The Fundamental Matrix Song
Two-view geometry

3d point lies somewhere along \( r \)

epipolar line

epipolar plane

epipolar line (projection of \( r \))

Image 1

Image 2
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 $\mathbf{p}$ is: $\mathbf{F}\mathbf{p}$

- *Epipolar constraint* on corresponding points: $\mathbf{q}^T \mathbf{F}\mathbf{p} = 0$
Fundamental matrix

- Two Special points: $e_1$ and $e_2$ (the epipoles): projection of one camera into the other
Fundamental matrix

- Two special points: $e_1$ and $e_2$ (the *epipoles*): projection of one camera into the other
- All of the epipolar lines in an image pass through the epipole
Rectified case

- Images have the same orientation, \( t \) parallel to image planes
- Where are the epipoles?
Epipolar geometry demo
Relationship with homography?

Images taken from the same center of projection? Use a homography!
Fundamental matrix – uncalibrated case

$\mathbf{F} \leftarrow$ the Fundamental matrix

\[
q^T \mathbf{K}_2^{-T} \mathbf{R} [t] \times \mathbf{K}_1^{-1} \mathbf{p} = 0
\]
Cross-product as linear operator

**Useful fact:** Cross product with a vector $\mathbf{t}$ can be represented as multiplication with a \textit{(skew-symmetric)} $3\times3$ matrix

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

$$\mathbf{t} \times \mathbf{\tilde{p}} = [\mathbf{t}]_\times \mathbf{\tilde{p}}$$
Fundamental matrix – calibrated case

\[ \tilde{p} = K_1^{-1} p \quad : \text{ray through } p \text{ in camera 1’s (and world) coordinate system} \]

\[ \tilde{q} = K_2^{-1} q \quad : \text{ray through } q \text{ in camera 2’s coordinate system} \]

\[ \tilde{q}^T R [t] \times \tilde{p} = 0 \]

\[ \tilde{q}^T E \tilde{p} = 0 \]

\[ E \leftarrow \text{the Essential matrix} \]
Properties of the Fundamental Matrix

- $F_p$ is the epipolar line associated with $p$

- $F^T q$ is the epipolar line associated with $q$

- $Fe_1 = 0$ and $F^T e_2 = 0$

- $F$ is rank 2

- How many parameters does $F$ have?
Rectified case

\[ R = I_{3 \times 3} \]
\[ t = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \]
\[ E = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \]
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

Questions?
Estimating $F$

• If we don’t know $K_1$, $K_2$, $R$, or $t$, can we estimate $F$ for two images?

• Yes, given enough correspondences
Estimating $F - 8$-point algorithm

• The fundamental matrix $F$ is defined by

$$x'^T F x = 0$$

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

• Let $x = (u, v, 1)^T$ and $x' = (u', v', 1)^T$, 

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

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 \( Af = 0 \), we seek \( f \) to minimize \( \| Af \| \), least eigenvector of \( A^TA \).
8-point algorithm – Problem?

- \( \mathbf{F} \) should have rank 2
- To enforce that \( \mathbf{F} \) is of rank 2, \( \mathbf{F} \) is replaced by \( \mathbf{F}' \) that minimizes \( \| \mathbf{F} - \mathbf{F}'\| \) subject to the rank constraint.

- This is achieved by SVD. Let \( \mathbf{F} = \mathbf{U} \Sigma \mathbf{V}^T \), where

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

then \( \mathbf{F}' = \mathbf{U} \Sigma' \mathbf{V}^T \) is the solution.
8-point algorithm

% Build the constraint matrix
A = [x2(1,:)'.*x1(1,:)' x2(1,:)'.*x1(2,:)'
    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
Results (ground truth)

- Ground truth with standard stereo calibration
Results (normalized 8-point algorithm)
So Far
I am a 3D Point

\[ X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \]
Similar Triangle

\[ X = Y \]

image plane
Similar Triangle

\[ X = Y \]

\[ X = [X] \]

\[ Y \]

\[ Z \]

image plane
When they take a picture:

\[
\lambda \begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}
\]

Up to a scale

\[
\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
\]
World Coor. $\rightarrow$ Camera Coor.

\[
\begin{bmatrix}
 x \\
 y \\
 1
\end{bmatrix} =
\begin{bmatrix}
 f & 0 & 0 \\
 0 & f & 0 \\
 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
 1 & 0 & 0 & 0 \\
 0 & 1 & 0 & 0 \\
 0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
 R \\
 t \\
 0^T \\
 1
\end{bmatrix}
\begin{bmatrix}
 X \\
 Y \\
 Z \\
 1
\end{bmatrix}
\]

\[
 K =
\begin{bmatrix}
 f & 0 & 0 \\
 0 & f & 0 \\
 0 & 0 & 1
\end{bmatrix}
\]

\[
 x =
\begin{bmatrix}
 x \\
 y \\
 1
\end{bmatrix}
\]

\[
 x = K [ R | t ] X
\]

Camera Parameter
Intrinsic
Extrinsic

P = K [ R | t ]
Computing vanishing points

- $P_\infty$ is a point at \textit{infinity}, \textbf{v} is its projection
- Depends only on line \textit{direction}
- Parallel lines $P_0 + t\mathbf{D}$, $P_1 + t\mathbf{D}$ intersect at $P_\infty$
Computing vanishing lines

