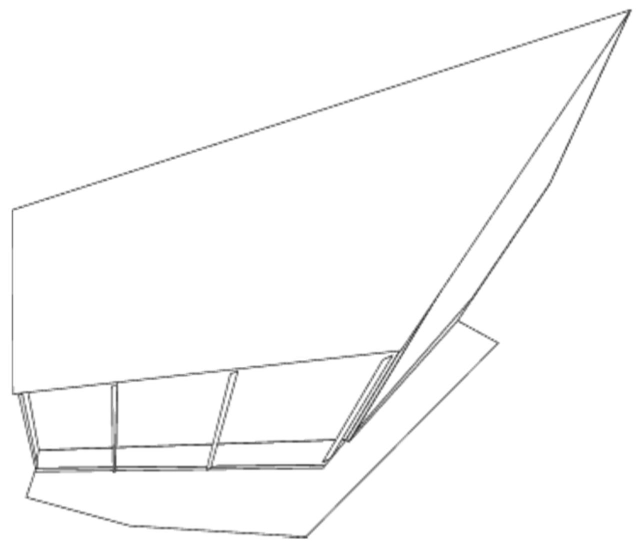
$\Rightarrow$ along same ray of P_2, idem for P'_2

two possibilities: $X_{2i}=HX_{1i}$, or points along baseline

key result:
allows reconstruction from pair of uncalibrated images







Stratified Reconstruction

(i) Projective reconstruction
(ii) Affine reconstruction
(iii) Metric reconstruction

Out of scope of CS 532

The Essential Matrix

~fundamental matrix for calibrated cameras (remove K)

$$E = [t]_\times R = R[R^T t]_\times$$

$$\hat{x}'^T E \hat{x} = 0 \qquad \left(\hat{x} = K^{-1}x;\ \hat{x}' = K^{-1}x'\right)$$

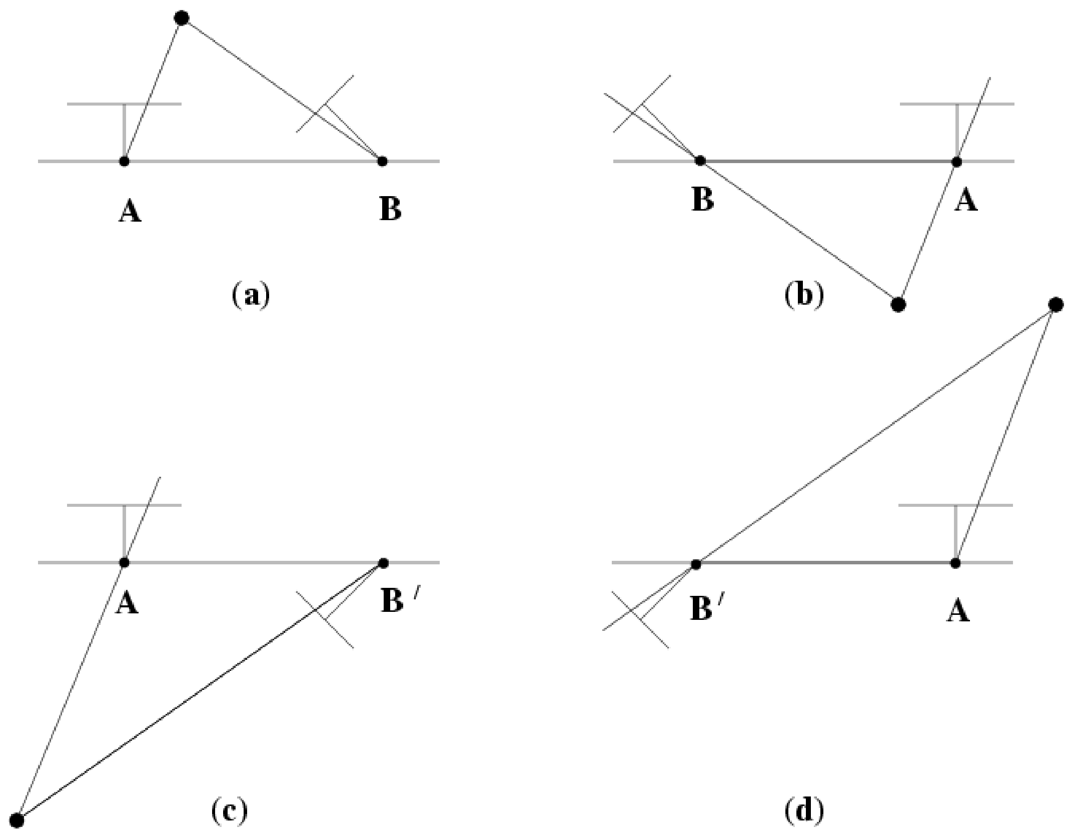
$$E = K'^T F K$$

5 d.o.f. (3 for R; 2 for t up to scale)

E is an essential matrix if and only if two singular values are equal (and the third=0)

$$E = U\mathrm{diag}(1,1,0)V^T$$

Four Possible Solutions from E



Given E and setting the first camera matrix P = [I | 0], there are four possible solutions for P’ (only one solution, however, where a reconstructed point is in front of both cameras)

Fundamental Matrix Estimation

Epipolar Geometry: Basic Equation

$$\mathrm{x'}^{\mathrm{T}}\,\mathrm{Fx} = 0$$

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

separate known from unknown

$$[x'x, x'y, x', y'x, y'y, y', x, y, 1][f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}]^{\mathrm{T}} = 0$$

(data) (unknowns)

(linear)

$$\begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathrm{f} = 0$$

$$\mathrm{Af} = 0$$

The Singularity Constraint

$$e'^T F = 0 \qquad Fe = 0 \qquad \det F = 0 \qquad \operatorname{rank} F = 2$$

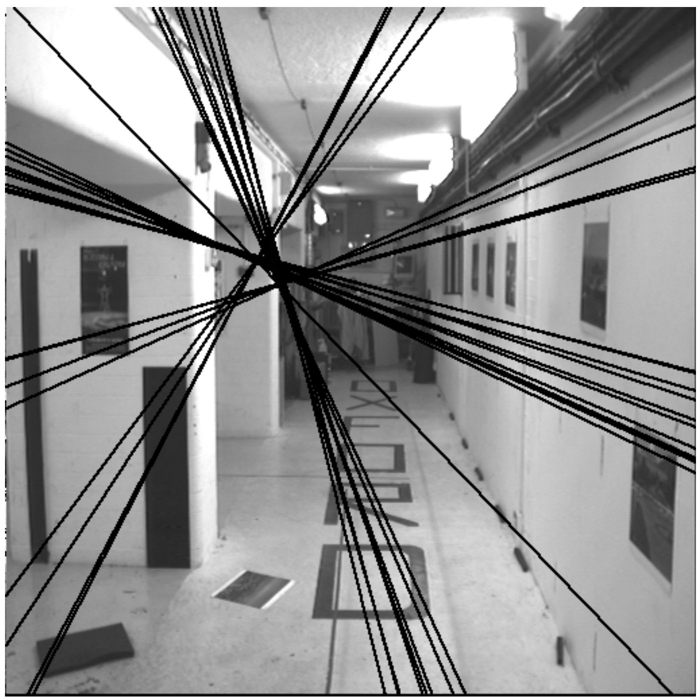
SVD from linearly computed F matrix (rank 3)

