The correspondence problem
Why?

• Multiple images can give a clue about 3D structure
Why? Reconstruction

• Need to find which pixel in image 2 matches which in image 1 - the correspondence problem
Reconstruction from correspondence

• Given known cameras, correspondence gives the location of 3D point (*Triangulation*)
Reconstruction from correspondence
Reconstruction from correspondence

• Specific application: depth cameras

https://realsense.intel.com/stereo/
Microsoft Kinect
Reconstruction from correspondence - Pose estimation

• Given a 3D point, correspondence gives relationship between cameras (*Pose estimation / camera calibration*)
Pose-estimation
Pose-estimation / Camera calibration

• Specific application: panorama stitching
  • We have two images – how do we combine them?
Pose-estimation / Camera calibration

• Specific application: panorama stitching
  • We have two images – how do we combine them?

Step 1: extract correspondence
Pose-estimation / Camera calibration

• Specific application: panorama stitching
  • We have two images – how do we combine them?

Step 1: extract correspondence
Step 2: align images
Other applications of correspondence

- Recognition: Match image to product view
Other applications of correspondence

• Image alignment
• Motion tracking
• Robot navigation
Correspondence can be challenging
Correspondence

by Diva Sian

by swashford
Harder case

by Diva Sian

by scgbt
Harder still?
Answer below (look for tiny colored squares...)

NASA Mars Rover images with SIFT feature matches
Dense correspondence

- Some applications demand correspondence for every pixel
  - For example dense 3D reconstruction
Sparse correspondence

• Sometimes a sparse set of correspondences are enough
  • E.g. estimating pose or camera relationships. Why?
  • Pose / camera relationships only consist of a small number of variables
  • Need only a little bit of information to recover it.
A general pipeline for correspondence

1. If sparse correspondences are enough, *choose points for which we will search for correspondences (feature points)*

2. For each point (or every pixel if dense correspondence), describe point using a *feature descriptor*

3. Find best matching descriptors across two images (*feature matching*)

4. Use feature matches to perform downstream task, e.g., pose estimation
A general pipeline for correspondence

1. If sparse correspondences are enough, choose points for which we will search for correspondences (feature points)

2. For each point (or every pixel if dense correspondence), describe point using a feature descriptor

3. Find best matching descriptors across two images (feature matching)

4. Use feature matches to perform downstream task, e.g., pose estimation
Sparse correspondence

• Which pixels should be searching correspondence for?
  • Feature points / keypoints
What makes a good feature point?
Characteristics of good feature points

• **Repeatability / invariance**
  • The same feature point can be found in several images despite geometric and photometric transformations

• **Saliency / distinctiveness**
  • Each feature point is distinctive
  • Fewer "false" matches
Goal: repeatability

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

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

No chance to find true matches!
Repeatability / invariance

• The feature detector should “fire” at consistent places in spite of rotation, translation etc.
Repeatability / invariance
Goal: distinctiveness

- The feature point should be distinctive enough that it is easy to match
  - Should *at least* be distinctive from other patches nearby
Where would you tell your friend to meet you?
Where would you tell your friend to meet you?
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
The aperture problem
The aperture problem

• Individual pixels are ambiguous
• Idea: Look at whole patches!
The aperture problem

- Individual pixels are ambiguous
- Idea: Look at whole patches!
The aperture problem

- Some local neighborhoods are ambiguous
The aperture problem
Corner detection

• Main idea: Translating window should cause large differences in patch appearance
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
Corner detection the math

• Consider shifting the window $W$ by $(u,v)$
  • how do the pixels in $W$ change?
• Write pixels in window as a vector:

$$\phi_0 = [I(0, 0), I(0, 1), \ldots, I(n, n)]$$

$$\phi_1 = [I(0 + u, 0 + v), I(0 + u, 1 + v), \ldots, I(n + u, n + v)]$$

$$E(u, v) = \|\phi_0 - \phi_1\|^2_2$$
Corner detection: the math

Consider shifting the window $W$ by $(u, v)$

• how do the pixels in $W$ change?

