**CMSC 491/691: Computer Vision** **Spring 2024**

# Lecture 6: 2024-02-14 Features II

*Lecturer: Tejas Gokhale* *Scribe: Jedan Davis*



## 6.1 Recap of Last Lecture

In the last lecture, we talked about features, what they are, how to detect them, and how to know if they are good. We also discussed Harris's Corner Detector and how we use Taylor Approximation to reduce computational costs used to generate corners.

High error = Good feature

We also covered Eigen thresholding to ensure we are using the best corners.

Multi-Scale 2D Blob Detector

We discussed applying different laplacian filters and deriving cross-scale maximums in order to find the best features while scaling.

## 6.2 Main Components of Features

1. Detection: This step involves pinpointing unique locations in an image known as interest points. We can use various algorithms like Harris corner detection or SIFT to achieve this.
2. Description: Once interest points are detected, feature descriptors are extracted to encapsulate the local appearance or texture around each point. These descriptors, often represented as vectors, encode crucial information about the image patch surrounding the interest point.
3. Matching: With feature descriptors extracted from different images, the goal is to establish correspondences between them.

## 6.3 Designing a Feature Descriptor

### 6.3.1 Challenges

- **Photometric Transformations**: like changes in lighting conditions or color variations can significantly alter the appearance of an image, making it challenging to design feature descriptors robust to such variations.

- **Geometric Transformations**: such as rotations, scaling, and perspective changes can distort the spatial arrangement of features, necessitating the design of descriptors invariant or robust to such transformations.

### 6.3.2 Requirements

- **Invariance**: The descriptor shouldn't change even if the image is transformed

- **Discriminability**: The descriptor should be highly unique for each point

## 6.4 SIFT (feature detector and descriptor)

### 6.4.1 Calculating SIFT Descriptor:

1. **Extract Window:** Around the detected feature, extract a 16x16 square window from the image.
2. **Compute Gradients:** Calculate the gradients within the window using appropriate methods
3. **Calculate Edge Magnitudes and Orientations:**
   - Compute edge magnitudes
   - Calculate edge orientations in degrees:
4. **Filter Edges:** Apply a gradient magnitude threshold (e.g., 0.05 * max(mag)) to remove weak edges: `mag[mag < threshold] = 0`.
5. **Create Edge Orientation Histogram:**
   - Construct a `histogram` of the surviving edge orientations
6. **Circular Shift:**
   - Find the index of the maximum value in the `histogram`
   - Circularly shift `histogram` to the right by `-max_index` positions to place the peak at the beginning.

   **Example Histogram Shifts:**

   - Original histogram: [100, 50, 20, 0, 0, 0, 0, 0]
   - Shifted histogram (to the right): [0, 0, 0, 0, 0, 50, 20, 100]

## 6.5 Feature matching

Given a feature in I 1, how to find the best match in I 2?

- Define distance function that compares two descriptors To define the difference between two features $f1$ and f2, you can use two different approaches:

1. Better approach (ratio distance):

   Calculate the distance between $f1$ and $f2$ as $||f1 - f2||$.

   Calculate the distance between $f1$ and its second-best match $f2'$ as $||f1 - f2'||$.

Define the ratio distance as the ratio of these distances:

$$ratio\,distance = \frac{||f1 - f2'||}{||f1 - f2||}$$

This approach is effective because it compares the distance between the best match $f2$ and the second-best match $f2'$, providing a measure of confidence in the match in I 2. Large values indicate ambiguous matches.

2. Simple approach (L2 distance):

   Calculate the distance between $f1$ and $f2$ as $||f1 - f2||$.

   This approach is straightforward but may yield small distances for ambiguous or incorrect matches, as it only considers the distance between the two features without comparing them to other potential matches.

- Test all the features in I 2, find the one with min distance

## 6.6 Evaluating Results

### 6.6.1 Performance of Feature Matcher

**True Positive (TP):**

*In feature matching, a true positive occurs when a matching algorithm correctly identifies a pair of features as corresponding points between two images.*

**False Positive (FP):**

*Conversely, a false positive arises when the algorithm incorrectly identifies a pair of features as matches. These features do not actually correspond to the same point in reality.*

### 6.6.2 ROC Curve

- **ROC curve:** This curve is generated by computing the true positive rate (TPR) and false positive rate (FPR) at a set of threshold values swept through the full range of possible thresholds.
- **Area under the ROC curve (AUC):** This value summarizes the performance of a feature pipeline. A higher AUC indicates better performance, with an AUC of 1 representing perfect classification.

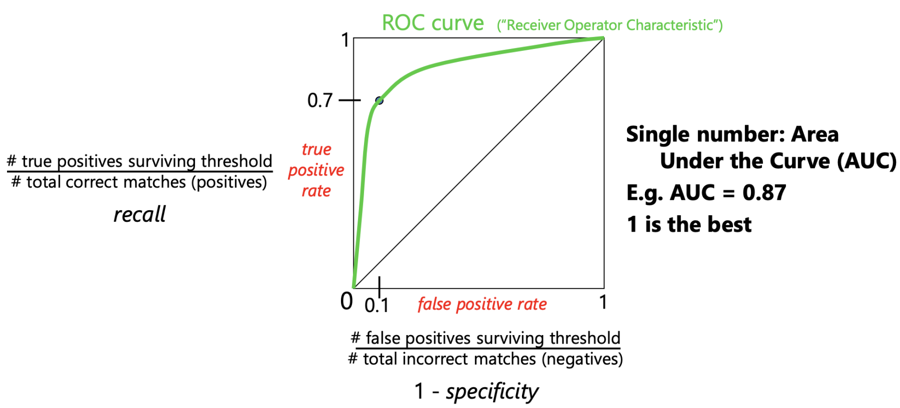


Figure 6.1: The ROC (Receiver Operating Characteristic) curve illuminates binary classifiers' performance by balancing true positive (correct "yes") and false positive (mistaken "yes") rates. A good ROC curve hugs the top-left corner, signifying strong performance in both detecting true positives and avoiding false alarms. The area under the curve (AUC) provides a single score summarizing the classifier's overall performance, facilitating model comparison and optimal operating point identification. ROC curves serve as diagnostic tools, offering insights into the efficacy of binary classifiers and aiding in understanding their strengths and weaknesses.