• Properties
  – $l$ is intersection of horizontal plane through $C$ with image plane
  – Compute $l$ from two sets of parallel lines on ground plane
  – All points at same height as $C$ project to $l$
    • points higher than $C$ project above $l$
  – Provides way of comparing height of objects in the scene
Measuring height

How high is the camera?
The cross ratio

• A Projective Invariant
  – Something that does not change under projective transformations (including perspective projection)

The cross-ratio of 4 collinear points

\[
\begin{vmatrix}
\vec{P}_1 & \vec{P}_2 & \vec{P}_3 & \vec{P}_4 \\
X_i & Y_i & Z_i & 1
\end{vmatrix}
\]

\[
\left| \frac{\| \vec{P}_3 - \vec{P}_1 \| \| \vec{P}_4 - \vec{P}_2 \|}{\| \vec{P}_3 - \vec{P}_2 \| \| \vec{P}_4 - \vec{P}_1 \|} \right|
\]

Can permute the point ordering

• 4! = 24 different orders (but only 6 distinct values)

This is the fundamental invariant of projective geometry
Measuring height

\[ v \approx (b \times b_0) \times (v_x \times v_y) \]

vanishing line (horizon)

\[ t \approx (v \times t_0) \times (r \times b) \]

\[
\frac{||t - b||}{||r - b||} = \frac{H}{R}
\]

image cross ratio
Two-view geometry

- epipolar line
- epipolar plane
- 3d point lies somewhere along r
- epipolar line (projection of r)

Image 1

Image 2
Fundamental matrix – uncalibrated case

\[ F = q^T K_2^{-T} R [t] \times K_1^{-1} p = 0 \]

- $K_1$: intrinsics of camera 1
- $K_2$: intrinsics of camera 2
- $R$: rotation of image 2 w.r.t. camera 1
- $F$: the Fundamental matrix
Fundamental matrix – calibrated case

\( \tilde{p} = K_1^{-1} p \) : ray through \( p \) in camera 1’s (and world) coordinate system

\( \tilde{q} = K_2^{-1} q \) : ray through \( q \) in camera 2’s coordinate system

\[ \tilde{q}^T R [t] \times \tilde{p} = 0 \]

\[ \tilde{q}^T E \tilde{p} = 0 \]

\( E \leftarrow \) the Essential matrix
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. 
  Computer Vision and Pattern Recognition, 1999.
Estimating F – 8-point algorithm

• The fundamental matrix F is defined by

\[ x' F x = 0 \]

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

• Let \( x=(u,v,1)^T \) and \( x'=(u',v',1)^T \),

\[
F = \begin{bmatrix}
    f_{11} & f_{12} & f_{13} \\
    f_{21} & f_{22} & f_{23} \\
    f_{31} & f_{32} & f_{33}
\end{bmatrix}
\]

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 \( Af = 0 \), we seek \( f \) to minimize \( \|Af\| \), least eigenvector of \( A^TA \).
What about more than two views?

- The geometry of three views is described by a $3 \times 3 \times 3$ tensor called the *trifocal tensor*.

- The geometry of four views is described by a $3 \times 3 \times 3 \times 3$ tensor called the *quadrifocal tensor*.

- After this it starts to get complicated...
Point-to-point Correspondences
Image Correspondence

Goal: Find map between two images
Image Correspondence

Goal: Find map between two images

- Sparse correspondences: map a small number of points
- Dense correspondences: map all points
This class: interest points

• Note: “interest points” = “keypoints”, also sometimes called “features”

• Many applications
  – tracking: which points are good to track?
  – recognition: find patches likely to tell us something about object category
  – 3D reconstruction: find correspondences across different views
This class: interest points

• Suppose you have to click on some point, go away and come back after I deform the image, and click on the same points again.
  – Which points would you choose?
Overview of Keypoint Matching

1. Find a set of distinctive keypoints
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

\[ d(f_A, f_B) < T \]
Goals for Keypoints

Detect points that are *repeatable* and *distinctive*
Key trade-offs

Detection of interest points

More Repeatable
- Robust detection
- Precise localization

More Points
- Robust to occlusion
- Works with less texture

Description of patches

More Distinctive
- Minimize wrong matches

More Flexible
- Robust to expected variations
- Maximize correct matches
Invariant Local Features

Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters.
Choosing distinctive interest points

• If you wanted to meet a friend would you say
  a) “Let’s meet on campus.”
  b) “Let’s meet on Green street.”
  c) “Let’s meet at Green and Wright.”
  – Corner detection

• Or if you were in a secluded area:
  a) “Let’s meet in the Plains of Akbar.”
  b) “Let’s meet on the side of Mt. Doom.”
  c) “Let’s meet on top of Mt. Doom.”
  – Blob (valley/peak) detection
Matching Image Features

1) Feature Detection: Identify image features
2) Feature Description: Extract feature descriptor for each feature
3) Feature Matching: Find candidate matches between features
4) Feature Correspondence: Find consistent set of (inlier) correspondences between features
Matching Image Features

1) Feature Detection:
   Identify image features

2) Feature Description:
   Extract feature descriptor for each feature

3) Feature Matching:
   Find candidate matches between features

4) Feature Correspondence:
   Find consistent set of (inlier) correspondences between features
Choosing interest points

Where would you tell your friend to meet you?
Choosing interest points

Where would you tell your friend to meet you?
Feature extraction: Corners

9300 Harris Corners Pkwy, Charlotte, NC

Slides from Rick Szeliski, Svetlana Lazebnik, and Kristin Grauman
Why extract features?

