Computer Vision II: Multiple View Geometry (IN2228)

Chapter 05 Correspondence Estimation
(Part 2 Large Motion)

Dr. Haoang Li

17 May 2023  12:00-13:30
Announcement

Updated Lecture Schedule

For updates, slides, and additional materials: https://cvg.cit.tum.de/teaching/ss2023/cv2

90-minute course; 45-minute course

- Wed 19.04.2023  Chapter 00: Introduction
- Thu 20.04.2023  Chapter 01: Mathematical Backgrounds
- Wed 26.04.2023  Chapter 02: Motion and Scene Representation (Part 1)
- Thu 27.04.2023  Chapter 02: Motion and Scene Representation (Part 2)
- Wed 03.05.2023  Chapter 03: Image Formation (Part 1)
- Thu 04.05.2023  Chapter 03: Image Formation (Part 2)
- Wed 10.05.2023  Chapter 04: Camera Calibration
- Thu 11.05.2023  Chapter 05: Correspondence Estimation (Part 1)
- Wed 17.05.2023  Chapter 05: Correspondence Estimation (Part 2)
- Thu 18.05.2023  No lecture (Public Holiday)
- Wed 24.05.2023  No lecture (Conference)
- Thu 25.05.2023  No lecture (Conference)

- Wed 31.05.2023  Chapter 06: 2D-2D Geometry (Part 1)
- Thu 01.06.2023  Chapter 06: 2D-2D Geometry (Part 2)
- Wed 07.06.2023  Chapter 06: 2D-2D Geometry (Part 3)
- Thu 08.06.2023  No lecture (Public Holiday)
- Wed 14.06.2023  Chapter 07: 3D-2D Geometry (Part 1)
- Thu 15.06.2023  Chapter 07: 3D-2D Geometry (Part 2)
- Wed 21.06.2023  Chapter 08: 3D-3D Geometry (Part 1)
- Thu 22.06.2023  Chapter 08: 3D-3D Geometry (Part 2)
- Wed 28.06.2023  Chapter 09: Single-view Geometry (Part 1)
- Thu 29.06.2023  Chapter 09: Single-view Geometry (Part 2)
- Wed 05.07.2023  Chapter 10: Photometric Error (Direct Method)
- Thu 06.07.2023  Chapter 11: Bundle Adjustment and Optimization
- Wed 12.07.2023  Chapter 12: Robot Estimation
- Thu 13.07.2023  Knowledge Review
- Wed 19.07.2023  Chapter 13: SLAM and SFM (Part 2)
- Thu 20.07.2023  Chapter 13: SLAM and SFM (Part 1)

Videos and reading materials about the combination of deep learning and multi-view geometry

Foundation

Core part

Advanced topics and high-level task
Announcement

➢ Updated Lecture Schedule

Though we do not have lectures on 24 and 25 May, the exercise session will still be held normally on 24 May.

Videos and reading materials about the combination of deep learning and multi-view geometry

Tentative Exercise Schedule

Wed 24.05.2023  Exercise 4: Perspective Projection
Wed 31.05.2023  Exercise 5: Lucas-Kanade Method
Wed 14.06.2023  Exercise 6: Reconstruction from two views
Wed 21.06.2023  Exercise 7: Reconstruction from multiple views
Wed 05.07.2023  Exercise 8: Direct Image Alignment
Wed 12.07.2023  Exercise 9: Direct Image Alignment
Explanations for Previous Knowledge

Which Points Should We Track?

- Theoretical strategy
  
  \[ M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \]
  
  Eigenvectors
  
  \[ R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \]
  
  Eigenvalues

  \( I_x \) and \( I_y \): Directional derivatives

- Practical solution

  Only first image: judge each pixel

  Subsequent images: only consider the tracked points

Problem: relatively low efficiency due to judgement on all the pixels in an image
Explanations for Previous Knowledge

- **Brightness Consistency**

- Video $I$ (sequential images) is w.r.t. the time $t$. A tracked point’s position $x$ is also w.r.t. the time $t$.

