Recap

Gradient Histogram

A very powerful idea that is the basis for many image & feature descriptors.

Why? Hint: invariant in more than one way...

Look at a (local) region and calculate local gradients, then bin the gradients in a histogram.

How to get direction and magnitude of edges?

\[ m(x, y) = \sqrt{(L_x(x + 1, y) - L_x(x - 1, y))^2 + (L_y(x, y + 1) - L_y(x, y - 1))^2} \]

\[ \theta(x, y) = \tan^{-1} \left( \frac{L_y(x, y)}{L_x(x, y)} \right) \]

Most can be pre-calculated over the entire image with a single convolution.

Feature Matching

Given features of image 1 and of image 2, how do we find the best matching of the two sets?

- Define a distance metric between a pair of feature descriptors
- Find the best matching.

Some metrics are determined by the descriptor (Hamming, \(L_1\) norm).

\[ M(f_L, f_R) = ||f_L - f_R||_{L_2} < t \]

What is an optimal matching?

Naive: Pick the nearest neighbor (NN), set a threshold on distance.
Today

Feature Matching
Image Alignment
Panoramas
HW2!
Feature Matches
Feature Matching

Given features of image 1 and of image 2, how do we find the best matching of the two sets?

- Define a distance metric between a pair of feature descriptors
- Find the best matching.

**Naive:** Pick the nearest neighbor (NN), set a threshold on distance.

\[ M(f_L, f_R) = \| f_L - f_R \|_{L_2} < t \]

Apply: Ratio test and Reciprocity

SIFT matching, ratio, recip., and object finding
Receiver Operator Characteristic (ROC) Curve

- Vary the threshold $t$
- Calculate precision and recall
- Take a $t$ that has highest TP-rate, lowest FP-rate.

Precision: $\frac{TP}{(TP + FP)}$

Recall: $\frac{TP}{(TP + FN)}$
Feature Matching | Brute Force

For every feature $i$ in image 1

- Go over all features in image 2
- Take the one (k) that are closest to $i$

Complexity: $O(n^2)$

Good:

- Simple
- Easily parallelizable
- Exhaustive, deterministic

Bad:

- Slow
Feature Matching | k-d tree

Accelerate the matching process.

Preprocess the “training” dataset:

- Hierarchically subdivide the $k$-dimensional space (by sorting)
- If a bin has more than $m$ points - split to smaller spatial bins.

Build the tree: $O(k \cdot n \log n)$

Search the resulting tree: $O(\log n + m)$
Feature Matching | k-d tree

Accelerate the matching process.

Preprocess the "training" dataset:
- Hierarchically subdivide the k-dimensional space (by sorting)
- If a bin has more than m points - split to smaller spatial bins.

Build the tree: $O(k \cdot n \log n)$

Search the resulting tree: $O(\log n + m)$

$\%timeit -n 50 \quad \text{matches2to1} = \text{bf.knnMatch}(\text{des2}, \text{des1}, k=2)$
50 loops, best of 3: 17.1 ms per loop

$\%timeit -n 50 \quad \text{matches2to1} = \text{flann.knnMatch}(\text{des2}, \text{des1}, k=2)$
50 loops, best of 3: 10.5 ms per loop

*Speedup: ~ x1.7* with ~1000 features

$\%timeit -n 50 \quad \text{matches2to1} = \text{bf.knnMatch}(\text{des2}, \text{des1}, k=2)$
50 loops, best of 3: 148 ms per loop

$\%timeit -n 50 \quad \text{matches2to1} = \text{flann.knnMatch}(\text{des2}, \text{des1}, k=2)$
50 loops, best of 3: 33 ms per loop

*Speedup: ~ x4.4* with ~3300 features
Questions?
Image Alignment
Panoramic view, right?
Panoramic view, right?
Panoramic view, right?
Why align, stitch images together?

