Lecture 05
Point Feature Detection and Matching

Davide Scaramuzza
Lab Exercise 3 - Today afternoon

- Room ETH HG E 1.1 from 13:15 to 15:00
- Work description: implement the Harris corner detector and tracker
Outline

- Filters for Feature detection
- Point-feature extraction: today and next lecture
Filters for Feature Detection

• In the last lecture, we used filters to reduce noise or enhance contours

• However, filters can also be used to detect “features”
  – Goal: reduce amount of data to process in later stages, discard redundancy to preserve only what is useful (leads to lower bandwidth and memory storage)
    • Edge detection (we have seen this already; edges can enable line or shape detection)
    • Template matching
    • Keypoint detection
Filters for Template Matching

- Find locations in an image that are similar to a template
- If we look at filters as templates, we can use correlation (like convolution but without flipping the filter) to detect these locations
Filters for Template Matching

- Find locations in an image that are similar to a template
- If we look at filters as templates, we can use correlation (like convolution but without flipping the filter) to detect these locations
Where’s Waldo?
Where’s Waldo?

Scene

Template
Where’s Waldo?
Summary of filters

• **Smoothing filter:**
  – has positive values
  – sums to 1 → preserve brightness of constant regions
  – removes “high-frequency” components: “low-pass” filter

• **Derivative filter:**
  – has opposite signs used to get high response in regions of high contrast
  – sums to 0 → no response in constant regions
  – highlights “high-frequency” components: “high-pass” filter

• **Filters as templates**
  • Highest response for regions that “look similar to the filter”
Template Matching

- What if the template is not identical to the object we want to detect?
- Template Matching will only work if scale, orientation, illumination, and, in general, the appearance of the template and the object to detect are very similar. What about the pixels in template background (object-background problem)?
Consider images $H$ and $F$ as vectors, their correlation is:

$$
\langle H, F \rangle = \|H\| \|F\| \cos \theta
$$

In Normalized Cross Correlation (NCC), we consider the unit vectors of $H$ and $F$, hence we measure their similarity based on the angle $\theta$. If $H$ and $F$ are identical, then NCC = 1.

$$
\cos \theta = \frac{\langle H, F \rangle}{\|H\| \|F\|} = \frac{\sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u, v)F(u, v)}{\sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u, v)^2} \sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u, v)^2}}
$$
Other Similarity measures

• **Sum of Absolute Differences (SAD)** (used in optical mice)

\[
SAD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} |H(u,v) - F(u,v)|
\]

• **Sum of Squared Differences (SSD)**

\[
SSD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} (H(u,v) - F(u,v))^2
\]

• **Normalized Cross Correlation (NCC)**: takes values between -1 and +1 (+1 = identical)

\[
NCC = \frac{\sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v)F(u,v)}{\sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v)^2} \sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v)^2}}
\]
Zero-mean SAD, SSD, NCC

To account for the difference in mean of the two images (typically caused by illumination changes), we subtract the mean value of each image:

- **Zero-mean Sum of Absolute Differences (ZSAD)** (used in optical mice)
  \[
  ZSAD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \left| (H(u,v) - \mu_H) - (F(u,v) - \mu_F) \right|
  \]

- **Zero-mean Sum of Squared Differences (ZSSD)**
  \[
  ZSSD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \left( (H(u,v) - \mu_H) - (F(u,v) - \mu_F) \right)^2
  \]

- **Zero-mean Normalized Cross Correlation (ZNCC)**
  \[
  ZNCC = \frac{\sum_{u=-k}^{k} \sum_{v=-k}^{k} (H(u,v) - \mu_H)(F(u,v) - \mu_F)}{\sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} (H(u,v) - \mu_H)^2} \sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} (F(u,v) - \mu_F)^2}}
  \]

ZNCC is invariant to affine intensity changes: \[ I'(x,y) = \alpha I(x,y) + \beta \]
Census Transform

• Maps an image patch to a bit string:
  – if a pixel is greater than the center pixel its corresponding bit is set to 1, else to 0
  – For a $w \times w$ window the string will be $w^2 - 1$ bits long

• The two bit strings are compared using the Hamming distance, which is the number of bits that are different. This can be computed by counting the number of 1s in the Exclusive-OR (XOR) of the two bit strings

Advantages

• More robust to object-background problem

• No square roots or divisions are required, thus very efficient to implement, especially on FPGA

• Intensities are considered relative to the center pixel of the patch making it invariant to monotonic intensity changes
Outline

• Filters for feature extraction
• Point-feature (or keypoint) extraction: today and next lecture
Point-feature extraction and matching - Example

SVO with a single camera on Euroc dataset

Video from “Forster, Pizzoli, Scaramuzza, SVO: Semi-Direct Visual Odometry, ICRA’14”
What do we need point features for?