• Motivation: panorama stitching
  • We have two images – how do we combine them?
Local features: main components

1) Detection: Identify the interest points

2) Description: Extract vector feature descriptor surrounding each interest point.

\[ x_1 = [x_1^{(1)}, \ldots, x_d^{(1)}] \]

\[ x_2 = [x_1^{(2)}, \ldots, x_d^{(2)}] \]

3) Matching: Determine correspondence between descriptors in two views
Characteristics of good features

- **Repeatability**
  - The same feature can be found in several images despite geometric and photometric transformations

- **Saliency**
  - Each feature is distinctive

- **Compactness and efficiency**
  - Many fewer features than image pixels

- **Locality**
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion
Goal: interest operator repeatability

• We want to detect (at least some of) the same points in both images.

No chance to find true matches!

• Yet we have to be able to run the detection procedure \textit{independently} per image.
Goal: descriptor distinctiveness

• We want to be able to reliably determine which point goes with which.

• Must provide some invariance to geometric and photometric differences between the two views.
Local features: main components

1) Detection: Identify the interest points

2) Description: Extract vector feature descriptor surrounding each interest point.

3) Matching: Determine correspondence between descriptors in two views
Many Existing Detectors Available

<table>
<thead>
<tr>
<th>Method</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hessian &amp; Harris</td>
<td>[Beaudet ‘78], [Harris ‘88]</td>
</tr>
<tr>
<td>Laplacian, DoG</td>
<td>[Lindeberg ‘98], [Lowe 1999]</td>
</tr>
<tr>
<td>Harris-/Hessian-Laplace</td>
<td>[Mikolajczyk &amp; Schmid ‘01]</td>
</tr>
<tr>
<td>Harris-/Hessian-Affine</td>
<td>[Mikolajczyk &amp; Schmid ‘04]</td>
</tr>
<tr>
<td>EBR and IBR</td>
<td>[Tuytelaars &amp; Van Gool ‘04]</td>
</tr>
<tr>
<td>MSER</td>
<td>[Matas ‘02]</td>
</tr>
<tr>
<td>Salient Regions</td>
<td>[Kadir &amp; Brady ‘01]</td>
</tr>
<tr>
<td>Others…</td>
<td></td>
</tr>
</tbody>
</table>
Feature Detection

Goals:

• Repeatability
  – The same feature can be found in several images despite geometric and photometric transformations

• Distinctiveness
  – Each feature has a distinguishing description

• Compactness and efficiency
  – Many fewer features than image pixels

• Locality
  – A feature occupies a relatively small area of the image; robust to clutter and occlusion
Feature Detection: Repeatability

• We want to detect (at least some of) the same features in both images.

• Yet we have to be able to run the detection procedure *independently* per image.
Feature Detection: Repeatability

• We want to detect (at least some of) the same features in both images.

No chance to find true matches!

• Features should appear at “stable” locations that are invariant to typical image variations
Feature Detection: Distinctiveness

• We want to be able to reliably determine which feature goes with which

• Features should appear at locations with distinctive appearances
• What points would you choose?
Feature Detection

Some feature point detection methods:

• Corners
  – Harris
• Scale-space blobs
  – SIFT
Feature Detection

Some feature point detection methods:

• **Corners**
  – Harris

• **Scale-space blobs**
  – SIFT
Corner Detection: Basic Idea

- We should easily recognize the point by looking through a small window.
- Shifting a window in *any direction* should give a *large change* in intensity.

**"flat" region:** no change in all directions

**"edge":** no change along the edge direction

**"corner":** significant change in all directions

Source: A. Efros
Finding Corners

• Key property: in the region around a corner, image gradient has two or more dominant directions

• Corners are repeatable and distinctive

Corner Detection: Mathematics

Change in appearance of window $w(x,y)$ for the shift $[u,v]$:

$$E(u, v) = \sum_{x,y} w(x, y) \left[ I(x + u, y + v) - I(x, y) \right]^2$$
Corner Detection: Mathematics

Change in appearance of window $w(x,y)$ for the shift $[u,v]$:

$$E(u,v) = \sum_{x,y} w(x,y) \left[ I(x+u, y+v) - I(x, y) \right]^2$$
Corner Detection: Mathematics

Change in appearance of window \( w(x,y) \) for the shift \([u,v]\):

\[
E(u,v) = \sum_{x,y} w(x,y) \left[ I(x+u, y+v) - I(x, y) \right]^2
\]

Window function
Shifted intensity
Intensity

Window function \( w(x,y) = \) 1 in window, 0 outside
or
Gaussian

Source: R. Szeliski
Corner Detection: Mathematics

Change in appearance of window \( w(x,y) \) for the shift \([u,v]\):

\[
E(u,v) = \sum_{x,y} w(x,y) \left[ I(x+u, y+v) - I(x, y) \right]^2
\]

We want to find out how this function behaves for small shifts
Corner Detection: Mathematics

Change in appearance of window \( w(x,y) \) for the shift \([u,v]\):

\[
E(u,v) = \sum_{x,y} w(x,y) \left[ I(x+u, y+v) - I(x, y) \right]^2
\]

We want to find out how this function behaves for small shifts.