- The human eye has ~210°_h x 150°_v FOV
- Our images/cameras have narrow FOV
- Object is just. too. big.
- We want to deliver the “big picture”
Image Alignment | Applications

Panoramas
Google Street View
Image Alignment | Applications

360 VR
Aerial mapping, Surveillance, Drones!
Image Alignment | Applications

Image search, Search in image

[OpenCV]
Image Alignment | Applications

Video stabilization

[youtube]
Is image alignment hard?

Between any two images we may have:

- Little overlap
- Occlusion
- Clutter (confusing features)
- Intensity changes
- Distortions
- Different hardware, optics
- ...

→ Alignment should be robust to a lot of variance.
What if we were aligning real paper images?

- We can only rotate and translate.

This was done: Panography.
Feature Alignment

Instead of looking at pixels, we look at **interest points**.

- Less data to process
- More potential noise (from false matching)

Method overview:

1. Find key points and feature descriptors **(Last time)**
2. Match descriptors **(we saw today)**
3. Apply an overall geometric configuration **(Today)**
   - Affine, Homography
4. Find optimal parameters **(Today)**
   - Fitting, LSQ, robust LSQ
5. Blend
Alignment as a Fitting Problem

Over the feature matching:

- **Design a goodness of fit measure**

\[
\hat{T} = \arg\min_T f(T(X_1), X_2)
\]

\[
f(T(X_1), X_2) = \sum_i \|T(x_{1i}) - x_{2i}\|
\]

- **Design an optimization scheme**
  - Avoid local minima
  - Robust to outliers
Simple Line Fitting: Least Squares

Input: Points \( X = [(x_1, y_1), \ldots, (x_n, y_n)] \)

Output: (Best) Line through all points \( y = \hat{a}x + \hat{b} \)

\[
\begin{align*}
a, b &= \arg\min_{a,b} E(X) = \sum_{i=1}^{n} (y_i - ax_i - b)^2 \\
E(X) &= \sum_{i=0}^{n} \left\| (x_i, 1) \begin{bmatrix} a \\ b \end{bmatrix} - y_i \right\|^2 = \|Ax - y\|^2 \\
      &= y^T y - 2(Ax)^T y + (Ax)^T (Ax)
\end{align*}
\]

\[
\frac{\partial E}{\partial x} = -2A^T y + 2A^T Ax = 0 \\
A^T Ax = A^T y \implies x = (A^T A)^{-1} A^T y
\]
Least Squares

Good:
- Linear (fast, closed form)

Bad:
- Sensitive to outliers (not robust)
- Only detects a single line

→ Use a robust estimator.
Robust Least Squares
Robust Least Squares

Choose between \textit{squared} or \textit{absolute} loss:

\[
E(x_i) = \begin{cases} 
\|y_i - l(x_i)\|^2 & \text{if } \left| \frac{y_i - l(x_i)}{\sigma} \right| < \epsilon \\
|y_i - l(x_i)| & \text{if } \left| \frac{y_i - l(x_i)}{\sigma} \right| \geq \epsilon 
\end{cases}
\]
RANSAC

RANDom SAmple Consensus

So far - use all the points for matching & minimize the effect of outliers (robust estimators)

RANSAC idea: use only a subset of the points, discard outliers from the model fitting.

Method:

1. Choose a minimal set of samples
2. Estimate parameters (e.g. with LSq)
3. Count the number of inliers
4. Repeat 1-3 \( n \) times, pick best configuration
5. Refine the solution using only inliers
Questions?
Let's assume $T$ is Affine:

\[
\begin{pmatrix}
    x'_i \\
    y'_i \\
    1
\end{pmatrix}
= \begin{pmatrix}
    m_1 & m_2 & t_x \\
    m_3 & m_4 & t_y \\
    0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    x_i \\
    y_i \\
    1
\end{pmatrix}
\]