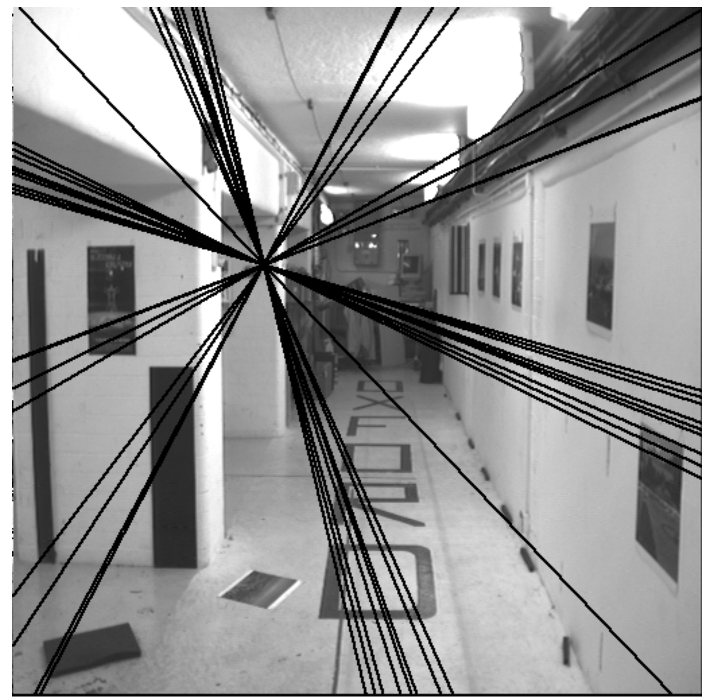
$$F = U \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} V^T = U_1 \sigma_1 V_1^T + U_2 \sigma_2 V_2^T + U_3 \sigma_3 V_3^T$$

Compute closest rank-2 approximation $\quad \min \|F - F'\|_F$

$$F' = U \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & 0 \end{bmatrix} V^T = U_1 \sigma_1 V_1^T + U_2 \sigma_2 V_2^T$$

The Singularity Constraint





The Minimum Case - 7 Point Correspondences

$$\begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_7 x_7 & x'_7 y_7 & x'_7 & y'_7 x_7 & y'_7 y_7 & y'_7 & x_7 & y_7 & 1 \end{bmatrix} \mathrm{f} = 0$$

$$\mathrm{A} = \mathrm{U}_{7\mathrm{x}7} \mathrm{diag}\left(\sigma_1, \ldots, \sigma_7, 0, 0\right) \mathrm{V}_{9\mathrm{x}9}{}^{\mathrm{T}}$$

$$\Rightarrow \ \mathrm{A}[\mathrm{V}_8 \mathrm{V}_9] = 0_{9\mathrm{x}2}$$

$$\mathrm{x}_i{}^{\mathrm{T}} (\mathrm{F}_1 + \lambda \mathrm{F}_2) \mathrm{x}_i = 0, \forall i = 1 \ldots 7$$

One parameter family of solutions – results in 1 or 3 real solutions

but $\mathrm{F}_1 + \lambda \mathrm{F}_2$ not automatically rank 2

The NOT Normalized 8-point Algorithm

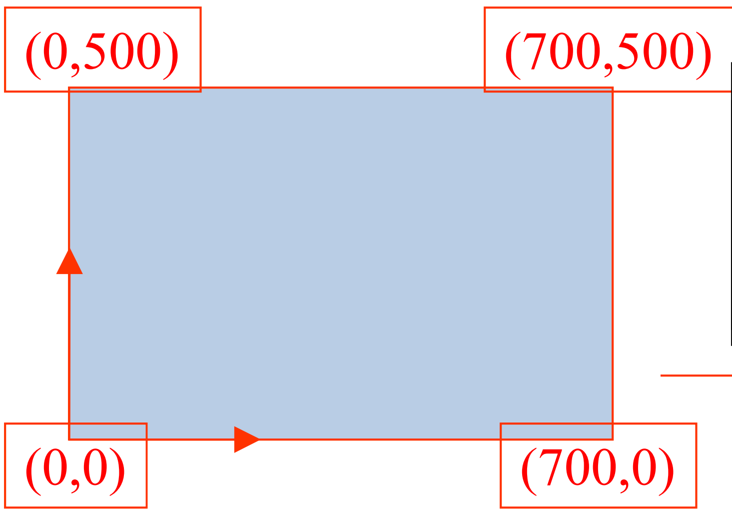
$$\begin{bmatrix} x_1x_1' & y_1x_1' & x_1' & x_1y_1' & y_1y_1' & y_1' & x_1 & y_1 & 1 \\ x_2x_2' & y_2x_2' & x_2' & x_2y_2' & y_2y_2' & y_2' & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx_n' & y_nx_n' & x_n' & x_ny_n' & y_ny_n' & y_n' & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

$$\begin{array}{ccccccccc} \sim 10000 & \sim 10000 & \sim 100 & \sim 10000 & \sim 10000 & \sim 100 & \sim 100 & \sim 100 & 1 \end{array}$$

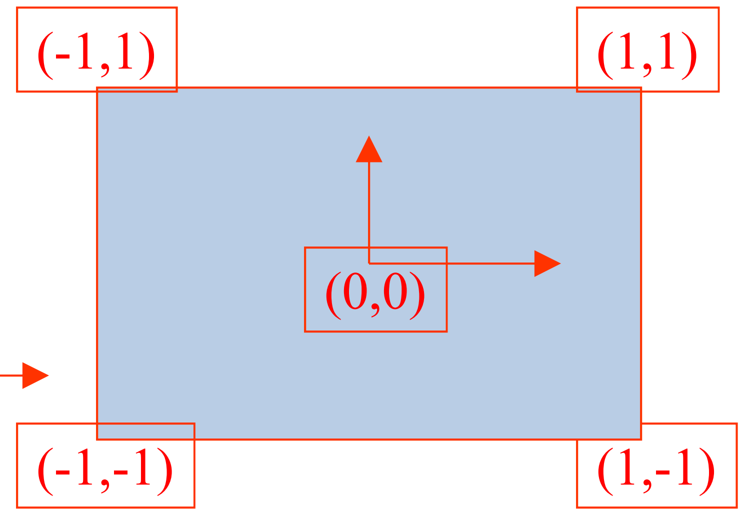
! Orders of magnitude difference between column of data matrix
$\rightarrow$ least-squares yields poor results

The Normalized 8-point Algorithm

Transform image to [-1,1]x[-1,1]

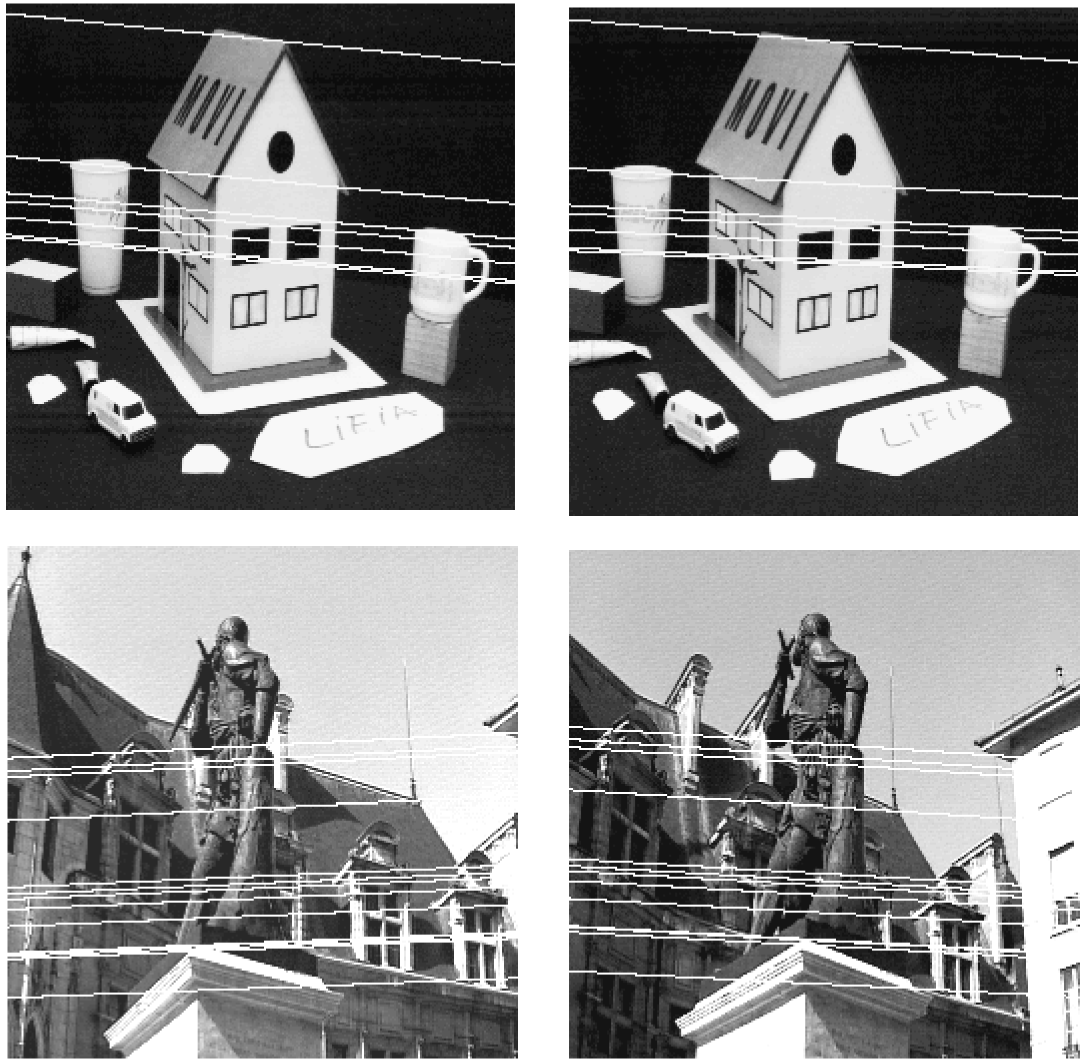


$$\begin{bmatrix} \frac{2}{700} & 0 & -1 \\ & \frac{2}{500} & -1 \\ & & 1 \end{bmatrix}$$