Local quadratic approximation of \( E(u,v) \) in the neighborhood of \((0,0)\) is given by the second-order Taylor expansion:

\[
E(u,v) \approx E(0,0) + [u \ v] \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} [u \ v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} [u \ v]
\]
Corner Detection: Mathematics

\[ E(u, v) = \sum_{x, y} w(x, y) \left[ I(x + u, y + v) - I(x, y) \right]^2 \]

Second-order Taylor expansion of \( E(u,v) \) about \((0,0)\):

\[
E(u, v) \approx E(0,0) + [u \ v] \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} [u \ v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} [u \ v]
\]

\[
E_u(u, v) = \sum_{x,y} 2w(x, y) \left[ I(x + u, y + v) - I(x, y) \right] I_x(x + u, y + v)
\]

\[
E_{uu}(u, v) = \sum_{x,y} 2w(x, y) I_x(x + u, y + v) I_x(x + u, y + v)
\]

\[
+ \sum_{x,y} 2w(x, y) \left[ I(x + u, y + v) - I(x, y) \right] I_{xx}(x + u, y + v)
\]

\[
E_{uv}(u, v) = \sum_{x,y} 2w(x, y) I_y(x + u, y + v) I_x(x + u, y + v)
\]

\[
+ \sum_{x,y} 2w(x, y) \left[ I(x + u, y + v) - I(x, y) \right] I_{xy}(x + u, y + v)
\]
Corner Detection: Mathematics

\[ E(u, v) = \sum_{x, y} w(x, y) \left[ I(x + u, y + v) - I(x, y) \right]^2 \]

Second-order Taylor expansion of \( E(u,v) \) about \((0,0)\):

\[ E(u, v) \approx E(0,0) + [u \ v] \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} [u \ v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} [u \ v] \]

\[ E(0,0) = 0 \]
\[ E_u(0,0) = 0 \]
\[ E_v(0,0) = 0 \]

\[ E_{uu}(0,0) = \sum_{x,y} 2w(x, y)I_x(x, y)I_x(x, y) \]

\[ E_{vv}(0,0) = \sum_{x,y} 2w(x, y)I_y(x, y)I_y(x, y) \]

\[ E_{uv}(0,0) = \sum_{x,y} 2w(x, y)I_x(x, y)I_y(x, y) \]
Corner Detection: Mathematics

\[
E(u, v) = \sum_{x, y} w(x, y) \left[ I(x + u, y + v) - I(x, y) \right]^2
\]

Second-order Taylor expansion of \(E(u, v)\) about \((0,0)\):

\[
E(u, v) \approx [u \ v] \begin{bmatrix}
\sum_{x, y} w(x, y) I_x^2(x, y) & \sum_{x, y} w(x, y) I_x(x, y) I_y(x, y) \\
\sum_{x, y} w(x, y) I_x(x, y) I_y(x, y) & \sum_{x, y} w(x, y) I_y^2(x, y)
\end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
\]

\[
E(0,0) = 0
\]
\[
E_u(0,0) = 0
\]
\[
E_v(0,0) = 0
\]
\[
E_{uu}(0,0) = \sum_{x, y} 2w(x, y) I_x(x, y) I_x(x, y)
\]
\[
E_{vv}(0,0) = \sum_{x, y} 2w(x, y) I_y(x, y) I_y(x, y)
\]
\[
E_{uv}(0,0) = \sum_{x, y} 2w(x, y) I_x(x, y) I_y(x, y)
\]


Corner Detection: Mathematics

The quadratic approximation simplifies to

$$E(u, v) \approx [u \ v] \ M \begin{bmatrix} u \\ v \end{bmatrix}$$

where $M$ is a second moment matrix computed from image derivatives:

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} = \sum \nabla I (\nabla I)^T$$
Corners as distinctive interest points

\[ M = \sum w(x, y) \begin{bmatrix} I_xI_x & I_xI_y \\ I_yI_x & I_yI_y \end{bmatrix} \]

2 x 2 matrix of image derivatives (averaged in neighborhood of a point).

Notation:

\[ I_x \Leftrightarrow \frac{\partial I}{\partial x} \quad I_y \Leftrightarrow \frac{\partial I}{\partial y} \quad I_xI_y \Leftrightarrow \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \]
Interpreting the second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form. Let’s try to understand its shape.

$$E(u,v) \approx [u \ v] \ M \ [u \ v]$$

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
Interpreting the second moment matrix

First, consider the axis-aligned case (gradients are either horizontal or vertical)

\[ M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \]

If either \( \lambda \) is close to 0, then this is **not** a corner, so look for locations where both are large.
Interpreting the second moment matrix

Consider a horizontal “slice” of $E(u, v)$:

$$[u \ v] \ M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

This is the equation of an ellipse.
Consider a horizontal “slice” of $E(u, v)$: $[u \ v] \ M \ \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.