Or in Homogeneous coordinates:
2D Affine transformation:
- Parallel lines and planes remain parallel, but not angles or lengths.
- 6 DoF: 4 for rotation, scale and shear + 2 for translation
- Still linear → we can use Linear Least Squares to solve, and possibly add robustness to outliers
Affine Alignment

\[ E(X) = \sum_i^n \left\| \begin{pmatrix} x'_i \\ y'_i \end{pmatrix} - \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} t_x \\ t_y \end{pmatrix} \right\|_{L_2} \]

\[
= \left\| \begin{pmatrix} \ldots \\ x_i \\ y_i \\ 0 \\ 0 \\ x_i \\ y_i \\ 0 \\ 0 \\ \ldots \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{pmatrix} - \begin{pmatrix} \ldots \\ x'_i \\ y'_i \end{pmatrix} \right\|_{L_2}
\]

A non-homogeneous system of linear equations, with 6 DoF.
Affine Alignment | Result

Somewhat aligned, But it will simply not do.
Homographies

What can we do besides Affine? a Homography.

- It does not preserve parallel lines.
- Still linear - we can use the same exact fitting process.

We added two more elements - what do they do?

Allow us to model perspective transformation.
Homography-based Alignment

Affine? Meh.

Homography! Yay!
Questions?
Panoramas
Mosaics: Geometry

Assume: Pure rotation

Method: We project all the images on a common plane.
Projective Panorama
What happens to the overlapping points?

Assume: Pure rotation

In camera 2:
\[
\begin{pmatrix}
X_2 \\
Y_2 \\
Z_2
\end{pmatrix} = K_2^{-1} \begin{pmatrix}
x_2 \\
y_2 \\
1
\end{pmatrix}
\]

Move to camera 1:
\[
\begin{pmatrix}
X_1 \\
Y_1 \\
Z_1
\end{pmatrix} = R_1 R_2^T K_2^{-1} \begin{pmatrix}
x_2 \\
y_2 \\
1
\end{pmatrix}
\]

Undo 2,
Apply 1

Project on image 1:
\[
\begin{pmatrix}
x_1 \\
y_1 \\
1
\end{pmatrix} \approx \begin{pmatrix}
x_2 \\
y_2 \\
1
\end{pmatrix}
\]

3x3 Homography

⇒ Homography is:
(1) Undo projection 2,
(2) rotate from cam 2 to cam 1,
(3) apply projection 1
A 360° Panorama With Homographies?

EEK.

Not really.
360° Panoramas

Imagine the images are on the wall of a cylinder...

View from inside the cylinder.
A 360° Panorama | Cylindrical Warping

Image coordinates → Cylindrical coordinates

Unit cylinder

Unwrapped cylinder

$\theta$

$h$

$(X, Y, Z)$

$(\hat{x}, \hat{y}, \hat{z})$
Easier to go the inverse, from cylinder to image:

\[ \theta = (x_{cyl} - x_c) / f \]
\[ h = (y_{cyl} - y_c) / f \]
\[ \hat{x} = \sin \theta \]
\[ \hat{y} = h \]
\[ \hat{z} = \cos \theta \]
Aligning in Cylindrical Coordinates

We rotated the camera by $\theta$.

How does this affect the cylindrical image?
Aligning in Cylindrical Coordinates

We rotated the camera by $\theta$.

How does this affect the cylindrical image?

It’s a (simple) translation.

Aligning cylindrical images needs only a translational model of 2 DoF.
How to Cut?

… Use Spherical Coordinates
Questions?
HW2: Image Stitching

Your goal is to create 2 panoramas:

1. Using homographies and perspective warping on a common plane (3 images).
2. Using cylindrical warping (many images).

In both options you should:

- Extract features and descriptors
- Match features
- Calculate the best model parameters
- Bonus (!!): Use your Laplacian Blending code to stitch the images together nicely

You will be given sample code to

- convert an image to cylindrical coordinates
- calculate affine or homography transform
- skeleton panorama maker

The rest is up to you...