Computer Vision

Lab Assignment - Local Features

ETH Zurich, Computer Science Department

Objective

In this lab assignment, you will implement your own feature detector and a basic matching protocol to establish pixel-wise correspondences between images. Along with the template source code, some test images are provided.
Don't modify the already implemented functions (`extractPatches.m`, `plotImageWithKeypoints.m`, `plotMatches.m`) and don't change the function interfaces.
You will need the MATLAB Image Processing Toolbox for this Lab Assignment.

1 Detection (Total: 60%)

In this section, you will implement a Harris corner detector to find interest points in an image. For this, you will start by computing the Harris response function $\mathcal{C}$ for all pixels in an image. Afterwards, a pixel (i, j) is selected as a keypoint in an image if the following two conditions are satisfied: $\mathcal{C}(i, j)$ is above a certain detection threshold T and $\mathcal{C}(i, j)$ is a local maxima in its 3×3 neighborhood. For more details, please refer to [1]. Add your code to `extractHarris.m`.

1.1 Image gradients (10%)

Compute the image gradients I_x and I_y in the x and y directions respectively. For this question, use the `conv2` function with the right convolutional filters.

$$I_x(i, j) = \frac{I(i, j+1) - I(i, j-1)}{2}, I_y(i, j) = \frac{I(i+1, j) - I(i-1, j)}{2} \tag{1}$$

1.2 Local auto-correlation matrix (10%)

Compute the elements of the local auto-correlation matrix defined by the following equation at each pixel position p:

$$M_p = \sum_{\mathrm{p'}} w_{p'-p} \begin{bmatrix} I_x(p')^2 & I_x(p')I_y(p') \\ I_y(p')I_x(p') & I_y(p')^2 \end{bmatrix} . \tag{2}$$

The local weighting w is generally chosen to be Gaussian with standard deviation σ (centered in $(0, 0)$). You can take advantage of the `imgaussfilt` function.

1.3 Harris response function (20%)

The Harris response function can be defined as $\mathcal{C}(i, j) = \det(M_{i,j}) - k \operatorname{Tr}^2(M_{i,j})$ with $k \in [0.04, 0.06]$, an empirically determined constant. Compute the Harris response function for all pixels using the closed-form formulas for the determinant and trace.

1.4 Detection criteria (20%)

Take into account the two detection conditions mentioned at the beginning of this section to obtain the final keypoints. For the local maximum check, you can use the `imregionalmax` function.
In the report, plot the detected keypoints using `plotImageWithKeypoints` for the test images (`blocks.jpg` and `house.jpg`) when varying the threshold T, the constant k and the standard deviation σ. Choose the most appropriate values to use for the rest of the assignment.
What possible issues do you notice with the detected keypoints?

2 Description & Matching (Total: 40% + Bonus 10%)

In this section, you will implement a matching protocol for image patches and test it out on the provided image pair (`I1.jpg`, `I2.jpg`). Add your code to `extractDescriptors.m` for Question 2.1 and `matchDescriptors.m` for Questions 2.2 and 2.3 respectively.

2.1 Local descriptors (10%)

Use the provided `extractPatches` function to extract 9×9 patches around the detected keypoints which will be used as descriptor. Make sure to first filter out the keypoints that are too close to the edges.

2.2 SSD one-way nearest neighbors matching (20%)

The sum-of-squared-differences (SSD) is defined as:

$$SSD(p, q) = \sum_i (p_i - q_i)^2 \ . \tag{3}$$

Use the `pdist2` function with the right distance parameter to compute the SSD between the descriptors of all features from the first image and the second image.
Implement a one-way nearest neighbors matching protocol where each feature from the first image is associated to its closest feature from the second image.
In the report, plot the obtained matches using the provided `plotMatches` function and comment the results.

2.3 Mutual nearest neighbors / Ratio test (10% + Bonus 10%)

Implement the mutual nearest neighbors matching protocol where for each one-way match, you check that it is also valid when swapping the images. Comment the results.
Implement the ratio test matching protocol where a one-way match is considered valid if the ratio between the first and the second nearest neighbor is lower than a given threshold (for instance, you can use 0.5). You can take advantage of the `mink` function. Comment the results.
If you implement both of them successfully, you will also receive the bonus points. Otherwise, you can chose which one to implement / comment.

Hand in

Hand in your commented MATLAB code and write a short report explaining the main steps of your implementation and addressing the questions from above. Make sure to include the essential parts of your code in the report!
Send the report together with your source code via the moodle submission system (do not send it over email).

References

[1] Chris Harris and Mike Stephens. "A combined corner and edge detector". In: *Proceedings of the Alvey Vision Conference*. 1988.