Diagonalization of $M$:

$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by $R$.
Visualization of second moment matrices
Visualization of second moment matrices
Interpreting the eigenvalues

Classification of image points using eigenvalues of $M$:

- **“Corner”**: $\lambda_1$ and $\lambda_2$ are large, $\lambda_1 \sim \lambda_2$; $E$ increases in all directions.
- **“Edge”**: $\lambda_2 >> \lambda_1$; $E$ increases in all directions.
- **“Flat” region**: $\lambda_1$ and $\lambda_2$ are small; $E$ is almost constant in all directions.
Corner response function

\[ R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \]

\( \alpha \): constant (0.04 to 0.06)
Harris corner detector

1) Compute $M$ matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response ($f >$ threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

Harris Detector [Harris88]

- Second moment matrix

\[ \mu(\sigma_1, \sigma_D) = g(\sigma_1) \star \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix} \]

1. Image derivatives (optionally, blur first)

\[ \det M = \lambda_1 \lambda_2 \]
\[ \text{trace } M = \lambda_1 + \lambda_2 \]

2. Square of derivatives

3. Gaussian filter \( g(\sigma) \)

4. Cornerness function – both eigenvalues are strong

\[ har = \det[\mu(\sigma_1, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_1, \sigma_D))^2] = g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2 \]

5. Non-maxima suppression
Harris Detector: Steps
Harris Detector: Steps

Compute corner response $R$
Harris Detector: Steps

Find points with large corner response: $R > \text{threshold}$
Harris Detector: Steps

Take only the points of local maxima of $R$
Harris Detector: Steps
Invariance and covariance

- We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations
  - **Invariance**: image is transformed and corner locations do not change
  - **Covariance**: if we have two transformed versions of the same image, features should be detected in corresponding locations
Affine intensity change

\[ I \rightarrow a \, I + b \]

- Only derivatives are used => invariance to intensity shift \( I \rightarrow I + b \)
- Intensity scaling: \( I \rightarrow a \, I \)

Partially invariant to affine intensity change

![Diagram showing invariant properties under affine intensity change]
• Derivatives and window function are shift-invariant

Corner location is invariant w.r.t. translation
Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is invariant w.r.t. rotation
Scaling

Corner

All points will be classified as edges

Corner location is not invariant to scaling!
Scale invariant interest points

How can we independently select interest points in each image, such that the detections are repeatable across different scales?
Feature Detection

Some feature point detection methods:

• Corners
  – Harris corner detector

• Scale-space blobs
  – SIFT
Blob detection

Intuition: centers of “blobs” provide stable feature points
Scale invariant interest points

Intuition: size of blobs provide feature scale

Image 1

Image 2

$s_1$

$s_2$

region size

region size
Blob detection

Laplacian of Gaussian: good operator for blob detection

\[
\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}
\]
Blob detection: scale selection

Laplacian-of-Gaussian = “blob” detector

\[ \nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \]
Blob detection

We define the *characteristic scale* as the scale that produces peak of Laplacian response
Example

Original image at $\frac{3}{4}$ the size
Original image at $\frac{3}{4}$ the size

Kristen Grauman
Scale invariant interest points

Interest points are local maxima in both position and scale.

\[ L_{xx}(\sigma) + L_{yy}(\sigma) \rightarrow \sigma_1 \]
\[ \sigma_2 \]
\[ \sigma_3 \]
\[ \sigma_4 \]
\[ \sigma_5 \]

Squared filter response maps

⇒ List of \((x, y, \sigma)\)
Scale-space blob detector: Example

Image credit: Lana Lazebnik
SIFT Feature Detector

Scale-space blob detection, but …

• Approximate the Laplacian with a difference of Gaussians (DoG)
  – More efficient to implement

• Reject points with bad contrast:
  – DoG smaller than 0.03 (image values in [0,1])

• Reject edges
  – Similar to the Harris detector; look at the autocorrelation matrix
SIFT Feature Detector

Approximate the Laplacian with a difference of Gaussians; more efficient to implement.

\[ L = \sigma^2 \left( G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right) \]

(\text{Laplacian})

\[ \text{DoG} = G(x, y, k\sigma) - G(x, y, \sigma) \]

(\text{Difference of Gaussians})

\[ I(k\sigma) - I(\sigma) = \text{DoG} \]
SIFT Feature Detector

Maxima of DoG
SIFT Feature Detector

After removing low contrast and edges

After removing low contrast and edges
SIFT Orientation assignment

By assigning a consistent orientation, the keypoint descriptor can be orientation invariant.

Let, for a keypoint, $L$ is the image with the closest scale.

- Compute gradient magnitude and orientation using finite differences:

$$\text{GradientVector} = \begin{bmatrix} L(x+1, y) - L(x-1, y) \\ L(x, y+1) - L(x, y-1) \end{bmatrix}$$

$$m(x,y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y)))$$
SIFT Orientation assignment

- Keypoint location = extrema location
- Keypoint scale is scale of the DOG image
SIFT Orientation assignment

- **Gaussian image**
  - (at closest scale, from pyramid)

- **Gradient magnitude**

- **Gradient orientation**
SIFT Orientation assignment

weighted gradient magnitude

weighted orientation histogram. Each bucket contains sum of weighted gradient magnitudes corresponding to angles that fall within that bucket.