• compare each pixel before and after by summing up the squared differences (SSD)

• this defines an SSD “error” $E(u,v)$:

$$E(u, v) = \sum_{(x, y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

• We want $E(u,v)$ to be as high as possible for all $u, v$!
Corner Detection: Mathematics

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

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

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

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

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 $[u,v]$:

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

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

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

We want to find out how this function behaves for small shifts
Small motion assumption

Taylor Series expansion of $I$:

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \text{higher order terms}$$

If the motion $(u,v)$ is small, then first order approximation is good

$$I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v$$

$$\approx I(x, y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide...
Corner detection: the math

Consider shifting the window $W$ by $(u,v)$:

- define an SSD “error” $E(u,v)$:

$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

$$\approx \sum_{(x,y) \in W} [I(x, y) + I_x u + I_y v - I(x, y)]^2$$

$$\approx \sum_{(x,y) \in W} [I_x u + I_y v]^2$$
Corner detection: the math

Consider shifting the window $W$ by $(u,v)$:

- define an “error” $E(u,v)$:

$$E(u, v) \approx \sum_{(x,y) \in W} [I_x u + I_y v]^2$$

$$\approx A u^2 + 2B uv + C v^2$$

Thus, $E(u,v)$ is locally approximated as a quadratic error function.
Interpreting the second moment matrix

Recall that we want $E(u,v)$ to be as large as possible for all $u,v$

What does this mean in terms of $M$?

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

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

Second moment matrix
\[ E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

\[ A = \sum_{(x,y) \in W} I_x^2 \]
\[ B = \sum_{(x,y) \in W} I_x I_y \]
\[ C = \sum_{(x,y) \in W} I_y^2 \]

Flat patch:
\[ I_x = 0 \]
\[ I_y = 0 \]

\[ M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \]

\[ M \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]

\[ E(u, v) = 0 \quad \forall u, v \]
\[ E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

\[ A = \sum_{(x,y) \in W} I_x^2 \]

\[ B = \sum_{(x,y) \in W} I_x I_y \]

\[ C = \sum_{(x,y) \in W} I_y^2 \]

Vertical edge: \( I_y = 0 \)

\[ M = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} \]

\[ M \begin{bmatrix} 0 \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]

\[ E(0, v) = 0 \ \forall v \]
\[ E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

\[
A = \sum_{(x,y)\in W} I_x^2 \\
B = \sum_{(x,y)\in W} I_x I_y \\
C = \sum_{(x,y)\in W} I_y^2
\]

Horizontal edge: \( I_x = 0 \)

\[ M = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix} \]

\[ M \begin{bmatrix} u \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]

\[ E(u, 0) = 0 \ \forall u \]
What about edges in arbitrary orientation?
\[ E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \]

\[ M \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow E(u, v) = 0 \]

Solutions to \( Mx = 0 \) are directions for which \( E \) is 0: window can slide in this direction without changing appearance.
\[ E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \]

Solutions to \( Mx = 0 \) are directions for which \( E \) is 0: window can slide in this direction without changing appearance.

For corners, we want no such directions to exist.
Eigenvalues and eigenvectors of $M$

• $Mx = 0 \Rightarrow Mx = \lambda x$: $x$ is an eigenvector of $M$ with eigenvalue 0

• $M$ is 2 x 2, so it has 2 eigenvalues $(\lambda_{max}, \lambda_{min})$ with eigenvectors $(x_{max}, x_{min})$

• $E(x_{max}) = x_{max}^T M x_{max} = \lambda_{max} ||x_{max}||^2 = \lambda_{max}$ (eigenvectors have unit norm)

• $E(x_{min}) = x_{min}^T M x_{min} = \lambda_{min} ||x_{min}||^2 = \lambda_{min}$
Eigenvalues and eigenvectors of $M$

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

Eigenvalues and eigenvectors of $M$

1. Define shift directions with the smallest and largest change in error
2. $x_{\text{max}}$ = direction of largest increase in $E$
3. $\lambda_{\text{max}}$ = amount of increase in direction $x_{\text{max}}$
4. $x_{\text{min}}$ = direction of smallest increase in $E$
5. $\lambda_{\text{min}}$ = amount of increase in direction $x_{\text{min}}$

$$M x_{\text{max}} = \lambda_{\text{max}} x_{\text{max}}$$

$$M x_{\text{min}} = \lambda_{\text{min}} x_{\text{min}}$$
Interpreting the eigenvalues

- \( \lambda_{\text{max}}, \lambda_{\text{min}} \) are small; \( E \) is almost 0 in all directions
- \( \lambda_{\text{max}} \approx \lambda_{\text{min}} \gg 0 \)
  E very high in all directions
- \( \lambda_{\text{max}} \gg \lambda_{\text{min}}, \lambda_{\text{min}} \approx 0 \)
  E remains close to 0 along \( x_{\text{min}} \)

Corresponding to the diagram:
- Corner
- Edge
- Flat patch