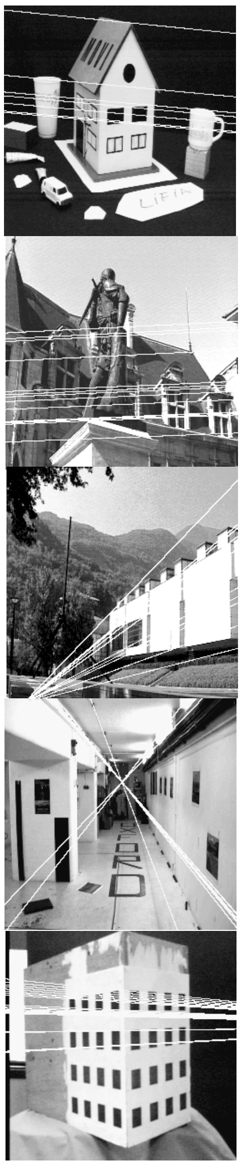
normalized least squares yields good results

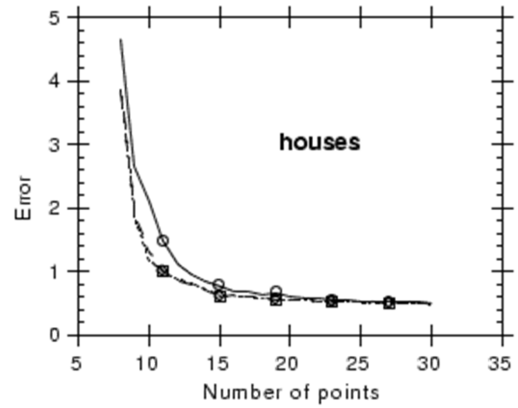
Some Experiments

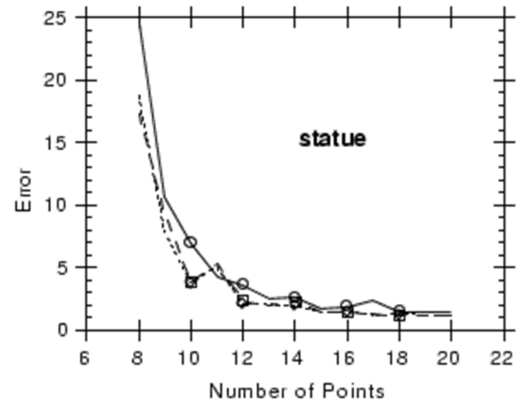


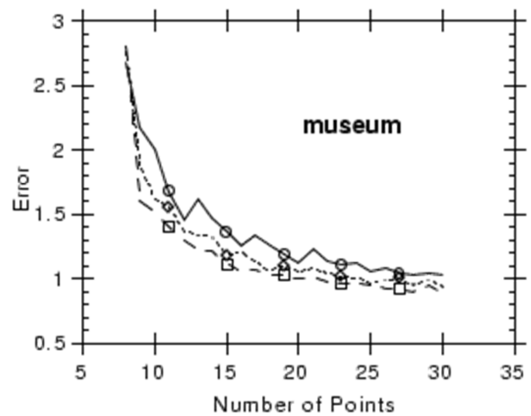
Some Experiments

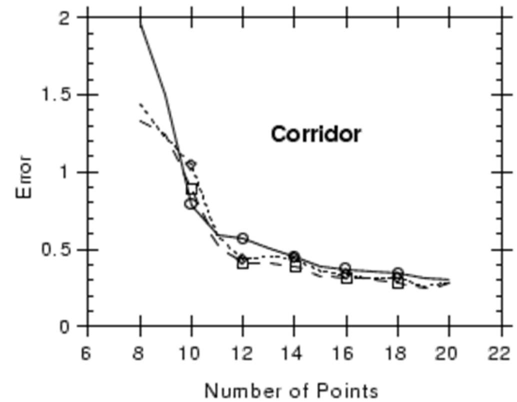
Some Experiments

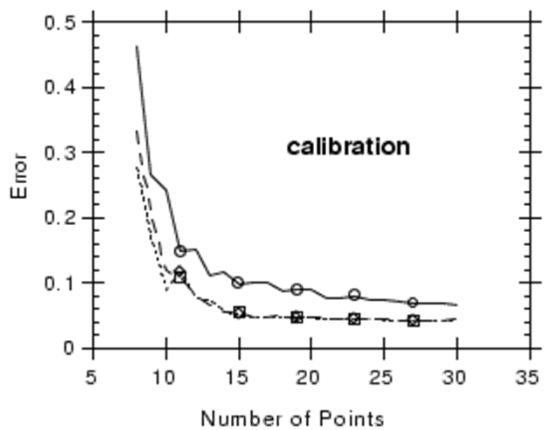












Residual error:

$$\sum_i d\left(\mathrm{x}'_i, F\mathrm{x}_i\right)^2 + d\left(\mathrm{x}_i, \mathrm{F}^{\mathrm{T}}\mathrm{x}'_i\right)^2$$

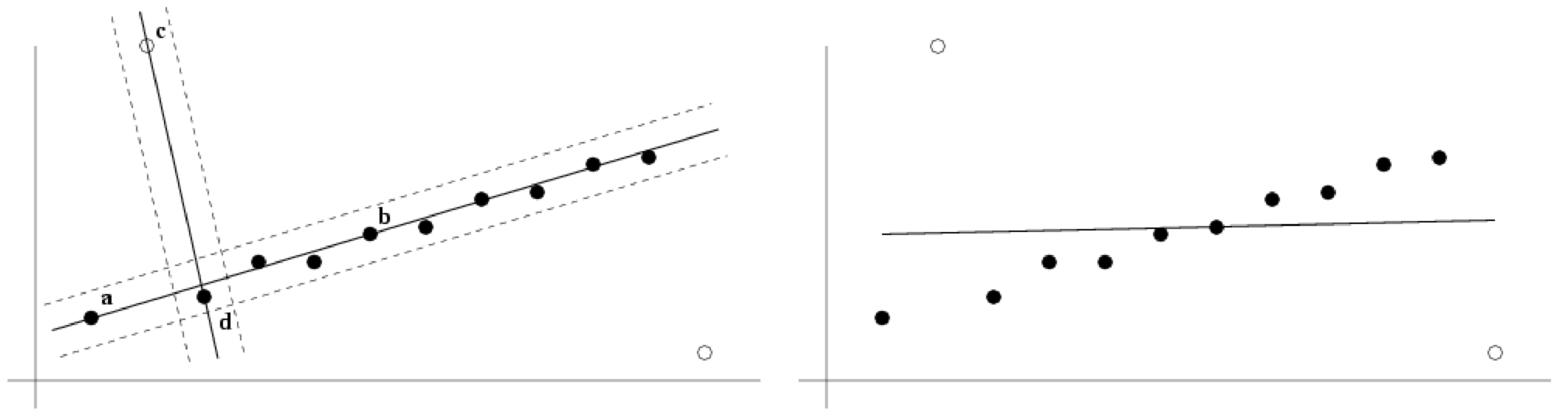
(for all points!)

