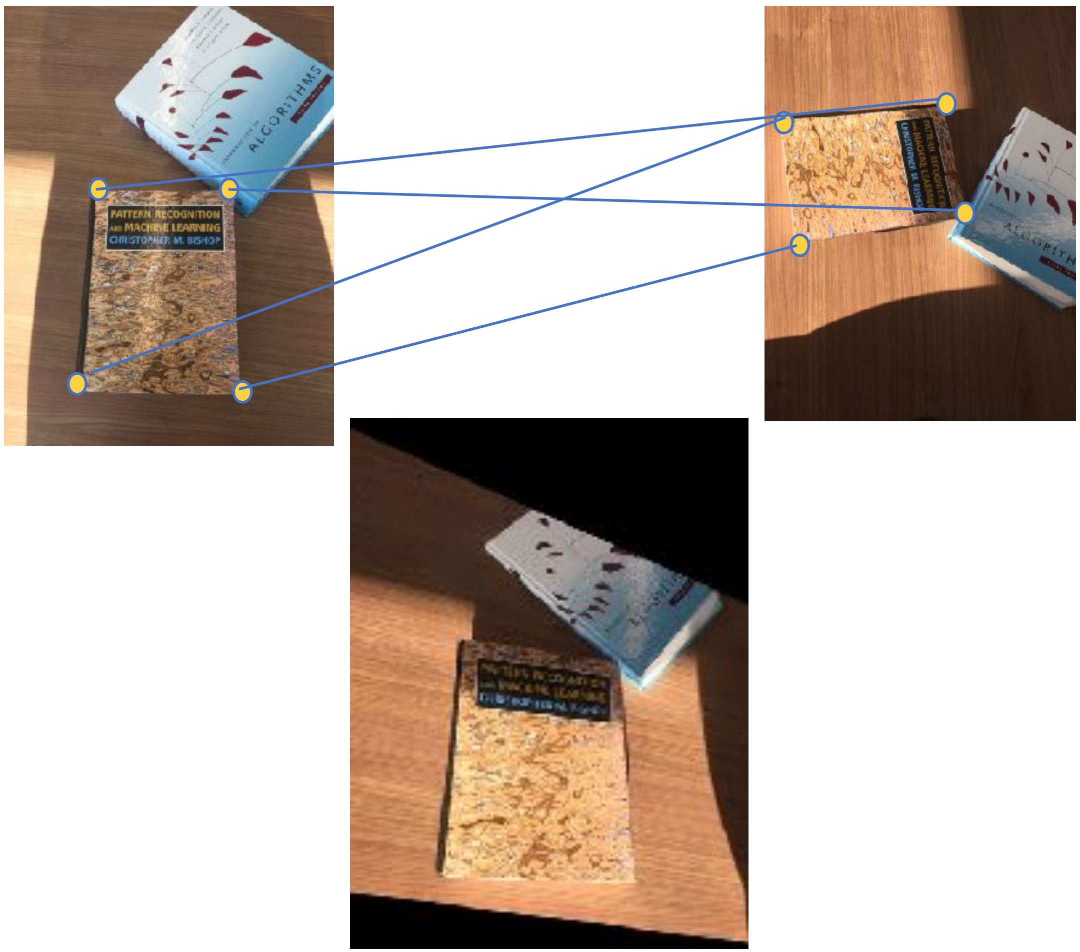
Homography

Homography


ALGORITHMS
PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP


PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP

Homography



ALGORITHMS
PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP

2D homography (projective transformation)

Definition
:

A 2D *homography* is an invertible mapping h from P^2 to itself such that three points x_1,x_2,x_3 lie on the same line if and only if $h(x_1)$,$h(x_2)$,$h(x_3)$ do.