- We use the pixel coordinate system whose origin is located at the top-left of image.

$$I(x(t), t) = \text{const.} \quad \forall t,$$

Frames obtained by a static camera

Sequential images

Moving object

A tracked point

Moving camera
Explanations for Previous Knowledge

- Chicken-and-egg Problem

✓ Definition
A problem has two sets of unknown parameters. Parameters are mutually determined.

✓ Examples

\[
x' = x\cos(a_3) - y\sin(a_3) + a_1
\]
\[
y' = x\sin(a_3) + y\cos(a_3) + a_2
\]

\{(x,y) \text{ and } (x',y') \} \text{ constitute a pair of unknown-but-sought correspondences}

\[p = (a_1, a_2,...,a_n)\] \text{ are warping parameters to estimate}
Explanations for Previous Knowledge

- Chicken-and-egg Problem

✓ Solution
Find additional constraint to solve parameters.

\[
SSD = \sum_{x \in T} \left[ I(W(x, p)) - T(x) \right]^2
\]

Additional constraint: Brightness consistency
Today’s Outline

- Overview of Indirect Method
- Feature Detector
- Feature Descriptor
Overview of Indirect Method

- Recap on Problem Formulation

✓ Core idea: To compute the transformation between two images, we resort to detecting and matching features (points or lines).
✓ **Pros**: They can cope with large frame-to-frame motions and strong illumination changes.
✓ **Cons**: They are slow due to costly feature extraction, matching, and outlier removal (e.g., RANSAC).

Direct method (KLT):
A one-step strategy

Indirect method: A two-step strategy

Feature detection
Feature matching
Transformation estimation
Overview of Indirect Method

Recap on Problem Formulation

✓ Pipeline for alignment between template and current images
1. Detect and match features that are invariant to scale, rotation, viewpoint changes (e.g., SIFT)

2. Transformation computation and Geometric verification (e.g., 4 point RANSAC for planar objects, or 5 or 8 point RANSAC for 3D objects)

3. Refine the initial estimation by minimizing the sum of squared reprojection errors between the observed feature $f^i$ in the current image and the warped corresponding feature $W(x^i, p)$ from the template

\[ p = \underset{p}{\text{argmin}} \sum_{i=1}^{N} \| W(x^i, p) - f^i \|^2 \]

Geometric feature (point) distance
Feature Detector

Definition of Blob

- A blob is a group of **connected pixels** in an image that share some common property (e.g., grayscale value).
- In the image below, the colored regions are blobs, and **blob detection** aims to identify and mark these regions.
Feature Detector

- Comparison between Corner and Blob

- A **corner** is defined as the intersection of two or more edges
  - Corners have **high localization** accuracy
  - Corners are **less distinctive than blobs**
  - E.g., Moravec, Harris, Shi-Tomasi, SUSAN, FAST

- **Blob** introduced before
  - Blobs have less localization accuracy than corners
  - Blobs are more distinctive than corners
  - E.g., MSER, LOG, DOG (SIFT), SURF, CenSurE, etc.

Methods shown in blue will be mainly discussed
Feature Detector

- Corner Detection

- Key observation: in the region around a corner, the image gradient has multiple dominant directions (e.g., vertical, horizontal, and diagonal).

- **Shifting a window** in any direction should cause large intensity changes around a corner.

---

“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 all directions 
(i.e., SSD >> 0 in all directions)
Feature Detector

- Corner Detection

- We use Sum of Squared Differences (SSD) to measure the brightness change.
- Consider the reference patch and a patch shifted by \((\Delta x, \Delta y)\). The Sum of Squared Differences between them is

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

SSD along the direction along the diagonal

A pair of corresponding pixels

A shifted patch

Reference patch
Feature Detector

- Corner Detection

- Moravec’s method
  - For two patches, compute the sum of squared differences (SSD) between all pairs of corresponding pixels.
  - A lower SSD indicates higher similarity between two patches.
  - Consider SSD along multiple directions. The interest measurement of a patch is defined as the smallest SSD.
  - If a patch’s interest measurement is higher than a threshold, the patch center is a corner.