Recommendations:

1. Do not use unnormalized algorithms
2. Quick and easy to implement: 8-point normalized
3. Better: enforce rank-2 constraint during minimization
4. Best: Maximum Likelihood Estimation
   (minimal parameterization, sparse implementation)

Robust Estimation

- What if set of matches contains gross outliers?



RANSAC

Objective

Robust fit of model to data set S which contains outliers

Algorithm

(i) Randomly select a sample of s data points from S and instantiate the model from this subset.

(ii) Determine the set of data points S_i which are within a distance threshold t of the model. The set S_i is the consensus set of samples and defines the inliers of S.

(iii) If the subset of S_i is greater than some threshold T, re-estimate the model using all the points in S_i and terminate

(iv) If the size of S_i is less than T, select a new subset and repeat the above.

(v) After N trials the largest consensus set S_i is selected, and the model is re-estimated using all the points in the subset S_i

How many samples?

- Choose t so probability for inlier is α (e.g. 0.9)
  - Or empirically
- Choose N so that, with probability *p*, at least one random sample is free from outliers. e.g. *p* =0.99

$$\left(1-(1-e)^s\right)^N = 1-p$$

$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Acceptable consensus set?

- Typically, terminate when inlier ratio reaches expected ratio of inliers

$$T = (1 - e)n$$

Adaptively determining the number of samples

e is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield e=0.2

- $N=\infty$, *sample_count* =0
- While N > *sample_count* repeat
  - Choose a sample and count the number of inliers
  - Set e=1-(number of inliers)/(total number of points)
  - Recompute N from e
  - Increment the *sample_count* by 1
- Terminate

$$N = \log(1-p) / \log\left(1-(1-e)^s\right)$$

RANSAC for F Estimation

Step 1. Extract features

Step 2. Compute a set of potential matches

Step 3. do

Step 3.1 select minimal sample (i.e. 7 matches)

Step 3.2 compute solution(s) for F

(generate hypothesis)

Step 3.3 determine inliers (verify hypothesis)

until p(*#inliers*,*#samples*)>95% or 99%

Step 4. Compute F based on all inliers

Step 5. Look for additional matches

Step 6. Refine F based on all correct matches

$$p = 1 - \left(1 - \left(\frac{\#inliers}{\#matches}\right)^7\right)^{\#samples}$$

#inliers	90%	80%	70%	60%	50%
#samples	5	13	35	106	382

Finding more matches

restrict search range to neighborhood of epipolar line
(±1.5 pixels)
relax disparity restriction (along epipolar line)

Degenerate Cases

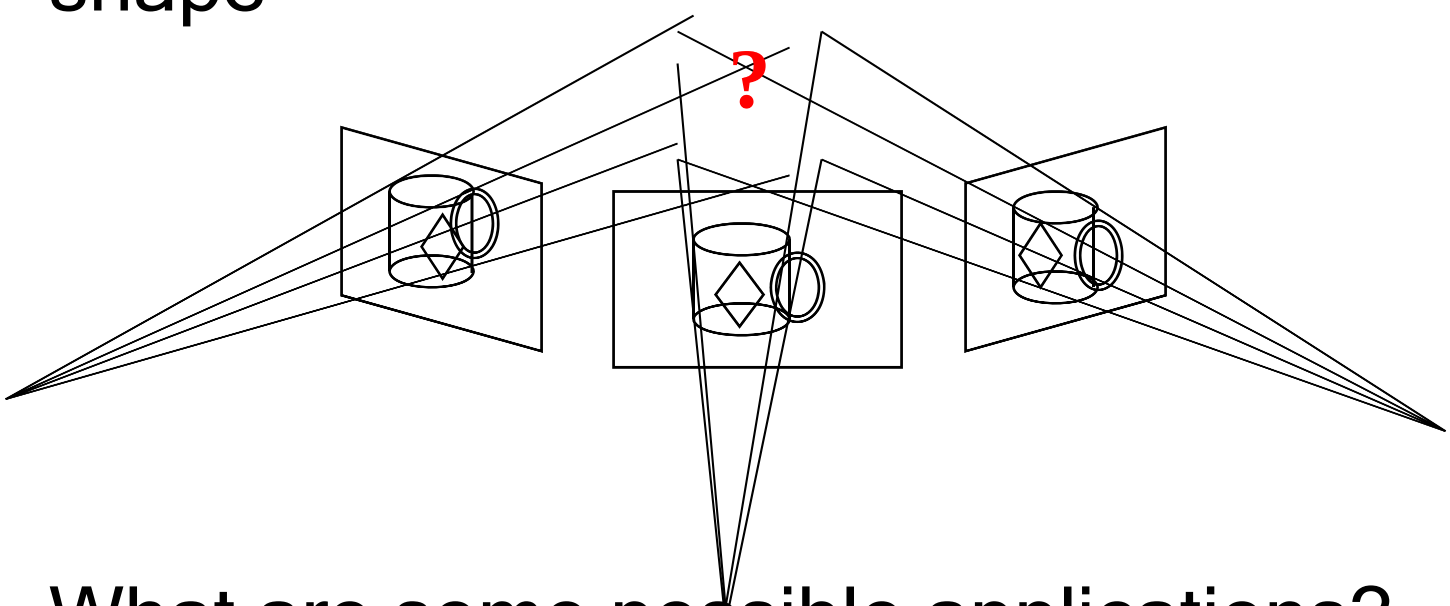
- Degenerate cases
  - Planar scene
  - Pure rotation
- No unique solution
  - Remaining DOF filled by noise
  - Use simpler model (e.g. homography)
- Model selection (Torr et al., ICCV´98, Kanatani, Akaike)
  - Compare H and F according to expected residual error (compensate for model complexity)

Stereo Matching

Slides by Rick Szeliski, Pascal Fua and P. Mordohai

Stereo Matching

- Given two or more images of the same scene or object, compute a representation of its shape



- What are some possible applications?

Stereo Matching

- Given two or more images of the same scene or object, compute a representation of its shape

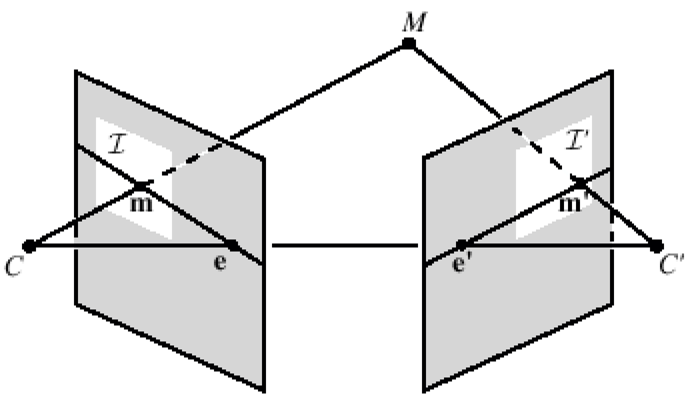
- What are some possible representations?
  - depth maps
  - volumetric models
  - 3D surface models
  - planar (or offset) layers

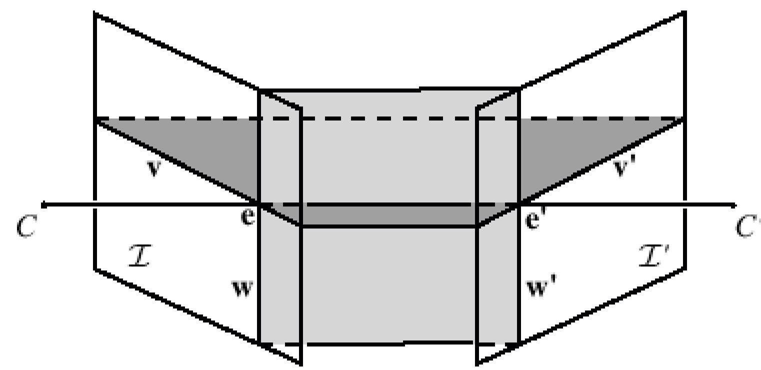
Stereo Matching

- What are some possible algorithms?
  - match "features" and interpolate
  - match edges and interpolate
  - match all pixels with windows (coarse-fine)
  - use optimization:
    - iterative updating
    - dynamic programming
    - energy minimization (regularization, stochastic)
    - graph algorithms