Line preservin
g

Theorem:

A mapping h: $P^2 \rightarrow P^2$ is a homography if and only if there exist a non-singular 3x3 matrix **H** such that for any point in P^2 represented by a vector x it is true that $h(x)=\mathbf{H}x$

Definition: Homography

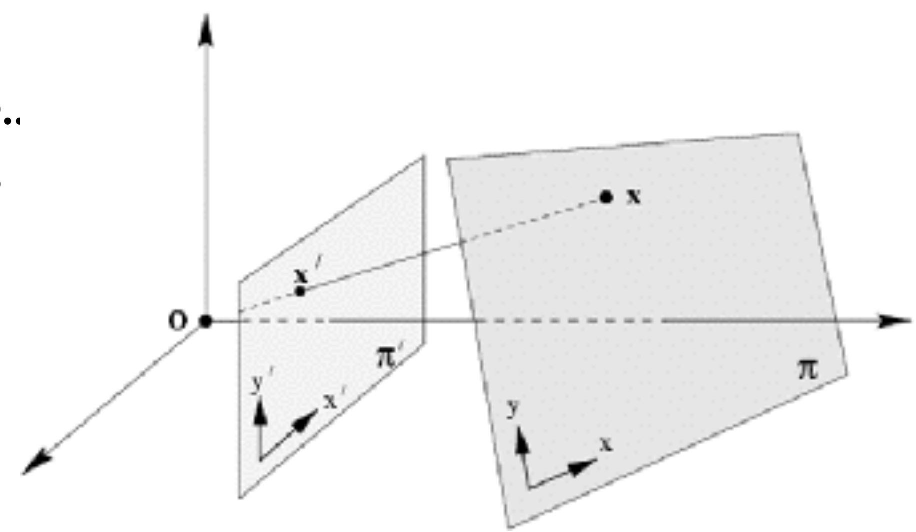
$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

Homography=projective transformation=projectivity=collineation

General homography

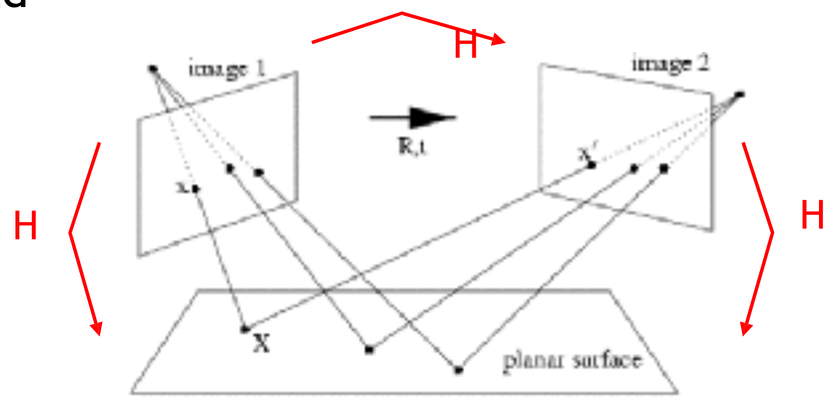
- Note: homographies are not restricted to P^2
- General definition:
  A homography is a non-singular, line preserving, projective mapping h: $P^n \rightarrow P^n$.
  It is represented by a square (n + 1)-dim matrix
  with $(n + 1)^2-1$ DOF

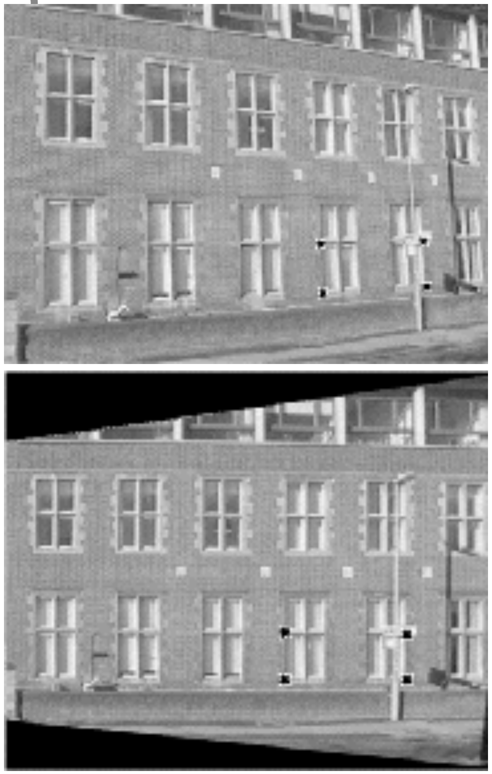
- Now back to the 2D case..
- Mapping between planes