- We have to physically shift the window. Can we make it more efficient?
Feature Detector

- Corner Detection

✓ Approximating with a 1st order Taylor expansion (introduced before):

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

We no longer minimize SSD but just measure SSD here!

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

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

This is a quadratic function in two variables (\(\Delta x, \Delta y\))
Feature Detector

- Corner Detection

✓ Matrix form of approximation result

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

\[
SSD(\Delta x, \Delta y) \approx \sum_{x, y \in \Omega} \left[ \Delta x \begin{bmatrix} I_x^2 & I_x I_y & \Delta x \\ I_x I_y & I_y^2 & \Delta y \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right]
\]

\[
= \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}
\]

This matrix encodes the SSD change

\[
M = \begin{bmatrix} \sum l_x l_x & \sum l_x l_y \\ \sum l_x l_y & \sum l_y l_y \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R
\]

- Eigenvalues
- Eigenvectors
- \(I_x\) and \(I_y\): Directional derivatives
- Direction of quickest change of SSD
- Direction of the slowest change of SSD
Feature Detector

- Corner Detection

✓ Conclusion
If both eigenvalues are much larger than 0 then we have a corner.

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

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

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \cos \frac{\pi}{4} & -\sin \frac{\pi}{4} \\ \sin \frac{\pi}{4} & \cos \frac{\pi}{4} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos \frac{\pi}{4} & \sin \frac{\pi}{4} \\ -\sin \frac{\pi}{4} & \cos \frac{\pi}{4} \end{bmatrix}$$

Representative results
Feature Detector

- Corner Detection

- The above method **without explicitly shifting patch** is called **Harris detector**
- The Harris detector is not scale invariant (same patch size is not applicable to different scales of images)

An example: we apply the same window/patch size to two images with different scales

All points will be classified as *edges*
Feature Descriptor

- Feature Matching Based on Descriptor

✓ Given a detected point in $I_1$, how to find the best match in $I_2$?

- Define point descriptors, e.g., Census, HOG, ORB, BRIEF, BRISK, FREAK.
- Define distance function that compares two descriptors, e.g., SSD, SAD, NCC or Hamming distance.
Feature Descriptor

Feature Matching Based on Descriptor

✔ A naive matching strategy: Brute force matching
• Here, we assume that detected points, point descriptions are both known.
• Compare each feature in $I_1$ against all the features in $I_2$ ($N^2$ comparisons, where $N$ is the number of features in each image).
• Select the point pair with the minimum distance, i.e., the closest description.
Feature Descriptor

- Feature Matching Based on Descriptor

✓ Issue with closest descriptor:
Algorithm can occasionally return good scores for false matches
A solution: compute ratio of distances to 1st to 2nd closest descriptor

\[
\frac{d_1}{d_2} < \text{Threshold (usually 0.8)}
\]

Distinctive enough

where:
\(d_1\) is the distance from the closest descriptor
\(d_2\) is the distance of the 2nd closest descriptor

Unreliable points due to repetitive pattern

Distinctive points
Feature Descriptor

- Distance Function Definition

- Similarity measurement (applicable to both 2D and 1D)
  - **Sum of Squared Differences (SSD):** always $\geq 0$. It’s exactly 0 only if $H$ and $F$ are identical
    \[
    SSD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} (H(u,v) - F(u,v))^2
    \]
  - **Sum of Absolute Differences (SAD):** always $\geq 0$. It’s 0 only if $H$ and $F$ are identical
    \[
    SAD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} |H(u,v) - F(u,v)|
    \]

$H$ and $F$ denote left and right patches/descriptors respectively.

