EN1610 & 2605 Image Understanding

Lab05: Feature Descriptors and Correspondences

The goal of this lab is to

- Learn how to describe a feature using region descriptors.
- Learn how to find feature correspondences using feature descriptors.

Problem 1. Feature Descriptors and Matching In order to find the correspondences between features from two images, we need to describe the distinction of each feature by a descriptor, and use the similarity measure between descriptors to see which feature from one image best match which feature from another image.

The necessary steps to follow in this question are:

1. **Detect Features:** Figure 1 shows a bunch of image pairs with different scenarios including change in planar rotation, change in views, change in illumination, change in scale, repetitive textures, *etc*. Detect corners on these image pairs using your code from the previous lab. Return a list of corner coordinates as the output of your corner detector.

2. **Extract Feature Descriptors:** Write a function with four inputs: *(i)* `image` as the input image, *(ii)* `corners` as a list of corner coordinates, *(iii)* `window_sz` as the size of the window attached to each corner, and *(iv)* `descriptor_type` as the type of the region descriptor. Return a set of region descriptors `D` for all corners as the function output.

   ```
   function D = region_descriptors(image,corners,window_sz,descriptor_type)
   ```

   A string is given as the input argument for `descriptor_type` which includes:

   - `"pixels"`: image intensities of pixels in a window attached to the corners detected from the input image.
   - `"histogram"`: bin values in a window attached to the corners detected from the input image.

   For histogram based region descriptors, be sure to minus each pixel intensity in the window with the mean and then normalize it before creating the histogram. Also, be sure that the number of bins has be the equal for all corners on all images.

   Try to pick good parameters including the window size, the number of bins in the histogram, *etc* for each pair of the images in Figure 1.

3. **Match Features:** Denote the region descriptors for all corners from the first image as `D1` and from the second image as `D2`, write a function that takes `D1` and `D2` and finds for each descriptor in `D1` the best match and the second best match in `D2`. Return the indices of

the best and the second best matches in `D1` and `D2` as `indx_1st_best` and `indx_2nd_best` as well as their respective similarities as `simi_1st_best` and `simi_2nd_best`.

```
function [indx_1st_best, indx_2nd_best, simi_1st_best, simi_2nd_best]
= find_matches(D1, D2, similarity_type)
```

A string is given as the input argument for `similarity_type` which includes:

- `"SSD"`: Sum of squared distance (SSD)
- `"NCC"`: Normalized Cross Correlation (NCC)
- `"Chi-Square"`: Histogram Chi-Square Distance

Please refer to the Appendix for each similarity measure. SSD and NCC are only used for the descriptor type `"pixel"` while Chi-Square distance is used only for the descriptor type `"histogram"`. The similarities for each metric is basically the *distance* between two descriptors, and thus the smaller the distance, the closer the two descriptors. However, be aware that if you use the dot product formulation for the normalized cross correlation, the similarity becomes how two descriptors *correlate* with each other, so the larger the correlation, the closer the two descriptors.

4. **Apply Lowe's Test Ratio:** now we have the best and the second best similarities between corners, pick the match if Lowe's test ratio is satisfied when the *distance* between two descriptors is used as their similarity measure:

   $$\texttt{simi\_1st\_best} < \texttt{simi\_2nd\_best} \cdot \alpha,$$

   where α is the test ratio, a scalar within the range from 0 to 1. Try to play with this ratio to find the best matching result.

5. **Visualization:** After finding the best matches between corners from the two images, pick the top N best matches ordered by the similarity measure. N can be either 10 or 20. Use the provided function in `visualize_matches.m` for visualization. Show your results for each region descriptor type, *i.e.*, `"pixels"` and `"histogram"` as well as each similarity type, *i.e.*, `"SSD"`, `"NCC"`, and `"Chi-Square"`. An example visualization result is shown in Figure 2.