Homographies in Computer vision

Rotating/translating camera, planar world



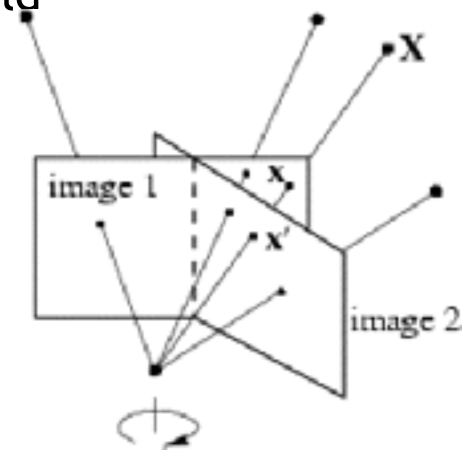


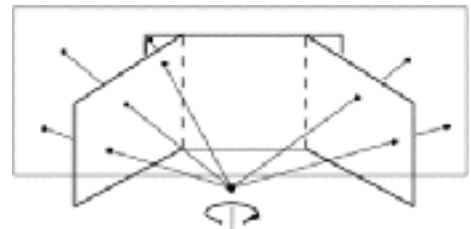
$$(x, y, 1)^T = x \propto PX = K\,[r_1 r_2 \cancel{r_3} t] \begin{pmatrix} X \\ Y \\ \cancel{0} \\ 1 \end{pmatrix} = H \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

What happens to the P-matrix, if Z is assumed zero?