2D patches

1D descriptors
Feature Descriptor

- Distance Function Definition

✓ Similarity measurement

- Normalized Cross Correlation (NCC): ranges between -1 and +1 and is exactly 1 if \(H\) and \(F\) are identical

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

\(1,1\) and \((1,1)\):
\[NCC = 1\]

\(1,1\) and \((-1,-1)\):
\[NCC = -1\]

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

\[
\mu_H = \frac{1}{n} \sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v) \quad \mu_F = \frac{1}{n} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v)
\]

\[
ZSSD = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \left( (H(u,v) - \mu_H) - (F(u,v) - \mu_F) \right)^2
\]

Zero-mean
Feature Descriptor

Properties of Descriptor

- Distinctiveness of a feature descriptor
  - A descriptor is a “description” of the pixel information **around** a feature.
  - “Distinctiveness” means that the descriptor can uniquely distinguish a feature from the other features **without ambiguity**.

- Robustness to geometric changes
  - Scale-invariant (for zooming)
  - Rotation-invariant
  - View point-invariant (for perspective changes)
Feature Descriptor

Properties of Descriptor

✓ Robustness to illumination changes
  • Small illumination changes are modelled with an affine transformation (so called affine illumination changes) changes:

\[ I'(x, y) = \alpha I(x, y) + \beta \]
Feature Descriptor

Traditional Method based on Patch Feature

- General pipeline
  - Determine the scale, rotation and viewpoint change of each patch *(introduced later)*.
  - Warp each patch into a canonical patch.
  - Establish patch correspondences based on similarity of the warped patch.
Feature Descriptor

➢ Scale of Descriptor

✓ Problem formulation
Two image patches have the same size, but are in the images with different scales. How can we match these patches?

Images with different scales

Two patches with the same size
Feature Descriptor

- Scale of Descriptor

✓ A possible solution
Keep the **left patch unchanged, and resize the right patch** with different tentative sizes.
Feature Descriptor

Scale of Descriptor

- Patch size search is time-consuming
  - We need to individually re-size all the patches in the right image.
  - Algorithm complexity is $N^2S$ assuming $N$ features per image and $S$ tentative sizes per feature.

- A better solution of automatic size selection: We aim to “automatically” assign each patch (both left and right) its own size.
Feature Descriptor

- Scale of Descriptor

- Overview of automatic size selection
  - Core idea: Assign a function to the image patch. The function extremum is scale invariant.
  - Try candidate scales in each image independently

- When the left patch is resized by $s_1$, and the right patch is resized by $s_2$, their associated functions both achieve (the same) extremum. $s_1$ and $s_2$ are our automatically determined sizes of patches.
- We care about which scale leads to the extremum. The value of extremum is not important.
Feature Descriptor

➤ Scale of Descriptor

✓ An example of automatic scale selection

• Patches with the same size correspond to different function values.

• Intuitively, only if two patches correspond to the same 3D area, their associated extreme values of function are the same.
Feature Descriptor

- Scale of Descriptor

- An example of automatic scale selection
  
  - Two patches with different sizes correspond to the same 3D areas. These two patches lead to the same extreme value of function.
  
  - Note: We determine the scale of a patch in each image independently. We try a set of candidate scales for a patch. The scale leading to extremum is the optimal scale.

What scale-invariant function should we assign to patch?
Feature Descriptor

- Scale of Descriptor

✓ Function properties
  - A “good” function for scale detection should have a single & sharp peak
  - **Sharp intensity changes** are good regions to monitor in order to identify the scale

- In our context, for ease of understanding, we do not consider a more complex case (multiple extrema)
Feature Descriptor

- Scale of Descriptor
  
  ✓ Function selection
  
  - Intuitively, a human can identify the scale (patch sizes) by comparing the areas with sharp brightness discontinuities. (we can easily identify the same 3D area at different scales)
  
  - Therefore, the ideal function for determining the scale should be able to highlight sharp discontinuities.
  
