Project 1 Feedback

Thank you all.

- There are no late days on proj0
- Numpy gearup beyond proj0 tutorial – OK
- Short time frame [my fault]
- Axes3D - OK
- Improvements to code description – OK
## Project 1 Convolution Speeds

<table>
<thead>
<tr>
<th>Name</th>
<th>Convolution speed (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jake Chanan</td>
<td>0.91</td>
</tr>
<tr>
<td>Alice Marbach</td>
<td>0.94</td>
</tr>
<tr>
<td>Andrew Cooke</td>
<td>1.07</td>
</tr>
<tr>
<td>Albert Webson</td>
<td>1.2</td>
</tr>
<tr>
<td>Reza Esfandiarpoor</td>
<td>1.2</td>
</tr>
<tr>
<td>Andrew Levy</td>
<td>1.22</td>
</tr>
<tr>
<td>James White</td>
<td>1.23</td>
</tr>
<tr>
<td>Da Huo</td>
<td>1.24</td>
</tr>
<tr>
<td>Troy Moo Penn</td>
<td>1.27</td>
</tr>
<tr>
<td>Michael T Lincoln</td>
<td>1.29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>FFT convolution (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>James Wang</td>
<td>0.45</td>
</tr>
<tr>
<td>Matthew Kovoor</td>
<td>0.75</td>
</tr>
<tr>
<td>Isaiah Liu</td>
<td>0.85</td>
</tr>
</tbody>
</table>
Hybrid images

Figure 4: *Left:* Dog in high frequency *Right:* Lion in low frequency

Figure 5: Hybrid Image scales of Lion and African Dog
When you realize the statue of Mona Lisa looks like Keith Urban

Thus, I took these two images, merged them, and got this result:
Hybrid Image at Different Scales
Figure 2: *Left:* My Friend, Larry, *Right:* My Other Friend, Jeffrey
1. My own hybrid image, titled **Seeing vs. Believing**

![Hybrid Image](image)

**Figure 7:** Hybrid image for Starbucks and Wendy’s. (cut-off frequency: 4)
And here’s the hybrid scales:
Filtering → Edges → Corners

Feature points

Also called interest points, key points, etc. Often described as ‘local’ features.
Correspondence across views

Matching points, patches, edges, or regions across images.

*Sparse or local correspondence vs. dense correspondence* (at every pixel).
Fundamental to Applications

- Image alignment
- 3D reconstruction
- Motion tracking (robots, drones, AR)
- Indexing and database retrieval
- Object recognition
Example application: Panorama stitching

We have two images – how do we estimate how to overlay them?
Local features: main components

1) Detection:
Find a set of distinctive key points.

2) Description:
Extract feature descriptor around each interest point as vector.
\[ \mathbf{x}_1 = [x^{(1)}_1, \ldots, x^{(1)}_d] \]

3) Matching:
Compute distance between feature vectors to find correspondence.
\[ d(\mathbf{x}_1, \mathbf{x}_2) < T \]
Goal: Distinctiveness

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

May be difficult in structured environments with repeated elements
Goal: Repeatability

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

With these points, there’s no chance to find true matches!

Under geometric and photometric variations.
Example: Object Detection

Finding *distinctive* and *repeatable* feature points can be difficult when we want our features to be invariant to large transformations:

- geometric variation (translation, rotation, scale, perspective)
- appearance variation (reflectance, illumination)
Goal: Compactness and Efficiency

We want the representation to be as small and as fast as possible

- Much smaller than a whole image

We’d like to be able to run the detection procedure *independently* per image

- Match just the compact descriptors for speed.
- *Difficult!* We don’t get to see ‘the other image’ at match time, e.g., object detection.
Characteristics of good features

Distinctiveness
Each feature can be uniquely identified

Repeatability
The same feature can be found in several images despite differences:
- geometrically (translation, rotation, scale, perspective)
- photometrically (reflectance, illumination)

Compactness and efficiency
Many fewer features than image pixels; run independently per image
Local features: main components

1) Detection:
Find a set of distinctive key points.

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

3) Matching:
Compute distance between feature vectors to find correspondence.
Detection: Basic Idea

We do not know which other image locations the feature will end up being matched against.

*But* we can compute how stable a location is in appearance with respect to small variations in its position.

**Strategy:** Compare image patch against local neighbors.
Corner Detection: Basic Idea

Recognize corners by looking at small window.

