16-720 Assignment 2: Feature Descriptors, Homographies, RANSAC

Umut Soysal

March 6, 2019

Abstract

Contents

1 Keypoint Detector

1.1 Gaussian Pyramid

In order to create a DoG pyramid, we will first need to create a Gaussian pyramid. Gaussian pyramids are constructed by progressively applying a low pass Gaussian filter to the input image.

1.2 DoG Pyramid

The DoG pyramid is obtained by subtracting successive levels of the Gaussian pyramid.

$$D_l(x, y, \sigma) = (G(x, y, \sigma_{l-1}) - G(x, y, \sigma_l)) * I(x, y) \tag{1}$$

$$D_l(x, y, \sigma) = G(x, y, \sigma_{l-1}) * I(x, y) - G(x, y, \sigma_l) * I(x, y) \tag{2}$$

which can be written as:

$$D_l(x, y, \sigma) = GP_{l-1} - GP_l \tag{3}$$

The function createDoGPyramid in keypointDetect.py file performs this operation:

Figure 1: Gaussian Pyramid for model_chickenbroth.jpg

Figure 2: DoG pyramid for example image

1.3 Edge Suppression

computePrincipalCurvature function is prepared in the file keypointDetect.py.

1.4 Detecting Extrema

getLocalExtrema function is implemented in keypointDetect.py

1.5 Putting it together

DoGDetector function is implemented in keypointDetect.py

Figure 3: Output of the keypointDetect.py

2 BRIEF Descriptor

2.1 Creating a Set of BRIEF Tests

makeTestPattern is submitted in BRIEF.py. testPattern.npy file can be found in the submission files.

2.2 Compute the BRIEF Descriptor

computeBrief function is implemented in BRIEF.py

2.3 Putting it all Together

BriefLite function is implemented in BRIEF.py

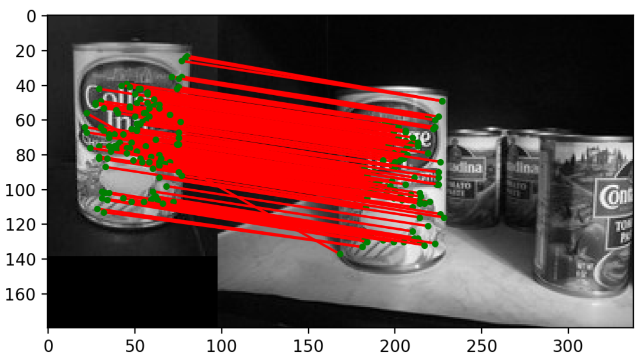

Figure 4: Matching of two photos

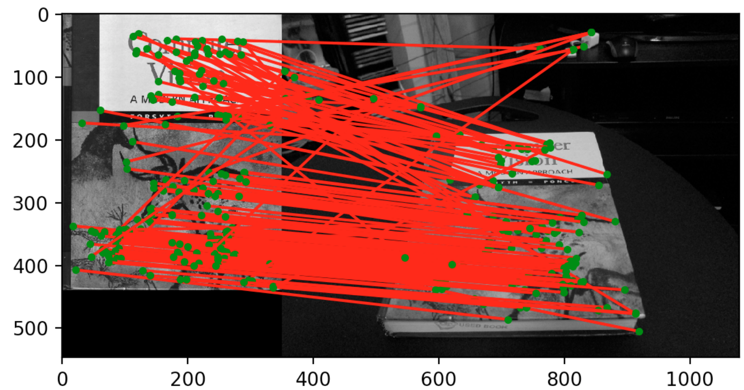

Figure 5: Matching of two book photos

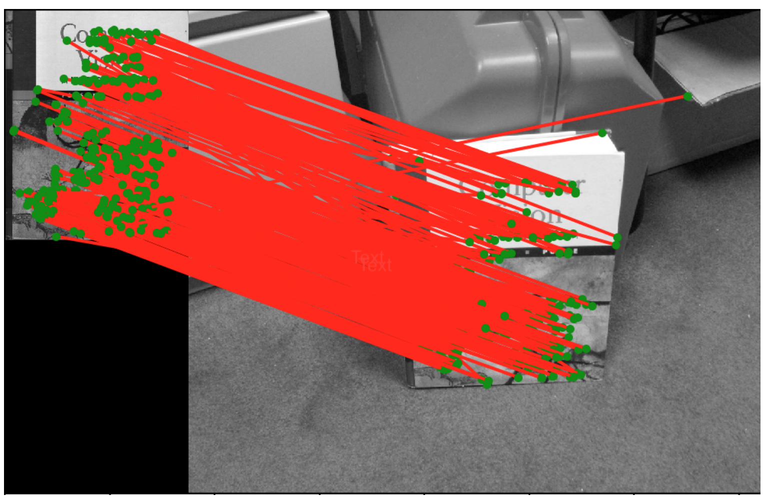

Figure 6: Another example of matching

2.4 Check Point: Descriptor Matching

2.5 BRIEF and rotations

3 Planar Homography: Theory

We have a set of N 2D homogenoeus coordinates $\{x_1, .., x_N\}$ taken at one camera view and $\{u_1, .., u_N\}$ taken at another. Suppose we know there exist an unknown homography $\mathbf{H}$ such that

$$\lambda_n x_n = H u_n \tag{4}$$

where n=1:N and λ is arbitrary scale factor

$$\lambda_n \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \tag{5}$$

$$x_1(h_{31}u_1 + h_{32}v_1 + h_{33}) = h_{11}u_1 + h_{12}v_1 + h_{13} \tag{6}$$

$$y_1(h_{31}u_1 + h_{32}v_1 + h_{33}) = h_{21}u_1 + h_{22}v_1 + h_{23} \tag{7}$$

$$-x_1(h_{31}u_1 + h_{32}v_1 + h_{33}) + h_{11}u_1 + h_{12}v_1 + h_{13} = 0 \tag{8}$$

$$-y_1(h_{31}u_1 + h_{32}v_1 + h_{33}) + h_{21}u_1 + h_{22}v_1 + h_{23} = 0 \tag{9}$$

can be expressed as:

$$\begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -x_1v_1 & -x_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -y_1u_1 & -y_1v_1 & -y_1 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \tag{10}$$

This equation can also be exressed in short form as:

$$Ah = 0 \tag{11}$$

There are 9 elements in H matrix however the degree of freedom of the matrix is 8. They can all be scaled up with respect to one element. Therefore 4 correspondences will be enough to solve the system (each correspondence gives two equations with x and y)

We can solve this system with SVD technique.

$$A = U\Sigma V^T \tag{12}$$

4 Planar Homographies: Implementation

function is implemented in planarH.py

5 RANSAC

Ransac algorithm is implemented in planarH.py

6 Stitching it together: Panoramas

Figure 7: q6_3.jpg