Recall the Visual-Odometry flow chart:

1. Image sequence
2. Feature detection
3. Feature matching (tracking)
4. Motion estimation (2D-2D, 3D-3D, 3D-2D)
5. Local optimization

Example of feature tracks
What do we need point features for?

Keypoint extraction is the key ingredient of motion estimation!

<table>
<thead>
<tr>
<th>Image sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature detection</td>
</tr>
<tr>
<td>Feature matching (tracking)</td>
</tr>
<tr>
<td><strong>Motion estimation</strong></td>
</tr>
<tr>
<td>2D-2D</td>
</tr>
<tr>
<td>Local optimization</td>
</tr>
</tbody>
</table>

$T_{k,k-1} = ?$
Point Features are also used for:

- Panorama stitching
- Object recognition
- 3D reconstruction
- Place recognition
- Indexing and database retrieval (e.g., Google Images or [http://tineye.com](http://tineye.com))
Image matching: why is it challenging?
Image matching: why is it challenging?

NASA Mars Rover images
Image matching: why is it challenging?

Answer below

NASA Mars Rover images with SIFT feature matches
Example: panorama stitching

This panorama was generated using AUTOSTITCH:
http://matthewalunbrown.com/autostitch/autostitch.html
Local features and alignment

• We need to align images
• How would you do it?
Local features and alignment

• Detect point features in both images
Local features and alignment

- Detect point features in both images
- Find corresponding pairs
Local features and alignment

- Detect point features in both images
- Find corresponding pairs
- Use these pairs to align the images
Matching with Features

• Problem 1:
  – Detect the **same** points **independently** in both images

  ![Images showing matching points with arrows indicating no match]

  **no chance to match!**

  We need a **repeatable** feature detector
Matching with Features

- Problem 2:
  - For each point, identify its correct correspondence in the other image(s)

We need a **reliable** and **distinctive** feature descriptor that is robust to *geometric* and *illumination* changes.
Geometric changes

- Rotation
- Scale (i.e., zoom)
- View point (i.e, perspective changes)
Illumination changes

Typically, small illumination changes are modelled with an affine transformation (so called *affine illumination changes*):

\[ I'(x, y) = \alpha I(x, y) + \beta \]
Invariant local features

Subset of local feature types designed to be invariant to common geometric and photometric transformations.

Basic steps:
1) Detect distinctive interest points
2) Extract invariant descriptors
Main questions

• What points are distinctive (i.e., features, keypoints, salient points), such that they are *repeatable*? (i.e., can be re-detected from other views)

• How to *describe* a local region?

• How to establish *correspondences*, i.e., compute matches?
What is a distinctive feature?

- Consider the image pair below with extracted patches
- Notice how some patches can be localized or matched with higher accuracy than others
Point Features: Corners vs Blob detectors

- A **corner** is defined as the intersection of one or more edges
  - A corner has high localization accuracy
  - Corner detectors are good for VO
  - It’s **less distinctive than a blob**
  - E.g., Harris, Shi-Tomasi, SUSAN, FAST

- A **blob** is any other image pattern, **which is not a corner**, that differs significantly from its neighbors in intensity and texture (e.g., a connected region of pixels with similar color, a circle, etc.)
  - Has less localization accuracy than a corner
  - Blob detectors are better for place recognition
  - It’s **more distinctive than a corner**
  - E.g., MSER, LOG, DOG (SIFT), SURF, CenSurE
The Moravec Corner detector (1980)

- How do we identify corners?
- We can easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity (e.g., in SSD) in at least 2 directions

“flat” region: no intensity change (i.e., SSD ≈ 0 in all directions)

“edge”: no change along the edge direction (i.e., SSD ≈ 0 along edge but ≫ 0 in other directions)

“corner”: significant change in at least 2 directions (i.e., SSD ≫ 0 in all directions)

Corner detection

• Key observation: in the region around a corner, image gradient has **two or more** dominant directions
• Corners are **repeatable** and **distinctive**

The Moravec Corner detector (1980)

“Sums of squares of differences of pixels adjacent in each of four directions (horizontal, vertical and two diagonals) over each window are calculated, and the window's interest measure is the minimum of these four sums.” [Moravec’80, Ch. 5]

The Harris Corner detector (1988)

- It implements the Moravec corner detector without having to physically shift the window but rather by just looking at the patch itself, by using differential calculus.

How do we implement this?