  - Solution: convolve image patch with a kernel that highlights edges

\[
f = \text{Kernel} \ast \text{Image}
\]
Feature Descriptor

- Scale of Descriptor

✓ Function selection

• Recap on Laplace operator

Apply Laplace operator to a pixel

\[
\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)
\]

Laplace operator highlights the pixels with sharp intensity discontinuity (e.g., edge pixels)
Feature Descriptor

- Scale of Descriptor

- Function selection

- Extension to the blob

It has been shown that the Laplacian of Gaussian kernel is optimal under certain assumptions [1]

- The Laplacian of Gaussian is a circularly symmetric filter defined as:

\[
\text{LoG}(x, y, \sigma) = \nabla^2 G_\sigma(x, y) = \frac{\partial^2 G_\sigma(x, y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x, y)}{\partial y^2}
\]

\(\nabla^2\) is the Laplacian operator: 

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

Feature Descriptor

➢ Scale of Descriptor

✓ Function selection

At what scale does the Laplacian achieve a maximum response (extremum) to a binary circle of radius r?

• To get the maximum response, the Laplacian has to be aligned to the circle.
• The maximum response occurs at $\sigma = r/\sqrt{2}$.
• A simplified but not general case with known circle. Generally, we have to test a set of candidate scales.

Circle pixels with low intensity (fixed)
Feature Descriptor

- Scale of Descriptor

- Scale determination

  - An ideal alignment between blob and LoG leads to a extrema
  - The correct scale is found at the local extremum
Feature Descriptor

- Scale of Descriptor

✓ Post-processing
When the right scale is found, the patches must be normalized to a canonical size so that they can be compared by SSD. Patch normalization is done via warping.
Feature Descriptor

- Rotation of Descriptor

- Determining patch orientation
  Eigenvectors of M matrix of Harris (introduced before) or dominant gradient direction (see next slide)

- Back-rotating patch through “patch warping”
  This step puts the patches back into a canonical orientation
Feature Descriptor

- Rotation of Descriptor

✓ A general pipeline to express patch orientation
  • Compute gradients vectors at each pixel within a patch
  • Build a histogram of gradient orientations, weighted by the gradient magnitudes (norm of vector).
  • Extract all local maxima in the histogram: each local maximum above a threshold is a candidate dominant orientation.

One patch with multiple pixels
Feature Descriptor

- **Viewpoint of Descriptor**

  - **Affine warping provides invariance to small viewpoint changes**
    - The second moment **matrix** $M$ of the Harris detector can be used to identify the two directions of fastest and slowest change of SSD around the feature.
    - Out of these two directions, an ellipse-shaped patch is extracted.
    - The region inside the ellipse is **normalized** to a canonical **circular patch**.

  ![Diagram of Feature Descriptor](image)

  - $M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = R^{-1} \begin{bmatrix} \hat{\lambda}_1 & 0 \\ 0 & \hat{\lambda}_2 \end{bmatrix} R$
  - $I_x$ and $I_y$: Directional derivatives.
  - $\lambda_1$ and $\lambda_2$: Eigenvalues.
  - Eigenvectors of $M$.
  - $R$: Rotation matrix.
  - Direction of quickest change of SSD.
  - Direction of slowest change of SSD.

  ![Affine transformation](image)
Feature Descriptor

Summary

✓ Scale, rotation, and affine-invariant patch matching

1. Scale assignment: compute the scale using the LoG operator. (scale-invariant)
2. Rotation assignment: use Harris or gradient histogram to find dominant orientation. (rotation-invariant)
3. View point transformation: use Harris eigenvectors to extract affine transformation parameters. (view point-invariant)
4. Warp the patch into a canonical patch
Feature Descriptor

- Disadvantages of Patch Feature-based Method

- If the warping is not estimated accurately, very small errors in rotation, scale, and viewpoint will affect matching score (e.g., SSD of patch features).

Is there a better strategy without directly using patch feature? Census feature (introduced in the next class).
Summary

- Overview of Indirect Method
- Point Detector
- Point Descriptor
Thank you for your listening!
If you have any questions, please come to me :-}