Rectification

- Project each image onto same plane, which is parallel to the baseline
- Resample lines (and shear/stretch) to place lines in correspondence, and minimize distortion

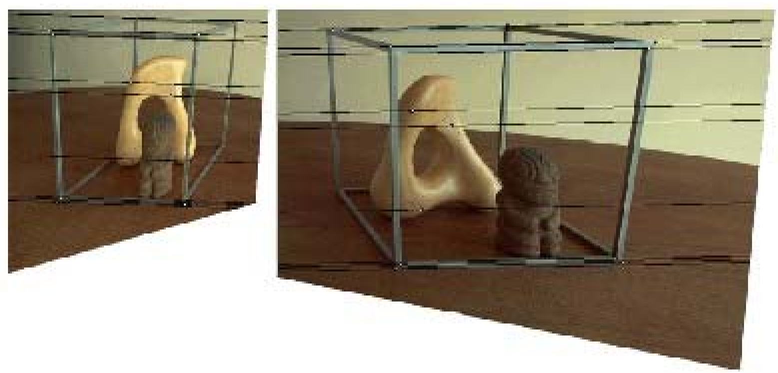




- Take rectification for granted in this course

Rectification

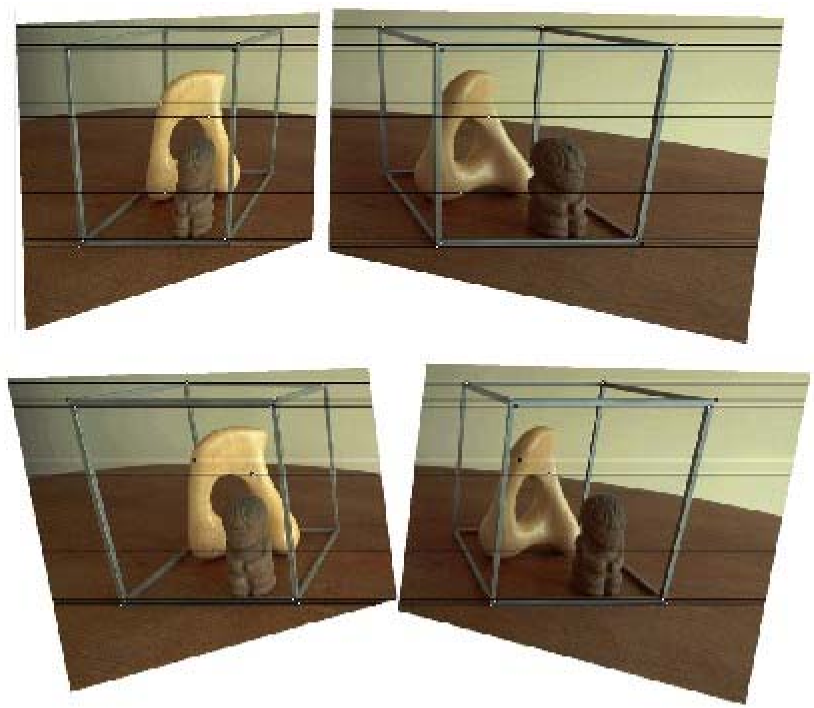
(a) Original image pair overlayed with several epipolar lines.



(b) Image pair transformed by the specialized projective mapping $\mathbf{H}_p$ and $\mathbf{H}'_p$. Note that the epipolar lines are now parallel to each other in each image.

BAD!

Rectification



(c) Image pair transformed by the similarity $\mathbf{H}_r$ and $\mathbf{H}'_r$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).

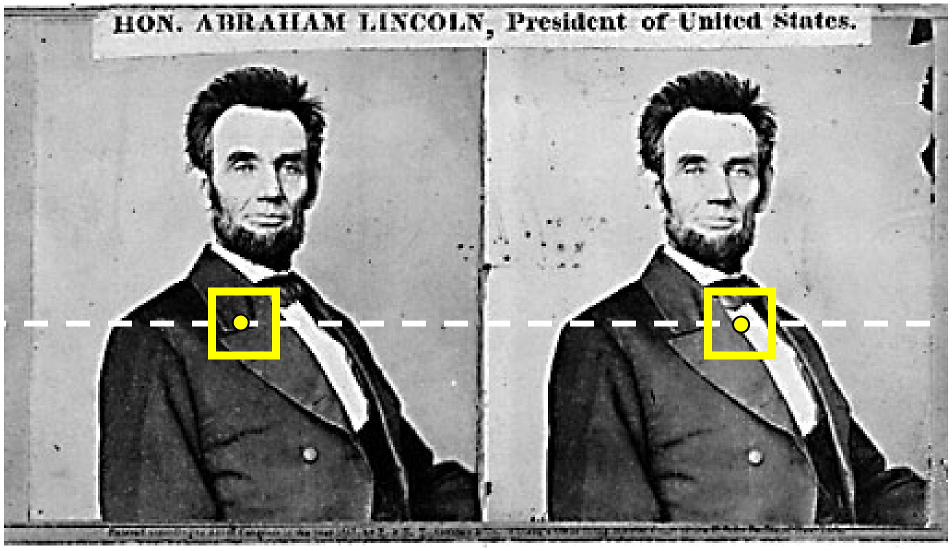
(d) Final image rectification after shearing transform $\mathbf{H}_s$ and $\mathbf{H}'_s$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

GOOD!

Finding Correspondences

- Apply feature matching criterion at *all* pixels simultaneously
- Search only over epipolar lines (many fewer candidate positions)

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***

Disparity

- Disparity d is the difference between the x coordinates of corresponding pixels in the left and right image

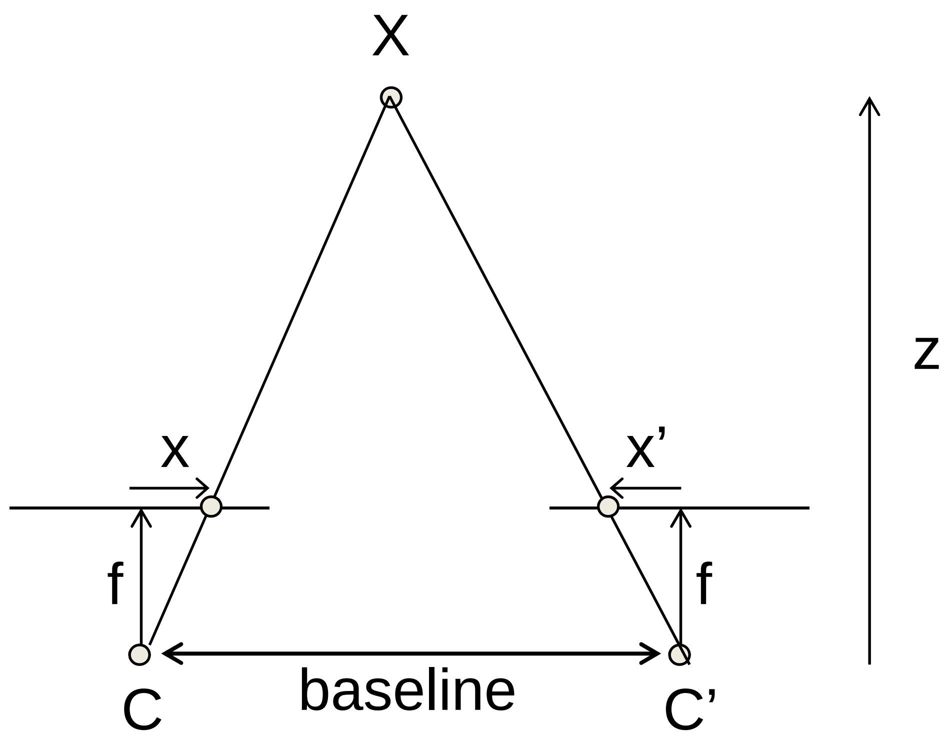
$$d = x_L - x_R$$

- Disparity is inversely proportional to depth

$$Z = \frac{bf}{d}$$

Stereo Reconstruction

$$Z = \frac{bf}{d}$$

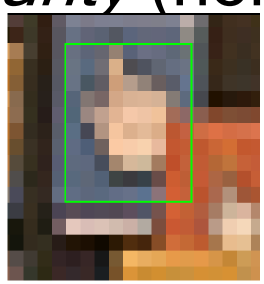
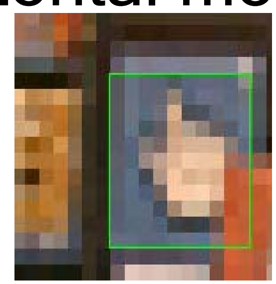