6. **How accurate are the matches?** Apply the ground truth homograph matrix provided in this lab, transform all the corners with correspondences from one image to another. Compute the Euclidean distance between the matched corner and the transformed corner; if the Euclidean distance is smaller than 1 or 2 pixels, mark the matches as a successful correspondence pair. Calculate the accuracy by dividing the number of successful matches over the total number of all matches.

7. Now use VLFeat to detect and extract SIFT key points and SIFT descriptors from the same pairs of images in Figure 1. Do SIFT feature matching, where the similarity measure

of the SIFT descriptors is simply the Euclidean distance. Make sure to use Lowe's test ratio to discard ambiguous SIFT feature correspondences.

8. Finally, discuss your results: which descriptor performs the best, and for what scenario? Where did the descriptors fail, and why do you think they fail?

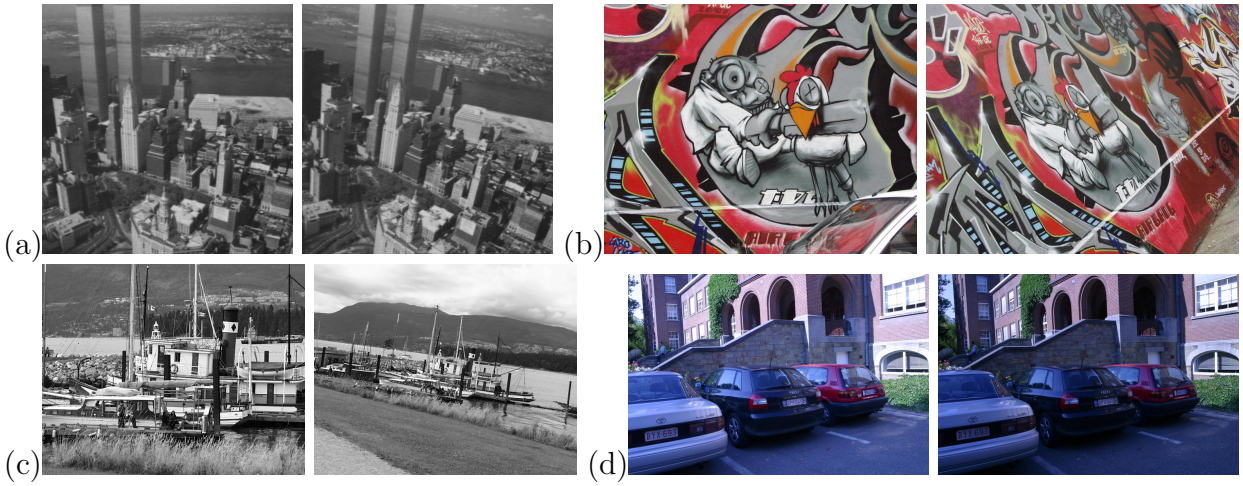

Figure 1: (a) Change in planar rotation (b) Change in views (c) Change in scale and planar rotation (d) Change in illumination

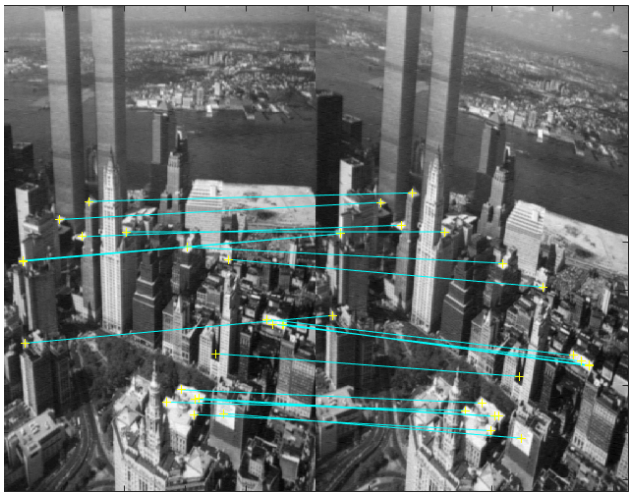

Figure 2: An example of top 20 feature correspondences.

