Homework 4 Questions

Instructions

- 4 questions.
- Write code where appropriate.
- Feel free to include images or equations.
- **Please use only the space provided and keep the page breaks.** Please do not make new pages, nor remove pages. The document is a template to help grading.
- If you really need extra space, please use new pages at the end of the document and refer us to it in your answers.

Questions

Q1: Imagine we were tasked with designing a feature point which could match all of the following three pairs of images. Which real world phenomena and camera effects might cause us problems? Use the OpenCV function $cornerHarris$ to investigate.

RISHLibrary — Chase — LaddObservatory

A1: In *RISHLibrary* and LaddObservatory, if we use cornerHarris function then we could notice that the most of the corner points detected are in the repetitive texture patterns in the brick wall and the floor. These kinds of repetitive patterns make the high density of feature points in those patterns. Since these points are repetitive, it is hard to distinguish the points and correctly match the point between the images. Since the feature points should be unique for discrimination, these patterns may affect the result of feature point detecting in a bad direction.

For *Chase*, we could notice the most of the points located in the alphabet letters in the statue. This is also the same problem with the repetitive patterns, hard to match the corresponding points between images. Also the letters could be occluded or overlapped to the other elements of the image, making the feature point complex and increase ambiguity in feature point detection and recognition.

Q2: In designing our feature point, what characteristics might we wish it to have? Describe the fundamental trade-off between feature point invariance and discriminative power. How should we design for this trade-off?

A2: A feature point is a point of the image which is distinct and easily detectable, and it usually represents the corner, edge, or blobs of the image. There are several characteristics that feature points should satisfy.

First, it should be unique enough to differentiate between different pairs of an image, or different images. With this character, we could easily match the corresponding features between different images.

Second, feature points should be invariant to image transformations, such as scaling and rotation. To find feature points that satisfy invariance we use various algorithms such as SIFT.

Lastly, the feature points should have repeatability, which means that we could detect the same features in different images of the same scene even though the condition varies.

We could talk about the trade-off between the invariance and discriminative power of the feature point. If we make the feature more invariant to transformations, we could ensure that we it could be detectable under a wide range of conditions.

However, too much invariance make the feature lose its specific informations. This means that could may not distinguish points between similar but different objects or scenes; the decrease of discriminative power.

Oppositely, high discriminative power in features leads to easy seperation between simillar images. However, if these features are too specific, they would not have enough invariance thus could not be detected in images with different conditions.

To design this trade-off, we should mainly consider two things. First, we should consider what is prior for the current task with the environment of the scene. For example, if the images are taken under varying lightings, we should give priority to the invariance for the light changes. However, if the lighting is identical and we should analze the detailed texture pattern, preserving the discriminative power might be more important.

To apply this in local feature point extraction, we could vary the window size to change the scale of the features. Coarse scales with larger windows are more invariant, so we could use this in initial matching; and for localization, we could use smaller windows and finer scale.

Q3: In the Harris corner detector, what do the eigenvalues of the 'M' second moment matrix represent? Discuss both how they relate to image intensity and how we can interpret them geometrically.

A3: Matrix 'M' represents the local characteristics of the image surface. Two eigenvalues represents the intensity changes in the local region of image, the window we applied correlation.

The eigenvalues represents the magnitude of the image intensity changes in the window. Large eigenvalue mean that there is a large intensity change in the direction of corresponding eigenvector. Thus, if both eigenvalues are large, it means that there are big changes in intensity around the pixel, meaning that it is a corner. If one eigenvalue is high and other is low, it means an edge, since there is an big intensity change for one direction. It both are small, it means the flat region.

Q4: Explain the difference between the Euclidean distance and the cosine similarity metrics between descriptors. What might their geometric interpretations reveal about when each should be used? Given a distance metric, what is a good method for feature descriptor matching and why?

A4: Euclidean distance measures the straight-line distance between two vectors.We could use this distacne when the absolute difference between the feature values is important. For example, like texture matching, the exact intensity values of the pixels are crucial, we could use Euclidian distances.

In contrast, cosine similarity measures the cosine of the angle between two vectors. Thus, cosine similarity measures how much two feature descriptors are similar in theri orientations. We could use this when the magnitudes of the feature descriptors are simillar but the orientations are different in the histogram.

For feature descriptor matching, we could use k-nearest neighbors algorithm for both Euclidean distance and cosine similarity metrics. K-NN algorithm is efficient because it is very flexible among diffent metrics. We could simply change the metric calculation part to apply with different metrics, and it acts as same as before we change the metrics. This makes k-NN easily appliable across different domains. We could choose Euclidean distance when the magnitude of decscriptors are dominant, and could choose cosine similarity if the orientations are dominant in the image domain that we have interest.