- Consider the reference patch centered at \((x, y)\) and the shifted window centered at \((x + \Delta x, y + \Delta y)\). The patch has size \(P\).

- The Sum of Squared Differences between them is:

\[
SSD(\Delta x, \Delta y) = \sum_{x, y \in P} (I(x, y) - I(x + \Delta x, y + \Delta y))^2
\]

- Let \(I_x = \frac{\partial I(x, y)}{\partial x}\) and \(I_y = \frac{\partial I(x, y)}{\partial y}\). Approximating with a 1\(^{\text{st}}\) order Taylor expansion:

\[
I(x + \Delta x, y + \Delta y) \approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y
\]

- This produces the approximation

\[
SSD(\Delta x, \Delta y) \approx \sum_{x, y \in P} \left( I_x(x, y)\Delta x + I_y(x, y)\Delta y \right)^2
\]

This is a simple quadratic function in two variables \((\Delta x, \Delta y)\)
How do we implement this?

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} \left( I_x(x, y) \Delta x + I_y(x, y) \Delta y \right)^2$$

- This can be written in a matrix form as

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$\Rightarrow SSD(\Delta x, \Delta y) \approx \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$M = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
How do we implement this?

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} \left( I_x(x, y) \Delta x + I_y(x, y) \Delta y \right)^2$$

This can be written in a matrix form as

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} [\Delta x \quad \Delta y] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$\Rightarrow SSD(\Delta x, \Delta y) \approx [\Delta x \quad \Delta y] M [\Delta x \quad \Delta y]$$

Notice that these are NOT matrix products but pixel-wise products!

$$M = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$
What does this matrix reveal?

• First, consider an edge or a flat region.

\[
M = \begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \lambda_2 \end{bmatrix}
\]

Edge

Flat region

• We can conclude that if either \( \lambda \) is close to 0, then this is not a corner.

• Now, let’s consider an axis-aligned corner:

\[
M = \begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}
\]

Corner

• This means dominant gradient directions are at 45 degrees with \( x \) and \( y \) axes

• What if we have a corner that is not aligned with the image axes?
General Case

Since $M$ is symmetric, it can always be decomposed into $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

- We can visualize $[\Delta x \ \Delta y]M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \text{const}$ as an ellipse with axis lengths determined by the eigenvalues and the two axes' orientations determined by $R$ (i.e., the eigenvectors of $M$)

- The two eigenvectors identify the directions of largest and smallest changes of SSD

\[ \frac{\lambda_{\text{max}}}{-\frac{1}{2}} \quad \frac{\lambda_{\text{min}}}{-\frac{1}{2}} \]

\[ \text{direction of the fastest change of SSD} \quad \text{direction of the slowest change of SSD} \]
How to compute $\lambda_1, \lambda_2, R$ from $M$

Eigenvalue/eigenvector review

• You can easily prove that $\lambda_1, \lambda_2$ are the **eigenvalues** of $M$.
• The **eigenvectors** and **eigenvalues** of a square matrix $A$ are the vectors $x$ and scalars $\lambda$ that satisfy:

$$Ax = \lambda x$$

• The scalar $\lambda$ is the **eigenvalue** corresponding to $x$
  – The eigenvalues are found by solving:  
    $$\det(A - \lambda I) = 0$$
  – In our case, $A = M$ is a 2x2 matrix, so we have 
    $$\det\begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix} = 0$$
  – The solution is: 
    $$\lambda_{1,2} = \frac{1}{2} \left[ (m_{11} + m_{22}) \pm \sqrt{4m_{12}m_{21} + (m_{11} - m_{22})^2} \right]$$
  – Once you know $\lambda$, you find the two eigenvectors $x$ (i.e., the two columns of $R$) by solving:
    $$\begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$
Visualization of $2^{nd}$ moment matrices
Visualization of 2\textsuperscript{nd} moment matrices

NB: the ellipses here are plotted proportionally to the eigenvalues and not as iso-SSD ellipses as explained before. So small ellipses here denote a flat region, and big ones, a corner.
Interpreting the eigenvalues

• Classification of image points using eigenvalues of $M$
• A corner can then be identified by checking whether the minimum of the two eigenvalues of $M$ is larger than a certain user-defined threshold
  \[ R = \min(\lambda_1, \lambda_2) > \text{threshold} \]
• $R$ is called “cornerness function”
• The corner detector using this criterion is called «Shi-Tomasi» detector

J. Shi and C. Tomasi (June 1994). "Good Features to Track," 9th IEEE Conference on Computer Vision and Pattern Recognition

$\lambda_1$ and $\lambda_2$ are small; $SSD$ is almost constant in all directions