We want a window shift in any direction to 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
Corner Detection by Auto-correlation

Change in appearance of window $w(x,y)$ for 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

Gaussian

Source: R. Szeliski
Corner Detection by Auto-correlation

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

\[
E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2
\]
Corner Detection by Auto-correlation

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

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

$I(x, y)$

$E(u, v)$

$w(x, y)$

$E(3, 2)$
Think-Pair-Share:

Correspond the three red crosses to (b,c,d).

\[ E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2 \]
Corner Detection by Auto-correlation

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

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

We want to discover how \( E \) behaves for small shifts

But this is very slow to compute naively.

\[
O(\text{window\_width}^2 \times \text{shift\_range}^2 \times \text{image\_width}^2)
\]

\[
O(11^2 \times 11^2 \times 600^2) = 5.2 \text{ billion of these}
\]

14.6k ops per image pixel
Corner Detection by Auto-correlation

Change in appearance of window $w(x,y)$ for 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 discover how $E$ behaves for small shifts

But we know the response in $E$ that we are looking for – strong peak.
Strategy:

Approximate $E(u,v)$ locally by a quadratic surface, and look for that instead.
Recall: Taylor series expansion

A function $f$ can be represented by an infinite series of its derivatives at a single point $a$:

$$f(a) + \frac{f'(a)}{1!}(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \cdots.$$

As we care about window centered, we set $a = 0$ (MacLaurin series)

Approximation of $f(x) = e^x$ centered at $f(0)$
Local quadratic approximation of $E(u,v)$ in the neighborhood of $(0,0)$ is given by the second-order Taylor expansion:

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

Notation: partial derivative
Local quadratic approximation of $E(u,v)$ in the neighborhood of $(0,0)$ is given by the \textit{second-order Taylor expansion}:

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

- Ignore function value; set to 0
- Ignore first derivative, set to 0
- Just look at shape of second derivative (2D quadratic surface)
The quadratic approximation simplifies to

\[ E(u, v) \approx [u \ v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{vu}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

where \( M \) is a second moment matrix computed from image derivatives:

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

\[ M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I(\nabla I)^T \]
Corners as distinctive interest points

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

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

Notation:

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

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

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

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

James Hays
Interpreting the second moment matrix

Let’s take horizontal “slices” of our approximation of $E(u, v)$:

Each coloured line in the diagram below is where
\[
\begin{bmatrix} u \\ v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{constant}
\]
Visualization of second moment matrix

Simple image: Checkerboard.

1 corner
4 edges
4 flat regions

Note: Edges show ‘wide’ response because image derivatives were blurred by Gaussian \( \sigma = 1 \) before visualizing.
Visualization of second moment matrices
For cornerness, we only care about the ‘steepness’, not the rotation. Can we ignore this somehow?
Linear algebra review

$M$ is symmetric. Symmetric matrices have orthogonal eigenvectors (i.e., a basis).

$M$ is square. Square matrices are diagonalizable if some matrix $P$ exists s.t. $M = P^{-1}AP$, where $A$ has only diagonal entries and $P$ represents a change of basis (in 2D, a rotation).
Consider a horizontal “slice” of $E(u, v)$: \[ [u \ v] \ M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const} \]
This defines 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 a rotation matrix $R$.

Note inverse relationship: larger eigenvalue = steeper slope; smaller ellipse in visualization.
Classification of image points using eigenvalues of $M$

- **Corner**: $\lambda_1$ and $\lambda_2$ are large, $\lambda_1 \approx \lambda_2$; $E$ increases in all directions.

- **Edge**: $\lambda_2 >\gg \lambda_1$

- **Flat** region: $\lambda_1$ and $\lambda_2$ are small; $E$ is almost constant in all directions.

$\lambda_1$ and $\lambda_2$ are small; $E$ is almost constant in all directions.
Classification of image points using eigenvalues of $M$

Cornerness score:

$$C = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$

$\alpha$: some constant (~0.04 to 0.06)
Linear algebra review

Determinant of a diagonal matrix is the \textit{product} of all eigenvalues. \( A \) is diagonal.

Trace of a square matrix is the \textit{sum} of its diagonal entries; and is the sum of its eigenvalues.
Classification of image points using eigenvalues of $M$

**Cornerness score:**

$$C = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$

$\alpha$: some constant (~0.04 to 0.06)

**Remember your linear algebra:**

Determinant: $\text{det}(A) = \prod_{i=1}^{n} \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_n$.

Trace: $\text{tr}(A) = \sum \lambda_i$.

$$C = \text{det}(M) - \alpha \text{trace}(M)^2$$

Avoids explicit eigenvalue computation!
This is the Harris corner detector!

1) Compute $M$ matrix for each window to recover a \textit{cornerness} score $C$.
   
   Note: We can find $M$ purely from the per-pixel image derivatives!

2) Threshold to find pixels which give large corner response ($C >$ threshold).

3) Find the local maxima pixels, i.e., non-maximal suppression.

0. Input image
   We want to compute \( M \) at each pixel.

1. Compute image derivatives (optionally, blur first).

2. Compute \( M \) components as squares of derivatives.

3. Gaussian filter \( g() \) with width \( \sigma \)
   \[
   g(I_x^2), \ g(I_y^2), \ g(I_x \circ I_y)
   \]

4. Compute cornerness
   \[
   C = \det(M) - \alpha \ \text{trace}(M)^2
   = g(I_x^2) \circ g(I_y^2) - g(I_x \circ I_y)^2
   - \alpha [g(I_x^2) + g(I_y^2)]^2
   \]

5. Threshold on \( C \) to pick high cornerness

6. Non-maximal suppression to pick peaks.
Harris Detector: Steps
Harris Detector: Steps

Compute corner response $C$
Harris Detector: Steps

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

Take only the points of local maxima of $C$
Harris Detector: Steps
Live Harris Demo