Homographies in Computer vision

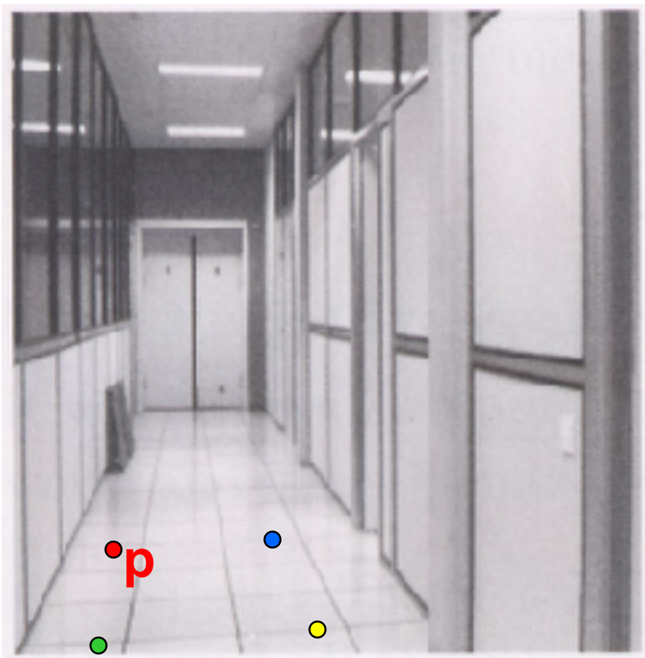
Rotating camera, arbitrary world

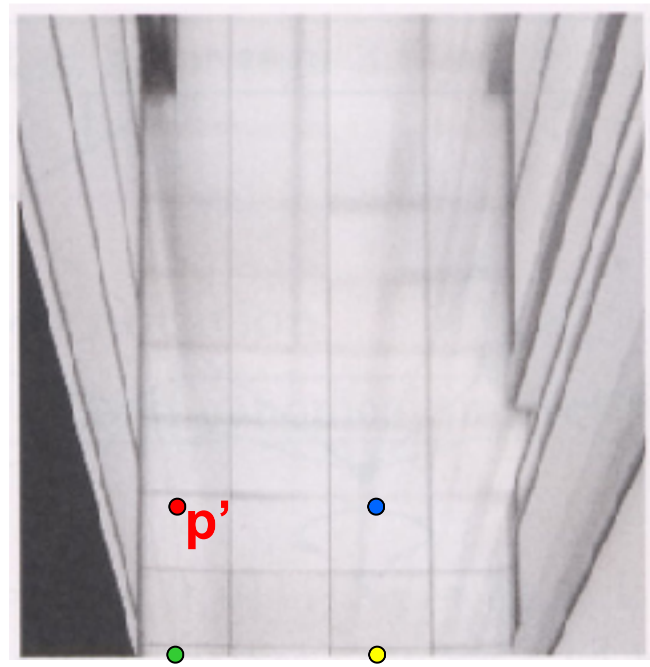




$$(x, y, 1)^T = x \propto PX = K\begin{pmatrix} r_1 & r_2 & r_3 & \cancel{t} \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ \cancel{1} \end{pmatrix} \propto KRK^{-1}x' = Hx'$$

What happens to the P-matrix, if t is assumed zero?





To unwarp (rectify) an image

- solve for homography **H** given **p** and **p’**
- solve equations of the form: w**p’** = **Hp**
  - linear in unknowns: w and coefficients of **H**
  - H is defined up to an arbitrary scale factor
  - how many points are necessary to solve for **H**?

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\begin{aligned} x'_i &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y'_i &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies

$$
\underbrace{\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & y'_1 x_1 & y'_1 y_1 & y'_1 \\
 & & & & \vdots & & & & \\
x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\
0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n
\end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 9}}
\underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\mathbf{h} \\ 9}}
=
\underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\substack{\mathbf{0} \\ 2n}}
$$

Linear least squares

- Since **h** is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Minimize $\|\mathbf{A}\hat{\mathbf{h}}\|^2$

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

- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

Inhomogeneous solution

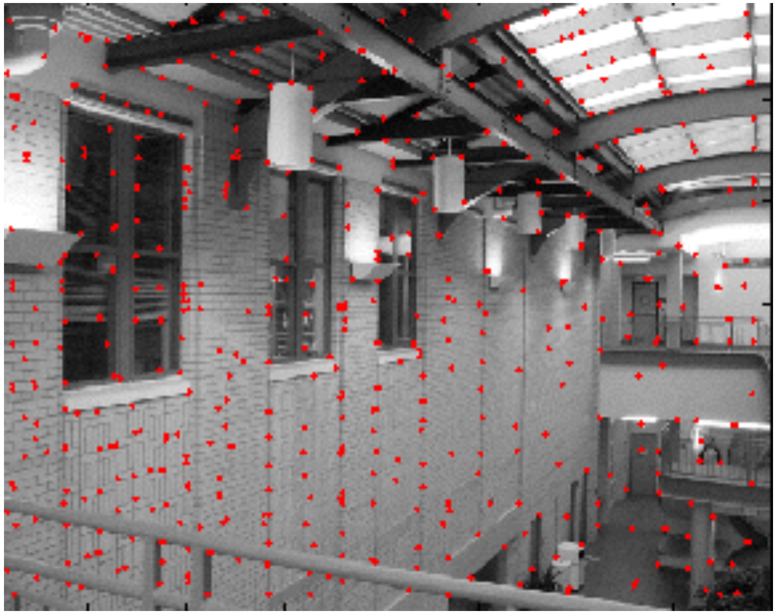
Since h can only be computed up to scale, impose constraint pick h_j=1, e.g. h_9=1, and solve for 8-vector