36 buckets
10 degree range of angles in each bucket, i.e.
0 <= ang < 10 : bucket 1
10 <= ang < 20 : bucket 2
20 <= ang < 30 : bucket 3 ...
SIFT Orientation assignment

weighted gradient magnitude

weighted orientation histogram.

gradient orientation

Orientation of keypoint is approximately 25 degrees
SIFT Orientation Assignment

Any peak within 80% of the highest peak is used to create a keypoint with that orientation

~15% assigned multiplied orientations, but contribute significantly to the stability
SIFT Feature Points
Matching Image Features

1) **Feature Detection:** Identify image features

2) **Feature Description:** Extract feature descriptor for each feature

3) **Feature Matching:** Find candidate matches between features

4) **Feature Correspondence:** Find consistent set of (inlier) correspondences between features
How should we represent the local area around each feature point (for matching)?
Raw Pixel Values?

The simplest way to describe the neighborhood around an interest point is to write down the list of luminances in a $k \times k$ window around the point to form a feature vector.
Raw Pixel Values?

The simplest way to describe the neighborhood around an interest point is to write down the list of luminances in a $k \times k$ window around the point to form a feature vector.
Raw Pixel Values?

Problems?
Window Feature Descriptor

Rotate image region around point based on the feature orientation

Scale image region around the point based on the feature scale

Concatenate luminance of all pixels in the $k \times k$ window around the point in the rotated/scaled window into a feature vector
Window Feature Descriptor

- e.g. scale, translation, rotation
Window Feature Descriptor

Problems?
Window Feature Descriptor

Sensitive to small shifts or rotations

region A

region B

vector a

vector b
Window Feature Descriptor

Sensitive to photometric differences

Figure from T. Tuytelaars ECCV 2006 tutorial
SIFT Feature Descriptor [Lowe 2004]

Use histograms to bin pixels within sub-patches according to their orientation.
SIFT Feature Descriptor

Gradient magnitude and orientation at each point weighted by a Gaussian

Orientation histograms: sum of gradient magnitude at each direction

In practice, 4x4 arrays of 8 bin histogram is used, a total of 128 features for one keypoint
SIFT Feature Descriptor

• Extraordinarily robust matching technique
  • Can handle changes in viewpoint
    • Up to about 60 degree out of plane rotation
  • Can handle significant changes in illumination
    • Sometimes even day vs. night (below)
  • Fast and efficient—can run in real time
  • Lots of code available
SIFT Feature Descriptor

NASA Mars Rover images
SIFT Feature Descriptor

NASA Mars Rover images with SIFT feature matches
Figure by Noah Snavely
Matching Image Features

1) Feature Detection:
   Identify image features

2) Feature Description:
   Extract feature descriptor for each feature

3) Feature Matching:
   Find candidate matches between features

4) Feature Correspondence:
   Find consistent set of (inlier) correspondences between features
Feature Matching

Goal: produce a set of candidate matches that contains…

- Many “correct” matches
- As few “wrong” matches as possible

We will consider only these candidate matches when searching for correspondences
Feature Matching

How?
Feature Matching

Feature Position, Scale, & Orientation
Feature Matching

Feature Position, Scale, & Orientation

Feature Descriptor:

0.2
0.7
0.4
0.6
0.8
0.6
...
Feature Matching

Descriptor Space
Feature Matching

Descriptor Space
Feature Matching

What is a good way to choose candidate matches?
Feature Matching

Simple method 1: create matches from each feature $A_i$ to feature $B_k$ in $B$ with minimum $d(A_i,B_k)$

$d(A_i,B_j)$ is Euclidean distance in descriptor space
Feature Matching

Simple method 1: create matches from each feature $A_i$ to feature $B_k$ in $B$ with minimum $d(A_i, B_k)$

$d(A_i, B_j)$ is Euclidean distance in descriptor space
Feature Matching

Simple method 1: create matches from each feature $A_i$ to feature $B_k$ in B with minimum $d(A_i,B_k)$

d($A_i,B_j$) is Euclidean distance in descriptor space
Feature Matching

Simple method 2: create matches from each feature $A_i$ to feature $B_k$ in B with minimum $d(A_i, B_k)$ iff $d(A_i, B_k) < \text{threshold}$

$d(A_i, B_j)$ is Euclidean distance in descriptor space
Feature Matching

Simple method 2: create matches from each feature $A_i$ to feature $B_k$ in $B$ with minimum $d(A_i, B_k)$ iff $d(A_i, B_k) < \text{threshold}$

$d(A_i, B_j)$ is Euclidean distance in descriptor space
Feature Matching

Ratio method: create matches from each feature $A_i$ to the closest feature in $B$ with minimum $d(A_i, B_k)$ iff $d(A_i, B_k) / d(A_i, B_{k2}) < \text{threshold}$

$B_k$ is the closest feature
$B_{k2}$ is the second closest feature

d$(A_i, B_j)$ is Euclidean distance in descriptor space
Feature Matching

Ratio method: threshold ratio of L2 distance to best match divided by L2 distance to 2nd best match

Typically threshold at ratio < 0.6

Lowe IJCV 2004
Mutual closest method: create matches between features $A_i$ and $B_k$ iff they are mutually closest among all pairs of features.
Feature Matching

Mutual closest method: create matches between features $A_i$ and $B_k$ iff they are mutually closest
Feature Matching

Mutual closest method: create matches between features $A_i$ and $B_k$ iff they are mutually closest

(A,B) is a bad match
Feature Matching

Mutual closest method: create matches between features $A_i$ and $B_k$ iff they are mutually closest

$(A, B)$ is a good match
So Far!