Problem 2. Improve Feature Matching Accuracy (This part is mandatory for ENGN2605 students but is optional for ENGN1610 students who will get 15 extra points if this problem is finished correctly.)

A lot of feature correspondences are incorrect due to many factors such as noise, repetitive features, *etc*. Two common methods have been used to improve the matching accuracy:

1. **Bidirectional consistency:** when finding the feature correspondences in problem 1, the similarity of one feature in image I is compared to all features in image $\bar{I}$, and the process continues until all features in image I have been compared. This is usually called a *forward matching*. However, if we start from a feature in image $\bar{I}$ and compare its similarity with all the features in image I, *i.e.*, a *backward matching*, the resultant correspondences might not necessarily be identical. The bidirectional consistency states that a stable and accurate feature correspondence should be found identical from both the forward and backward matching.

2. **Unique correspondence:** when we compare features in image I with features in image $\bar{I}$, multiple features in I could find their best match to a single feature in $\bar{I}$, *i.e.*, a many-to-one correspondence. The uniqueness of a correspondence states that if there are multiple features in I that found their best matches to only one feature in $\bar{I}$, the correspondence with the highest similarity should be picked as the only match and the rest of the correspondences should be removed.

Implement the above two methods in problem 1. What are the accuracy of feature correspondences after employing these two methods?

Problem 3. Efficient Feature Matching (This part is mandatory for ENGN2605 students but is optional for ENGN1610 students who will get 5 extra points if this problem is finished correctly.)

Although SIFT features are very robust in terms of feature correspondences, it is known to be very computationally inefficient. Thus, other feature descriptors have been proposed to speedup the computational time while preserving the matching accuracy. One of them is Binary Robust Independent Elementary Features (BRIEF). Read the paper from `https://www.cs.ubc.ca/~lowe/525/papers/calonder_eccv10.pdf`, and briefly summarize how a BRIEF descriptor describes a feature and why it is fast in matching features.

Appendix

Let $\gamma = (x, y)$ be a feature on image I, $\bar{\gamma} = (\bar{x}, \bar{y})$ be a feature on another image $\bar{I}$, and the image intensities of the window W attaching two features be $I_w(\gamma) = \{I(\xi, \eta) | (\xi, \eta) \in W(\gamma)\}$ and $\bar{I}_w(\bar{\gamma}) = \{I(\xi, \eta) | (\xi, \eta) \in W(\bar{\gamma})\}$. Below shows some similarity measures between γ and $\bar{\gamma}$.

1. Sum of Squared Distance (SSD)

$$SSD = \sum_{i \in W} \sum_{j \in W} \left(I(x + i, y + j) - \bar{I}(\bar{x} + i, \bar{y} + j) \right)^2 \tag{1}$$

2. Normalized Cross Correlation (NCC)

$$NCC = \sum_{i \in W} \sum_{j \in W} \left(\frac{I(x + i, y + j) - \mu}{||I_w(x, y)||} - \frac{\bar{I}(\bar{x} + i, \bar{y} + j) - \bar{\mu}}{||\bar{I}_w(\bar{x}, \bar{y})||} \right)^2, \tag{2}$$

where μ and $\bar{\mu}$ are the mean of the intensities in the windows $W(\gamma)$ and $W(\bar{\gamma})$, respectively. Alternatively, normalized cross correlation can be represented by the dot product of the two windows, *i.e.*,

$$\sum_{i \in W} \sum_{j \in W} \frac{\left(I(x + i, y + j) - \mu \right) \left(\bar{I}(\bar{x} + i, \bar{y} + j) - \bar{\mu} \right)}{||I_w(x, y)|| ||\bar{I}_w(\bar{x}, \bar{y})||}. \tag{3}$$

3. Chi-Square Distance
When creating histograms g and h for the image intensities of the two windows, the distance between the two histograms is measured by the Chi-square distance:

$$\chi^2(g, h) = \frac{1}{2} \sum_{i=1}^{n} \frac{(g_i - h_i)^2}{g_i + h_i}, \tag{4}$$

where n is the number of bins.