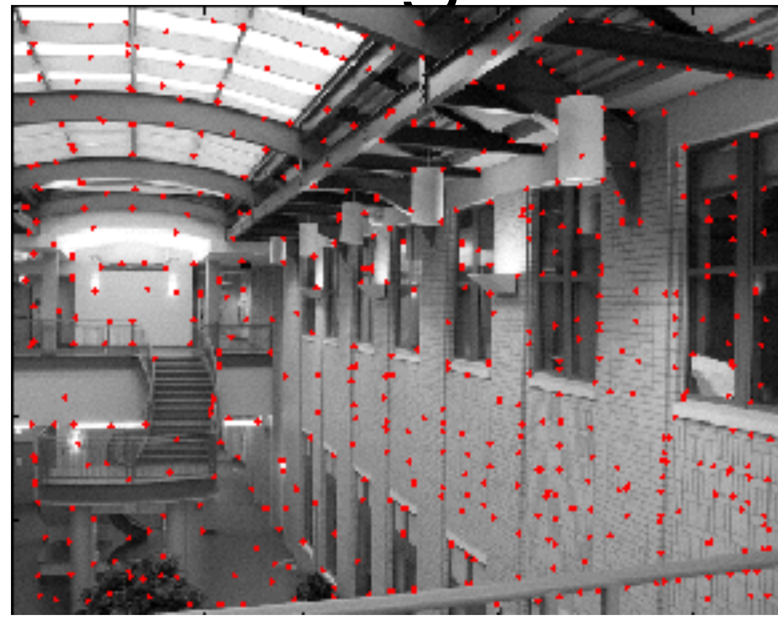
$$\begin{bmatrix} 0 & 0 & 0 & -x_i w_i' & -y_i w_i' & -w_i w_i' & x_i y_i' & y_i y_i' \\ x_i w_i' & y_i w_i' & w_i w_i' & 0 & 0 & 0 & x_i x_i' & y_i x_i' \end{bmatrix} \tilde{h} = \begin{pmatrix} -w_i y_i' \\ w_i x_i' \end{pmatrix}$$

Can be solved using linear least-squares

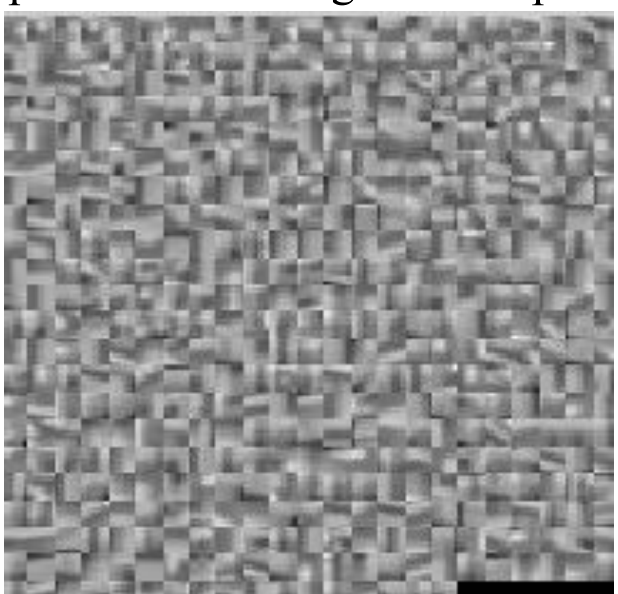
However, if h_9=0 this approach fails Also poor results if h_9 close to zero Therefore, not recommended