Finding Correspondences

- How do we determine correspondences?
  - *block matching* or *SSD* (sum squared differences)

$$SSD(x, y; d) = \sum_{(x',y') \in \mathrm{N}(x,y)} [\, I_L(x', y') - I_R(x' - d, y')]^2$$

  - *d* is the *disparity* (horizontal motion)

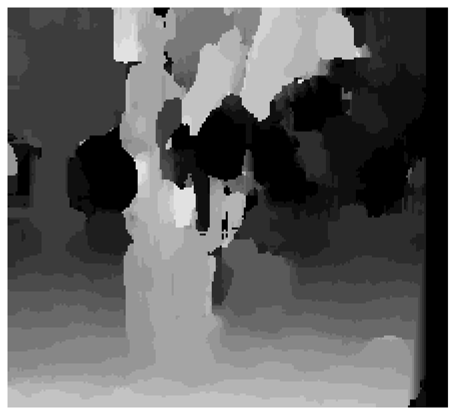
 

- How big should the neighborhood be?

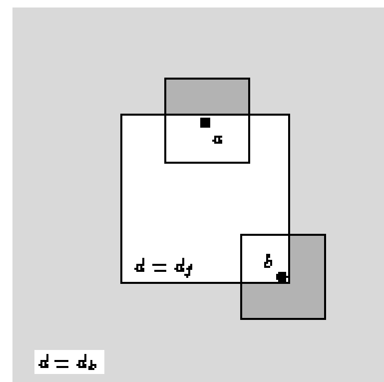
Neighborhood size

- Smaller neighborhood: more details
- Larger neighborhood: fewer isolated mistakes

w = 3



w = 20



Challenges

- Ill-posed inverse problem
  - Recover 3-D structure from 2-D information
- Difficulties
  - Uniform regions
  - Half-occluded pixels
  - Repeated patterns

Pixel Dissimilarity

- Sum of Squared Differences of intensities (SSD)

$$SSD(x, y; d) = \sum_{(x',y') \in \mathrm{N}(x,y)} [I_L(x', y') - I_R(x' - d, y')]^2$$

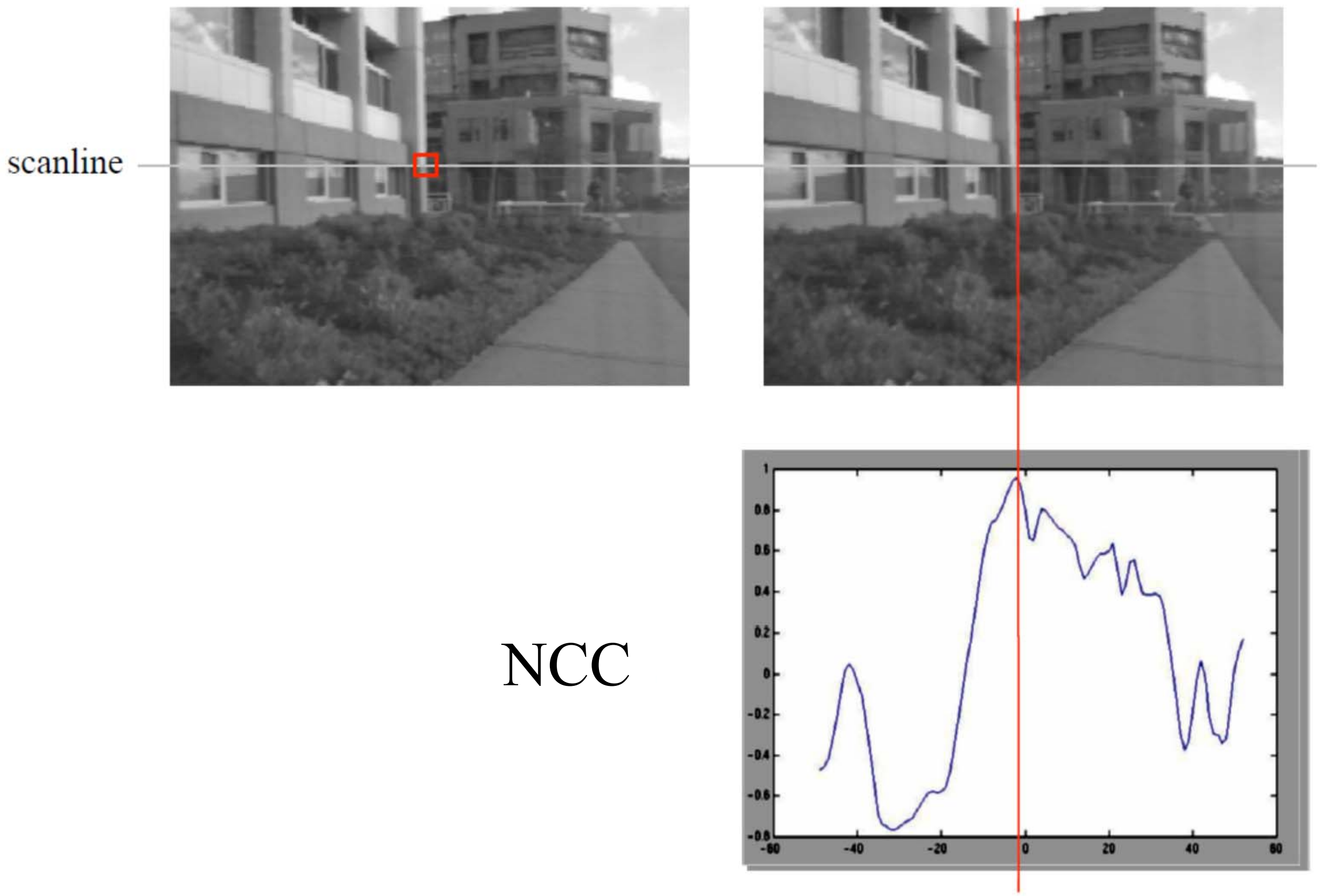
- Sum of Absolute Differences of intensities (SAD)

$$SAD(x, y; d) = \sum_{(x',y') \in \mathrm{N}(x,y)} |I_L(x', y') - I_R(x' - d, y')|$$

