Lecture 6 & 7
Matching, Alignment, and Mosaicing

Yonghao Lee

December 9, 2025

Contents

1 Part I: Feature Detection and Matching

1.1 The Matching Pipeline

To align images (e.g., for panoramas) or recognize objects, we follow a standard pipeline:

1. **Detection:** Find interesting points (keypoints) in the image.
2. **Description:** Build a descriptor (fingerprint) for each point.
3. **Matching:** Find corresponding pairs between images.
4. **Alignment:** Use these pairs to align images (e.g., compute Homography).

A good descriptor must be **Invariant** (to rotation, scale, lighting) and **Distinctive** (unique enough to match correctly).

1.2 Feature Descriptors

1.2.1 MOPS (Multi-Scale Oriented Patches)

MOPS is a simpler descriptor that uses raw pixel intensities.

- **Detection:** Uses Multi-scale Harris Corners.
- **Scale Invariance:** Extracts regions from a blurred image pyramid at scale s.
- **Rotation Invariance:** Detects the dominant orientation (θ) and rotates the patch to align with it.
- **The Descriptor:**
  1. Extract a 40×40 pixel patch centered at the keypoint.
  2. Downsample it to a tiny 8×8 patch (low resolution).
  3. **Intensity Normalization:** Normalize pixels ($I' = \frac{I-\mu}{\sigma}$) to handle lighting changes.

1.2.2 SIFT (Scale-Invariant Feature Transform)

SIFT is a robust descriptor based on gradient statistics.

- **Detection:** Uses Difference of Gaussians (DoG) extrema to find (x, y, s).
- **Canonical Orientation:** Computes a gradient direction histogram. The peak determines orientation.
- **The Descriptor:**
  1. Take a 16×16 region around the keypoint.
  2. Divide into a 4×4 grid of sub-regions.
  3. Compute a histogram of gradients with **8 bins** for each sub-region.
  4. **Dimensionality:** $4 \times 4 \times 8 = \mathbf{128}$ dimensions.
- **Illumination:** Normalized to unit length; values > 0.2 are clipped to handle glare.

1.3 Matching Strategy

The Ratio Test (Lowe's Ratio): To reject ambiguous matches, compare the distance to the nearest neighbor (1-NN) vs. the second nearest neighbor (2-NN).

$$\text{Ratio} = \frac{\text{Distance(1-NN)}}{\text{Distance(2-NN)}} < \text{Threshold} \tag{1}$$

1.4 RANSAC (Robust Model Fitting)

Standard Least Squares fails when outliers (wrong matches) are present. RANSAC is an iterative method to ignore outliers.

Algorithm 1 RANSAC Algorithm

1: **repeat**
2: **Sample:** Randomly select s points (min required for model).
3: **Model:** Solve for parameters using these s points.
4: **Score:** Count Inliers (points fitting this model within threshold δ).
5: **until** Max Iterations reached
6: **Refine:** Re-compute model using Least Squares on **all inliers** of the best model.

2 Part II: Image Alignment and Mosaicing

2.1 Motion Models

To stitch images, we must define the mathematical relationship between them.

- **Translation (2 DOF):** Shifting coordinates ($x' = x + t_x$). Only valid for flat scenes perpendicular to the camera.
- **Affine (6 DOF):** Includes translation, rotation, scale, and shear. Preserves parallelism.
- **Homography (8 DOF):** The most general planar transformation. Maps straight lines to straight lines but does not preserve parallelism (creates perspective convergence).

2.2 Homography on Points vs. Lines

A Homography H maps points from one view to another.

- **Points:** A point x transforms via matrix multiplication:

$$x' = Hx \tag{2}$$

- **Lines:** A line l is defined such that $l^T x = 0$. If points transform by H, the line must transform differently to maintain orthogonality. The transformation is the **Inverse Transpose**:

$$l' = H^{-T} l \tag{3}$$

2.3 Building Panoramas

To build a panorama, we rotate a camera around its optical center.

- **Pure Rotation:** If the camera center does not move, there is no parallax. The relationship between any two images is a Homography.
- **Planar Projection:** Warping images onto a flat plane works for small angles but causes massive distortion ("explosion") as the Field of View (FOV) approaches 90°.
- **Cylindrical Projection:** For 360° panoramas, we project images onto a cylinder.
  - Straight lines in the world become curved (conic sections) on the cylinder.
  - Alignment becomes a simple translation in the cylindrical coordinate space (images just slide sideways).

2.4 Video Mosaicing (Strip Mosaicing)

Instead of stitching full photos, we can use video to create panoramas.

- **Method:** Cut a narrow vertical strip from the center of each video frame and paste them together.
- **Why center strips?** The center of the lens has the least distortion.
- **Manifold Projection:** This effectively creates a "pushbroom" camera view.
- **Stereo Mosaicing:** If the camera rotates off-axis (not around the center), parallax is introduced. We can create two panoramas:
  - Left-looking strips → Left eye view.
  - Right-looking strips → Right eye view.

3 Part III: Compositing and Seam Finding

Once images are aligned, they must be blended to hide the transition.

3.1 The Problems

- **Exposure Differences:** One image is brighter than the other.
- **Misalignment:** Slight errors in Homography calculation.
- **Ghosting:** Moving objects (e.g., a person walking) appear in both images, creating semi-transparent ghosts.

3.2 Approach A: Blending (Smoothing)

Blending attempts to smooth the transition region.

1. **Alpha Blending (Feathering):** A weighted average where the weight α transitions linearly from 1 to 0 across the overlap.

$$I_{final} = \alpha I_{left} + (1 - \alpha) I_{right} \tag{4}$$

   Drawback: Can cause blurring if alignment is imperfect.

2. **Pyramid Blending:** Decomposes the image into frequency bands (Laplacian Pyramid).
   - **Low Frequencies (Color/Lighting):** Blended over a wide range.
   - **High Frequencies (Details/Edges):** Blended over a very narrow range.

   This preserves sharp details while smoothing out exposure differences.

3.3 Approach B: Optimal Seam Finding (Cutting)

For moving objects, blending causes ghosting. The solution is to find a jagged **cut** that avoids the moving object entirely. Ideally, the cut passes through regions where the two images are identical.

3.3.1 Dynamic Programming

Finds a seam from top to bottom by minimizing a cumulative cost function.

- **Cost Calculation:** $E(i, j) = (I_1(i, j) - I_2(i, j))^2 + \min(\text{neighbors from row above})$.
- **Process:** Forward pass to calculate costs, Backward pass to trace the optimal path.
- **Limitation:** Can only handle simple paths (cannot loop back or handle complex shapes).

3.3.2 Graph Cuts (Min-Cut / Max-Flow)

A more robust method that treats the image as a graph.

- **Nodes:** Pixels in the overlap region.
- **Terminals:** Source (S) represents Image A; Sink (T) represents Image B.
- **Edge Weights:** Represent the "cost" of cutting between pixels.
  - High difference ($|I_A - I_B|$ is large) $\rightarrow$ High weight (Hard to cut).

  - Low difference (Identical pixels) → Low weight (Easy to cut).

- **Min-Cut Theorem:** The minimum cut in the graph corresponds to the optimal seam that separates Image A from Image B while passing through the pixels with the least visual difference. This effectively routes the seam *around* moving objects.