Feature matching



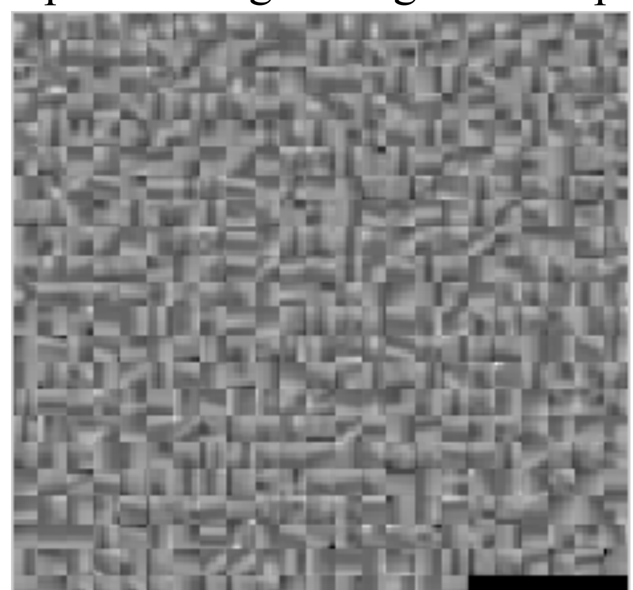


descriptors for left image feature points



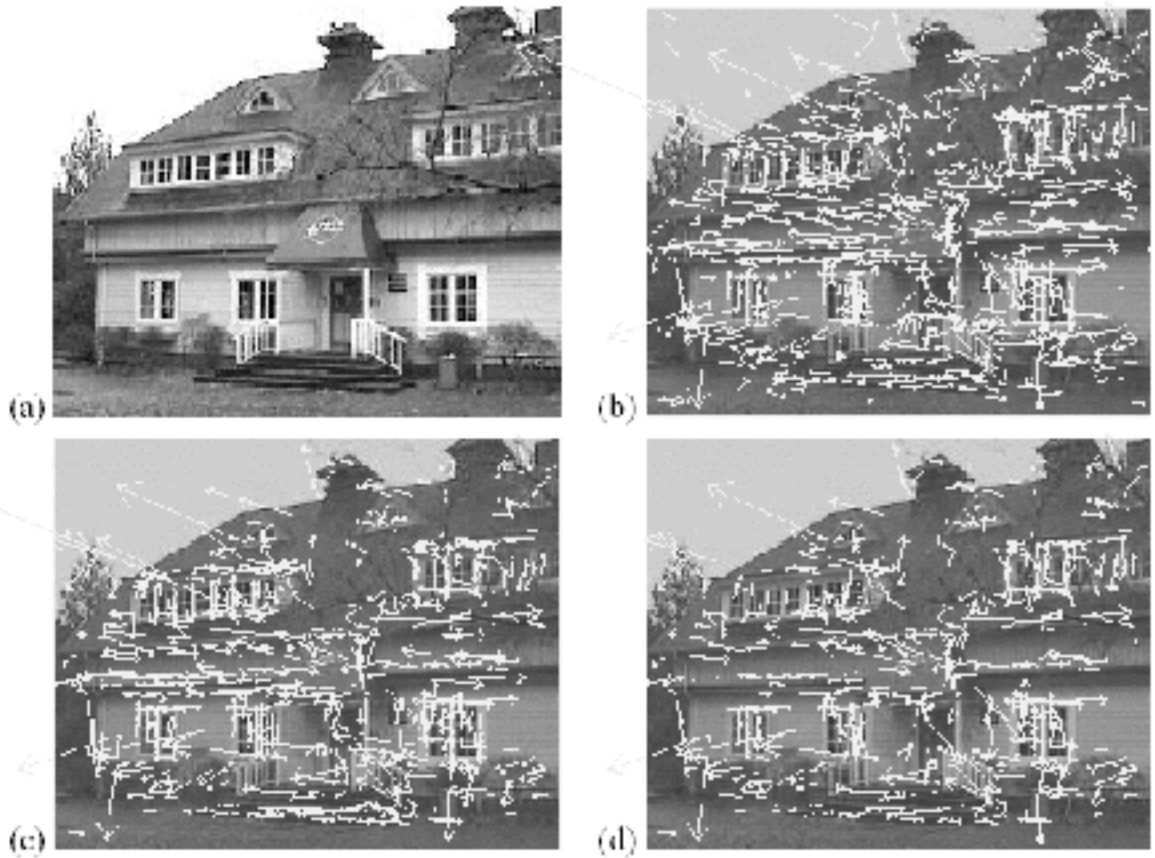
?

descriptors for right image feature points



SIFT features

- Example



(a) 233x189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures

Strategies to match images robustly

(a)<u>Working with individual features:</u> For each feature point, find most similar point in other image (SIFT distance)

