Project 2 Feature Detection and Matching

Overview

The goal of feature detection and matching is to identify a pairing between a point in one image and a corresponding point in another image.

In this project, you will write code to detect discriminating features (which are reasonably invariant to translation, rotation, and illumination) in an image and find the best matching features in another image.

Brief

Assigned: Wed, Oct 13, 2022
Due: **Thur, Nov 17, 2022 (by 10:00 PM) (turn in via QQ)**
Teams: The projetct must be done in groups of **2 students**. Each group must submit code assignment and one report for the assignment. Both are packed as compressed file （.rar format）and uploaded via QQ.

Description

The project has three parts: feature detection, feature description, and feature matching.

1. Feature detection

In this step, you will identify points of interest in the image using the Harris corner detection method. The steps are as follows (see the lecture slides/readings for more details). For each point in the image, consider a window of pixels around that point. Compute the Harris matrix H for (the window around) that point, defined as

$$H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

here the summation is over all pixels (x,y) in the window W. The weights $\omega_{x,y}$ should be chosen to be circularly symmetric (for rotation invariance). A common choice is to use a 3x3 or 5x5 Gaussian mask. I_x and I_y indicate the partial derivatives in x and y.
To find interest points, first compute the corner strength function

c(H) = determinant(H)/trace(H)
Alternately, you can calculate the corner response f.

$$f = Determinant(H) - \alpha\big(Trace(H)\big)^2$$

Though this is somewhat trickier because you have to select a value for k, typically between 0.04 and 0.06.
Once you have computed c or *f* for every point in the image, choose points where c or *f* is above a threshold. You also want it to be a local maximum in at least a 3x3 neighborhood.

2. Feature description

Now that you have identified points of interest, the next step is to come up with a descriptor for the feature centered at each interest point. This descriptor will be the representation you use to compare features in different images to see if they match.

You will implement two descriptors for this project.
For starters, you will implement a simple descriptor which is the pixel intensity values in the 5x5 neighborhood. This should be easy to implement and should work well when the images you are comparing are related by a translation.

Second, you will implement a simplified version of the MOPS descriptor (see the readings-simplified MOPS). You will compute an 8x8 oriented patch sub-sampled from a 40x40 pixel region around the feature. You should also normalize the patch and normalize using the mean and variance of the pixels. If the variance is very close to zero (less than 10^{-10} in magnitude) then you should just return an all-zeros vector to avoid a divide by zero error.

3. Feature matching

Now that you have detected and described your features, the next step is to write code to match them (i.e., given a feature in one image, find the best matching feature in another image).

The simplest approach is the following: compare two features and calculate a scalar *distance* between them. The best match is the feature with the smallest distance. You will implement two distance functions:

1. Sum of squared differences (SSD): This is the squared Euclidean distance between the two features.
2. The ratio test: Find the closest and second closest features by SSD distance. The ratio test distance is their ratio (i.e., SSD distance of the closest feature match divided by SSD distance of the second closest feature match).

Coding Rules

You may use NumPy, SciPy and OpenCV2 functions to implement mathematical, filtering and transformation operations. Do not use functions which implement keypoint detection or feature matching.

Here is a list of potentially useful functions (you are not required to use them):

scipy.ndimage.sobel

scipy.ndimage.gaussian_filter

scipy.ndimage.filters.convolve

scipy.ndimage.filters.maximum_filter

scipy.spatial.distance.cdist

cv2.warpAffine

np.max, np.min, np.std, np.mean, np.argmin, np.argpartition

np.degrees, np.radians, np.arctan2

What to Turn In

Code:

The code you need to write will be for your feature detection methods, your feature descriptor methods and your feature matching methods. All your required edits will be in **features.py**.

1. The function detectKeypoints in HarrisKeypointDetector is one of the main ones you will complete
2. You will need to implement two feature descriptors in SimpleFeatureDescriptor and MOPSFeaturesDescriptor classes.
3. You will implement a function for matching features. You will implement the matchFeatures function of SSDFeatureMatcher and RatioFeatureMatcher.
4. The function for the generation of ROC curves and AUC will be completed

Report:

Create a report in form of a pdf document:

- Run the features.py on the Yosemite dataset involving Simple or MOPS descriptors with SSD or Ratio distance. Include the resulting ROC curves, report AUC and comment their performance.
- Include the harris image on yosemite/yosemite1.jpg. Comment on what type of image regions are highlighted. Are there any image regions that should have been highlighted but aren't?
- Take a pair of your own images and visually show feature matching performance of MOPS + Ratio distance
- A table describing your division

  An example:

Task	People
1. Feature detection	Student A
2. Feature description	Student B
3. Feature matching	Student A
4. Evaluating and comparing algorithms	Student B
5. Writing report	Student A and B

Grading

- Code (20 Points)
- Report(10 Points)