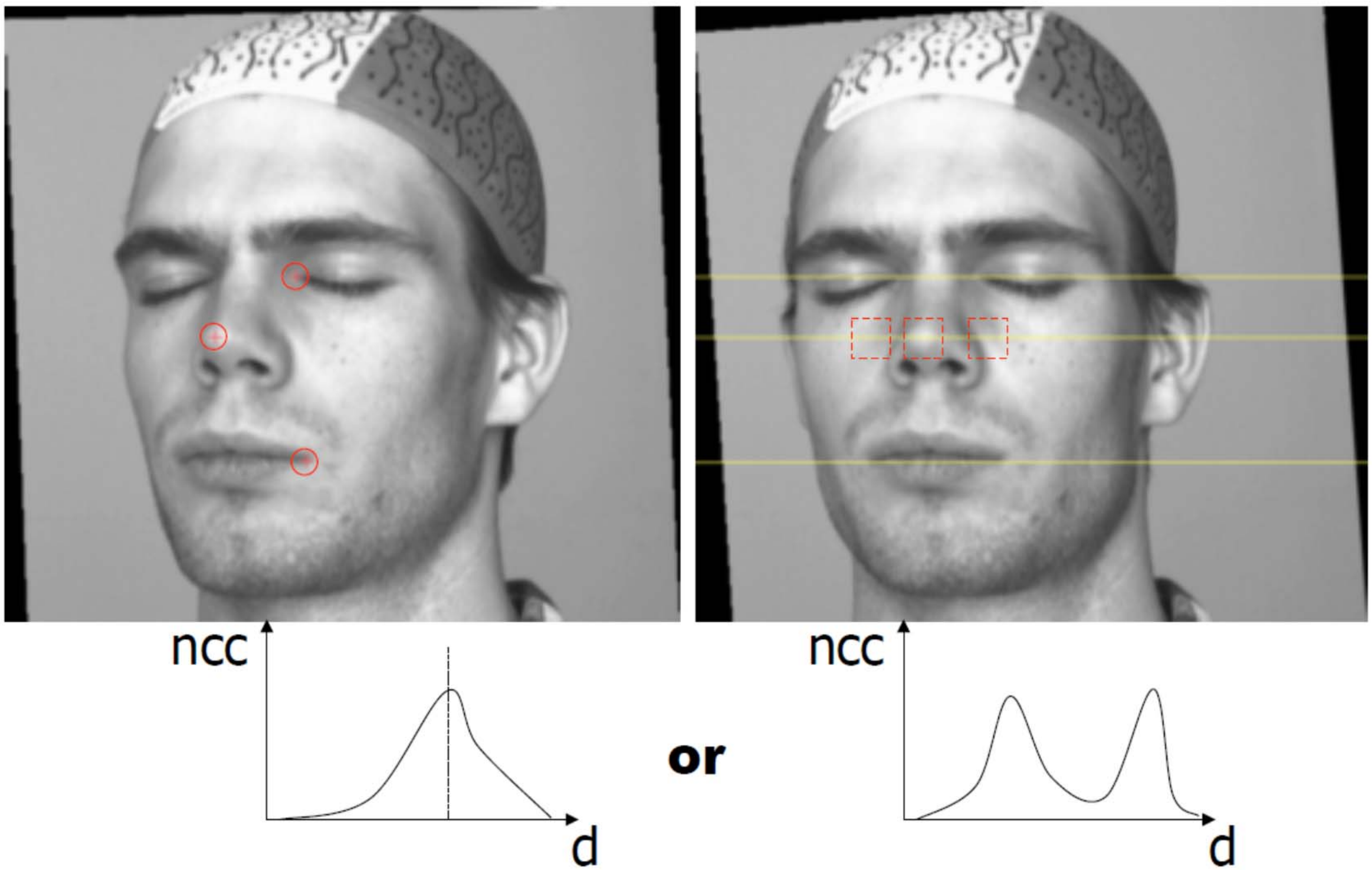
- Zero-mean Normalized Cross-correlation (NCC)

$$NCC(x, y, d) = \frac{\sum_{i \in W} (I_L(x_i, y_i) - \mu_L)(I_R(x_i - d, y_i) - \mu_R)}{\sigma_L \sigma_R}$$

Cost/Score Curve

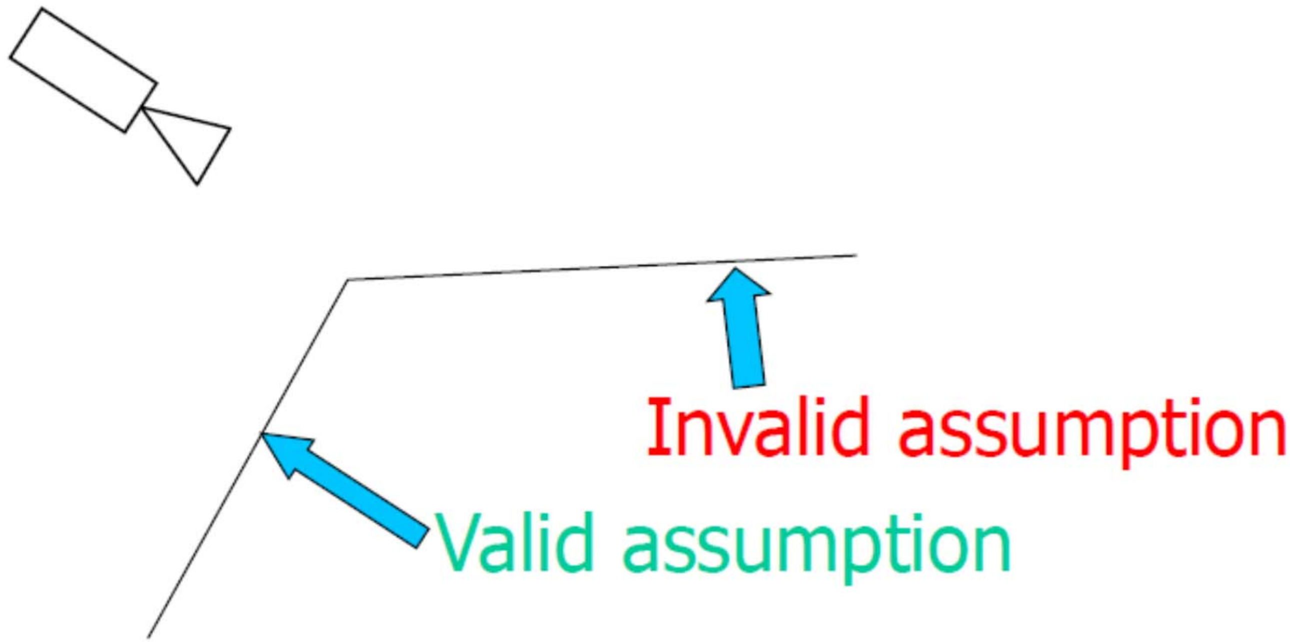


Cost/Score Curve



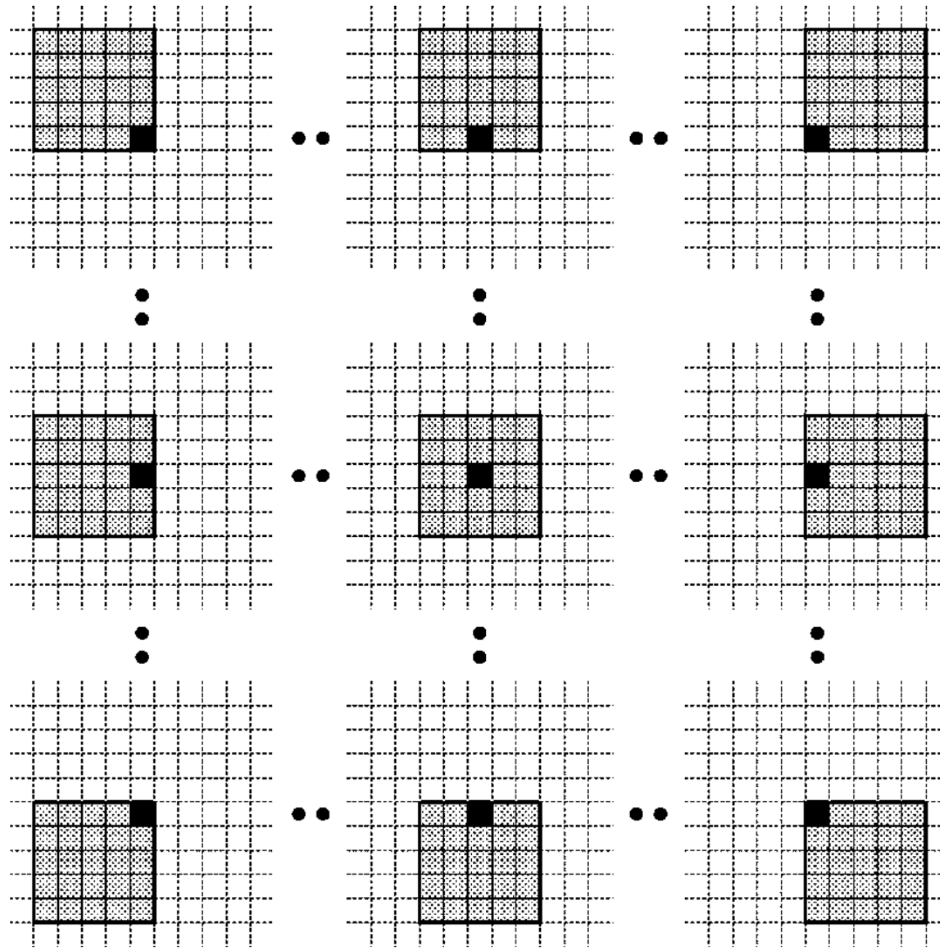
Fronto-Parallel Assumption

- The disparity is assumed to be the same in the entire matching window
  - equivalent to assuming constant depth