$\lambda_1$ and $\lambda_2$ are large,
\[ \Rightarrow R > \text{threshold} \]
\[ \Rightarrow \text{SSD increases in all directions} \]
Interpreting the eigenvalues

- Computation of $\lambda_1$ and $\lambda_2$ is expensive $\Rightarrow$ Harris & Stephens suggested using a different cornerness function:
  \[ R = \lambda_1 \lambda_2 - k (\lambda_1 + \lambda_2)^2 = \det(M) - k \text{trace}^2(M) \]

- $k$ is a magic number in the range (0.04 to 0.15)

\[ \lambda_2 \]

\[ \lambda_1 > > \lambda_2 \]

- "Corner" $\lambda_1$ and $\lambda_2$ are large,
  $\Rightarrow$ R $>$ threshold
  $\Rightarrow$ SSD increases in all directions

- "Edge" $\lambda_1 > > \lambda_2$

- "Flat region"
Harris Corner Detector

Algorithm:
1. Compute derivatives in $x$ and $y$ directions ($I_x, I_y$) e.g. with Sobel filter
2. Compute $I_x^2, I_y^2, I_x I_y$
3. Convolve $I_x^2, I_y^2, I_x I_y$ with a box filter to get $\sum I_x^2, \sum I_y^2, \sum I_x I_y$, which are the entries of the matrix $M$ (optionally use a Gaussian filter instead of a box filter to avoid aliasing and give more “weight” to the central pixels)
4. Compute Harris Corner Measure $R$ (according to Shi-Tomasi or Harris)
5. Find points with large corner response ($R > \text{threshold}$)
6. Take the points of local maxima of $R$
Harris Corner Detector

Image $I$  

Cornerness response $R$
Harris vs. Shi-Tomasi

Harris operator

Shi-Tomasi operator

53
Harris Detector: Workflow
Harris Detector: Workflow

• Compute corner response $R$
Harris Detector: Workflow

- Find points with large corner response: $R > threshold$
Harris Detector: Workflow

• Take only the points of local maxima of thresholded $R$ (non-maxima suppression)
Harris Detector: Workflow
Harris Detector: Some Properties

How does the size of the Harris detector affect the performance?

Repeatability:
• How does the Harris detector behave to common image transformations?
• Can it re-detect the same image patches (Harris corners) when the image exhibits changes in
  • Rotation,
  • View-point,
  • Scale (zoom),
  • Illumination?

• Solution: Identify properties of detector & adapt accordingly
Harris Detector: Some Properties

- Rotation invariance

Corner response $R$ is \textit{invariant to image rotation}
Harris Detector: Some Properties

• But: non-invariant to **image scale**!

All points will be classified as **edges**
Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability = \[
\frac{\text{# correspondences detected}}{\text{# correspondences present}}
\]

Scaling the image by \( \times 2 \)\n\( \Rightarrow \) \( \sim 18\% \) of correspondences get matched
Summary (things to remember)

- Filters as templates
- Correlation as a scalar product
- Similarity metrics: NCC (ZNCC), SSD (ZSSD), SAD (ZSAD), Census Transform
- Point feature detection
  - Properties and invariance to transformations
    - Challenges: rotation, scale, view-point, and illumination changes
  - Extraction
    - Moravec
    - Harris and Shi-Tomasi
      - Rotation invariance
- Reading:
  - Ch. 4.1 and Ch. 8.1 of Szeliski book
  - Ch. 4 of Autonomous Mobile Robots book
  - Ch. 13.3 of Peter Corke book
Understanding Check:

Are you able to:
• Explain what is template matching and how it is implemented?
• Explain what are the limitations of template matching? Can you use it to recognize cars?
• Illustrate the similarity metrics SSD, SAD, NCC, and Census transform?
• What is the intuitive explanation behind SSD and NCC?
• Explain what are good features to track? In particular, can you explain what are corners and blobs together with their pros and cons?
• Explain the Harris corner detector? In particular:
  – Use the Moravec definition of corner, edge and flat region.
  – Show how to get the second moment matrix from the definition of SSD and first order approximation (show that this is a quadratic expression) and what is the intrinsic interpretation of the second moment matrix using an ellipse?
  – What is the $M$ matrix like for an edge, for a flat region, for an axis-aligned 90-degree corner and for a non-axis—aligned 90-degree corner?
  – What do the eigenvalues of $M$ reveal?
  – Can you compare Harris detection with Shi-Tomasi detection?
  – Can you explain whether the Harris detector is invariant to illumination or scale changes?
  – What is the repeatability of the Harris detector after rescaling by a factor of 2?