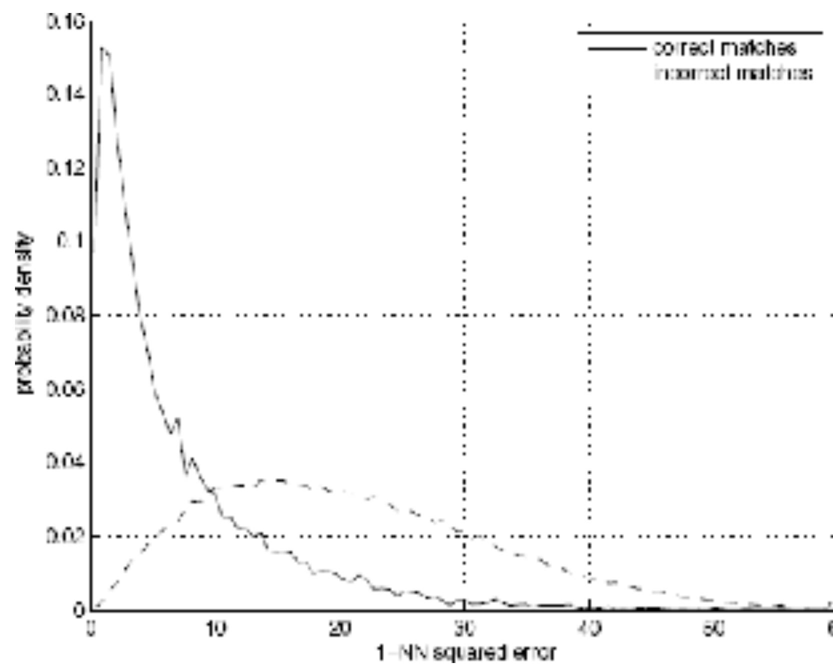
Reject ambiguous matches where there are too many similar points

(b)<u>Working with all the features:</u> Given some good feature matches, look for possible homographies relating the two images

Reject homographies that don't have many feature matches.

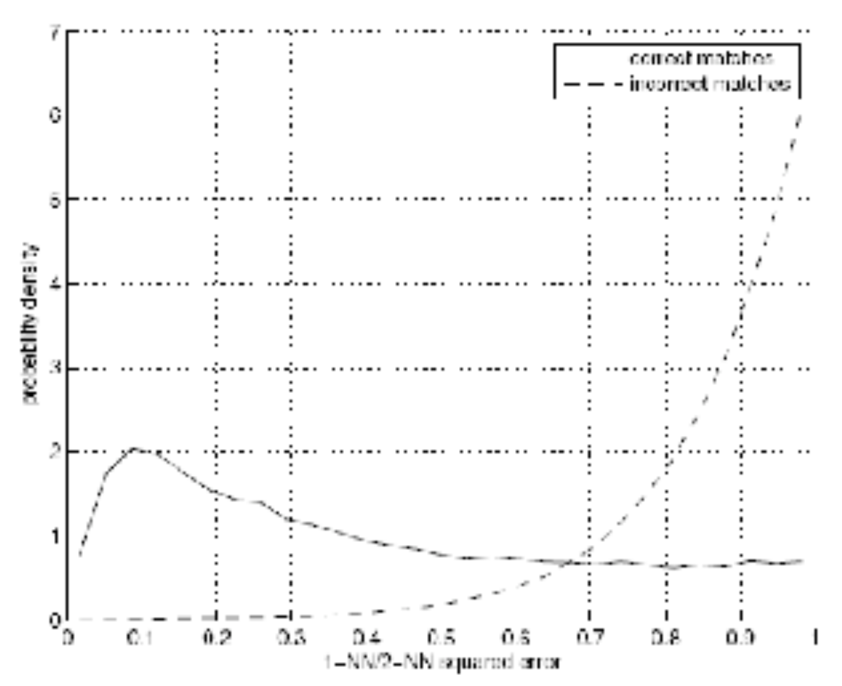
(a) Feature-space outlier rejection

- Let's not match all features, but only these that have "similar enough" matches?
- How can we do it?
  - SSD(patch1,patch2) < threshold
  - How to set threshold?
    Not so easy.