Yes! I can!

Last lecture before fall recess!
Image Correspondence

Goal: Find map between two images
Top three most important things in CV:

1. Correspondence
2. Correspondence
3. Correspondence

Takeo Kanade
Matching Image Features

1) Feature Detection:
   Identify image features

2) Feature Description:
   Extract feature descriptor for each feature

3) Feature Matching:
   Find candidate matches between features

4) Feature Correspondence:
   Find consistent set of (inlier) correspondences between features
Corner Detection: Basic Idea

- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity

“**flat**” region: no change in all directions

“**edge**”: no change along the edge direction

“**corner**”: significant change in all directions

Source: A. Efros
Interpreting the second moment matrix

Consider a horizontal “slice” of $E(u, v)$: 

$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

This is the equation of an ellipse.

Diagonalization of $M$:

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by $R$.
Visualization of second moment matrices
Visualization of second moment matrices
Corner response function

\[ R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2 \]

\[ \alpha: \text{constant (0.04 to 0.06)} \]
Image translation

- Derivatives and window function are shift-invariant

Corner location is invariant w.r.t. translation
Image rotation

Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is invariant w.r.t. rotation
Scaling

Corner location is not invariant to scaling!

All points will be classified as *edges*
Scale invariant interest points

Intuition: size of blobs provide feature scale

$\text{Region size}$

$s_1$

$s_2$

$\text{Image 1}$

$\text{Image 2}$
Blob detection: scale selection

Laplacian-of-Gaussian = “blob” detector

\[ \nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \]
SIFT Orientation assignment

Weighted gradient magnitude

Weighted orientation histogram.

Gradient orientation

Orientation of keypoint is approximately 25 degrees
Window Feature Descriptor

Rotate image region around point based on the feature orientation

Scale image region around the point based on the feature scale

Concatenate luminance of all pixels in the $k \times k$ window around the point in the rotated/scaled window into a feature vector

Image from Matthew Brown
Window Feature Descriptor

e.g. scale, translation, rotation
SIFT Feature Descriptor [Lowe 2004]

Use histograms to bin pixels within sub-patches according to their orientation.
Feature Matching

Ratio method: create matches from each feature $A_i$ to the closest feature in $B$ with minimum $d(A_i, B_k)$ iff $d(A_i, B_k) / d(A_i, B_{k2}) < \text{threshold}$

$B_k$ is the closest feature

$B_{k2}$ is the second closest feature

$d(A_i, B_j)$ is Euclidean distance in descriptor space
Feature Matching

Ratio method: threshold ratio of L2 distance to best match divided by L2 distance to 2nd best match

Typically threshold at ratio < 0.6

Lowe IJCV 2004
Feature Matching

Mutual closest method: create matches between features $A_i$ and $B_k$ iff they are mutually closest among all pairs of features.
Image Mosaicing

1) Feature Detection: Identify image features

2) Feature Description: Extract feature descriptor for each feature

3) Feature Matching: Find candidate matches between features

4) Feature Correspondence: Find consistent set of (inlier) correspondences between features
Image Mosaicing

1) Feature Detection:
   Identify image features

2) Feature Description:
   Extract feature descriptor for each feature

3) Feature Matching:
   Find candidate matches between features

4) Feature Correspondence:
   Find consistent set of (inlier) correspondences between features
Feature Correspondence

Candidate matches

Source: L. Lazebnik
Feature Correspondence

Correspondences

Source: L. Lazebnik
Feature Correspondence

How?

Source: L. Lazebnik
Feature Correspondence

Goal: produce a set of correspondences that …

- Contains $\geq 4$ matches to define a homography
- Contains only matches that agree on the same homography
- Aligns as many matching features as possible
Estimating the Homography

A mosaic has a natural interpretation in 3D

- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*

Source: Steve Seitz
Estimating the Homography

A homography is the transformation that projects an image onto a new view plane from the same viewpoint (center of projection)

Source: Steve Seitz
Observation 1: any combination of $\geq 4$ matches defines a homography
Observation 1: any combination of $\geq 4$ matches defines a homography
Observation 2: every homography $H$ provides a map $P' = H(p)$
Observation 3: can measure how “good” a map is based on how many matches are aligned by it

- **generator matches**: features defining the homography
- **inlier matches**: other features aligned by homography
- **outlier matches**: others (not shown)
RANSAC for Image Mosaics