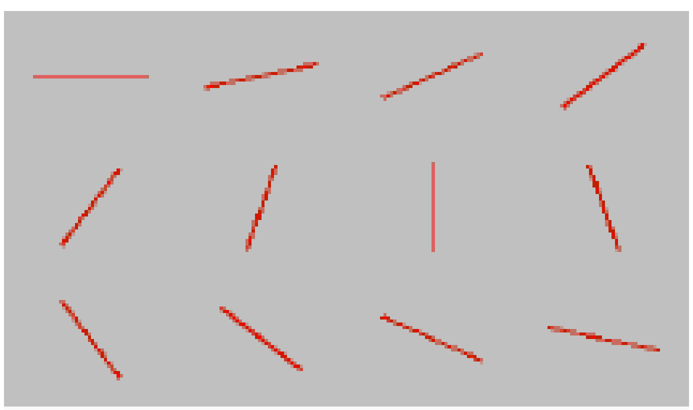
Shiftable Windows

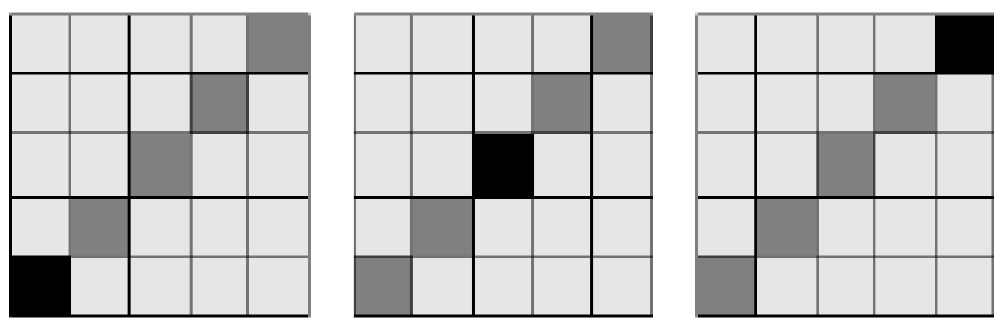
- Avoid having using matching windows that straddle two surfaces
  - Disparity will not be constant for all pixels
- Shift the window around the reference pixel
  - Keep the one with min cost (max NCC)



Rod-shaped Filters

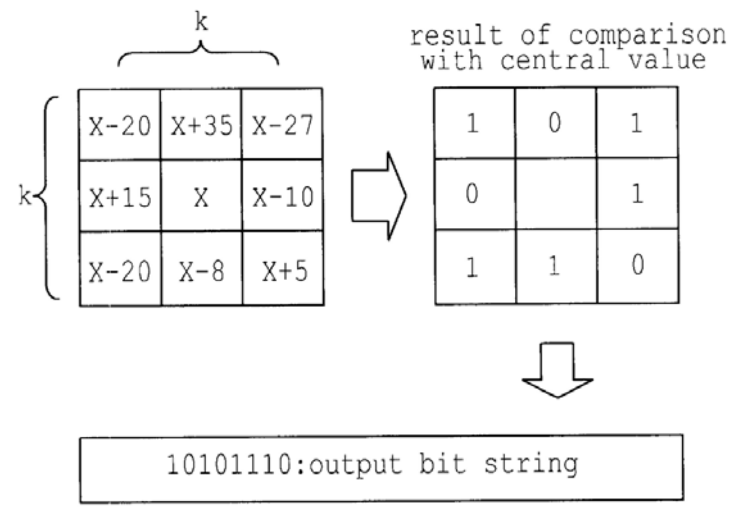
- Instead of square windows aggregate cost in rod-shaped shiftable windows
- Search for one that minimizes the cost (assume that it is an iso-disparity curve)





Alternative Dissimilarity Measures

- Rank and Census transforms
- Rank transform:
  - Define window containing R pixels around each pixel
  - Count the number of pixels with lower intensities than center pixel in the window
  - Replace intensity with rank (0..R-1)
  - Compute SAD on rank-transformed images
- Census transform:
  - Use bit string, defined by neighbors, instead of scalar rank
- Robust against illumination changes



Locally Adaptive Support

Apply weights to contributions of neighboring pixels according to similarity and proximity

(a) left support window (b) right support window (c) color difference between (a) and (b)

Locally Adaptive Support

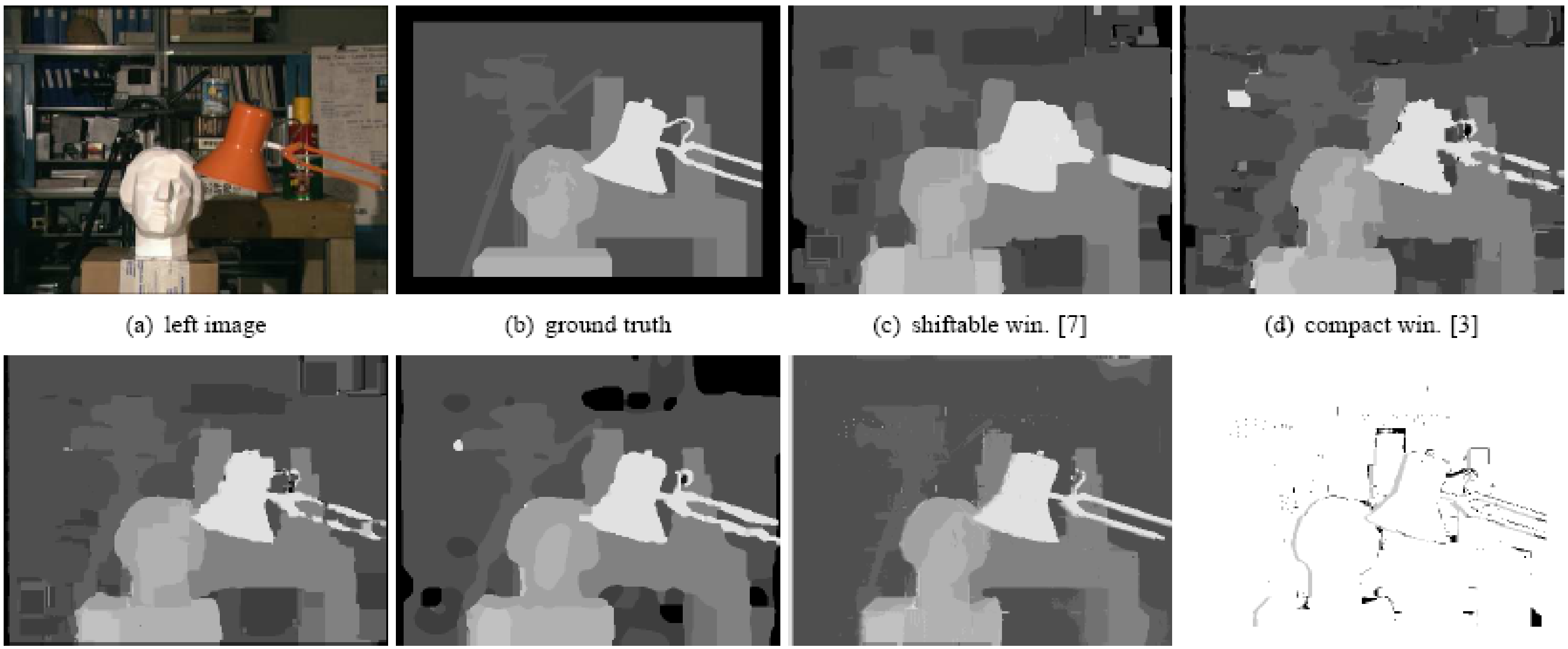
- Similarity in CIE Lab color space:

$$\Delta c_{pq} = \sqrt{(L_p - L_q)^2 + (a_p - a_q)^2 + (b_p - b_q)^2}$$

- Proximity: Euclidean distance

- Weights: $w(p,q) = k \cdot \exp\left(-(\frac{\Delta c_{pq}}{\gamma_c} + \frac{\Delta g_{pq}}{\gamma_p})\right)$

Locally Adaptive Support: Results



(a) left image (b) ground truth (c) shiftable win. [7] (d) compact win. [3]

(e) variable win. [4] (f) Bay. diff. [19] (g) our result (h) bad pixels (error $>$ 1)