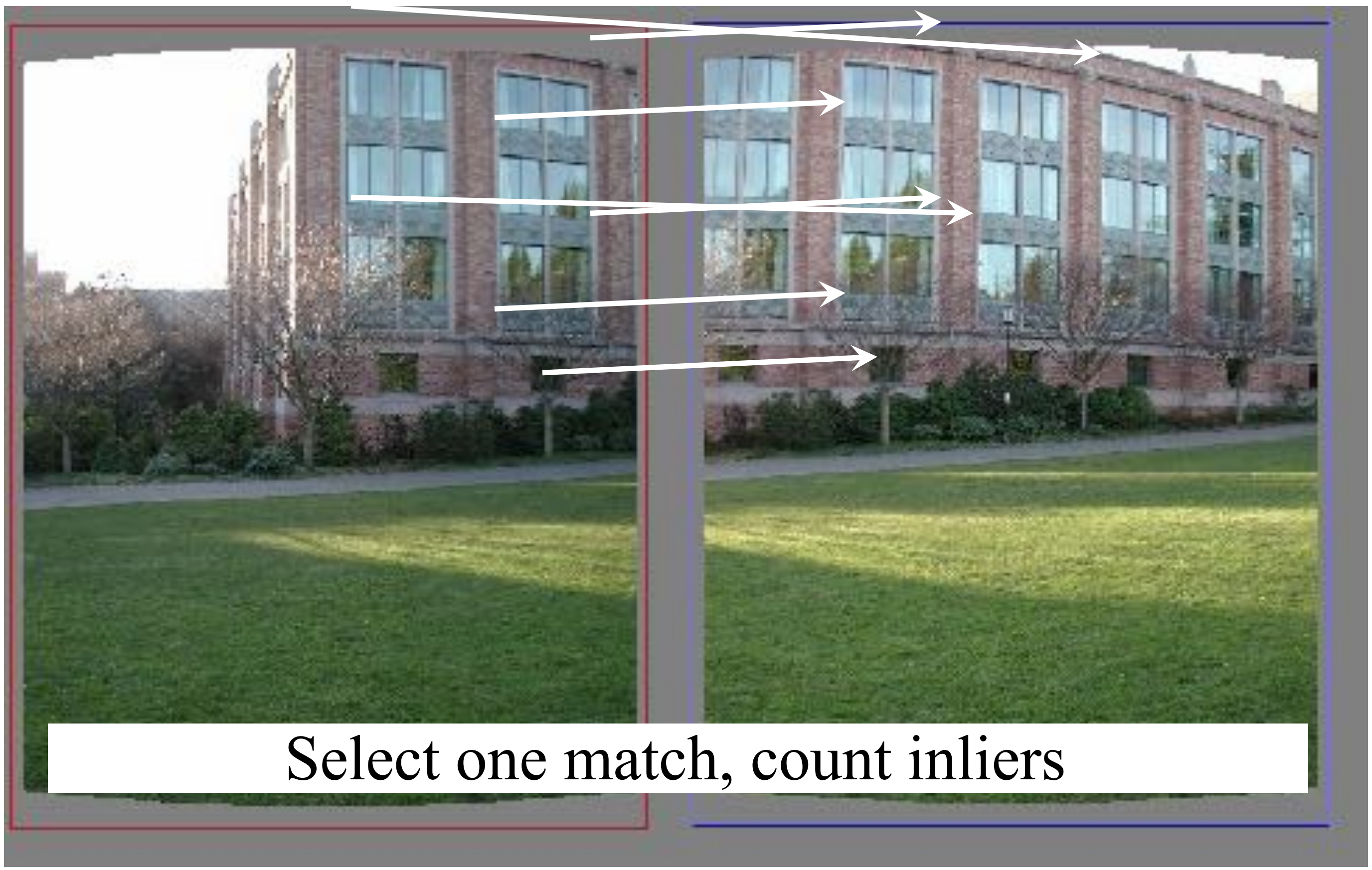


Feature-space outlier rejection

- A better way [Lowe, 1999]:
  - 1-NN: SSD of the closest match
  - 2-NN: SSD of the second-closest match
  - Look at how much better 1-NN is than 2-NN, e.g. 1-NN/2-NN
  - That is, is our best match so much better than the rest?



RAndom SAmple Consensus



RANSAC for estimating homography

RANSAC loop:

Select four feature pairs (at random)

Compute homography H (exact)

Compute inliers where $||p_i', H p_i|| < \varepsilon$

Keep largest set of inliers

Re-compute least-squares H estimate using all of the inliers