RANSAC loop:
1. Select four matches (at random)
2. Compute homography $H$ aligning those matches
3. Find inlier matches where $d(p_i', Hp_i) < \varepsilon$
4. Re-compute $H$ to align on all of its inliers (least squares)
5. Re-find inlier matches where $d(p_i', Hp_i) < \varepsilon$
6. $H^* = H$ if has $H$ largest set of inliers seen so far

Warp image by $H^*$
Composite images

Slide credit: Steve Seitz
RANSAC for Image Mosaics

RANSAC loop:
1. Select four matches (at random)
2. Compute homography $H$ aligning those matches
3. Find *inlier matches* where $d(p_i', Hp_i) < \varepsilon$
4. Re-compute $H$ to align on all of its inliers (least squares)
5. Re-find *inlier matches* where $d(p_i', Hp_i) < \varepsilon$
6. $H^* = H$ if has $H$ largest set of inliers seen so far
Warp image by $H^*$ and composite images

Source: L. Lazebnik
RANSAC for Image Mosaics

RANSAC loop:
1. Select four matches (at random)
2. Compute homography $H$ aligning those matches
3. Find inlier matches where $d(p_i', Hp_i) < \varepsilon$
4. Re-compute $H$ to align on all of its inliers (least squares)
5. Re-find inlier matches where $d(p_i', Hp_i) < \varepsilon$
6. $H^* = H$ if has $H$ largest set of inliers seen so far

Warp image by $H^*$ and composite images
RANSAC for Image Mosaics

RANSAC loop:
1. Select four matches (at random)
2. Compute homography $H$ aligning those matches
3. Find inlier matches where $d(p_i', Hp_i) < \varepsilon$
4. Re-compute $H$ to align on all of its inliers (least squares)
5. Re-find inlier matches where $d(p_i', Hp_i) < \varepsilon$
6. $H^* = H$ if has $H$ largest set of inliers seen so far
Warp image by $H^*$ and composite images

Source: L. Lazebnik
RANSAC loop:
1. Select four matches (at random)
2. Compute homography $H$ aligning those matches
3. Find inlier matches where $d(p_i', Hp_i) < \epsilon$
4. Re-compute $H$ to align on all of its inliers (least squares)
5. Re-find inlier matches where $d(p_i', Hp_i) < \epsilon$
6. $H^* = H$ if has $H$ largest set of inliers seen so far
Warp image by $H^*$ and composite images
RANSAC loop:
1. Select four matches (at random)
2. Compute homography $H$ aligning those matches
3. Find \textit{inlier matches} where $d(p_i', Hp_i) < \varepsilon$
4. Re-compute $H$ to align on all of its inliers (least squares)
5. Re-find \textit{inlier matches} where $d(p_i', Hp_i) < \varepsilon$
6. $H^* = H$ if has H largest set of inliers seen so far

Warp image by $H^*$ and composite images
Image Mosaicing (summary, so far)

1) Feature Detection: Identify image features

2) Feature Description: Extract feature descriptor for each feature

3) Feature Matching: Find candidate matches between features

4) Feature Correspondence: Find consistent set of (inlier) correspondences between features
4) Estimate homography:
   Solve linear system of equations

5) Warp one image to the other
   Apply homography transformation to every pixel

6) Composite images:
   Blend pixels in overlap area
Image Mosaicing (rest of the story)

4) Estimate homography:
   Solve linear system of equations

5) Warp one image to the other
   Apply homography transformation to every pixel

6) Composite images:
   Blend pixels in overlap area
Estimating the Homography

A mosaic has a natural interpretation in 3D

• The images are reprojected onto a common plane
• The mosaic is formed on this plane
• Mosaic is a *synthetic wide-angle camera*

Source: Steve Seitz
Estimating the Homography

A homography is the transformation that projects an image onto a new view plane from the same viewpoint (center of projection)

Source: Steve Seitz
What is a Homography?

A projective transform mapping between any two PPs with the same center of projection
  • rectangle should map to arbitrary quadrilateral
  • lines stay straight
  • but don’t stay parallel
What is a Homography?

A projective transform of 2D points in homogeneous coordinates can be represented by a 3x3 matrix

\[
\begin{bmatrix}
wx' \\
wy' \\
w
\end{bmatrix} = \begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

\((x, y)\) → Homogeneous Coordinates \((wx'/w, wy'/w)\) = \((x', y')\)
Estimating the Homography

To \textbf{compute} the homography given pairs of corresponding points in the images, we solve a system of equations

\[
\min ||Hp - p'||^2
\]
Image Mosaicing

4) Estimate homography:
   Solve linear system of equations

5) Warp one image to the other
   Apply homography transformation to every pixel

6) Composite images:
   Blend pixels in overlap area
Image Warping

Given a coordinate transform $H$ and a source image $f(x,y)$, compute a transformed image $g(H(x,y)) = f(x,y)$?
Image warping

Reverse mapping:
Get each pixel \( g(x',y') \) from its corresponding location \((x,y) = H^{-1}(x',y')\) in the source image \( f(x,y) \)

Slide from Alyosha Efros, CMU
Image Mosaicing

4) Estimate homography:
   Solve linear system of equations

5) Warp one image to the other
   Apply homography transformation to every pixel

6) Composite images:
   Blend pixels in overlap area