046746 – Computer Vision



Gal Oren 206232506   
Ohad Yaari 305337545

November 20, 2021

Homework Assignment: 1

# Features Descriptors

Part 1 – Keypoint Detector

At the first part of the assignment we will create several functions that by combining them we will be able to detect interesting points in the image. Those points, that describe image's features, will later be compared to the same object in the image, but from different point of view.

## Part 1.2 - Gaussian Pyramid

In a Gaussian pyramid, Each pixel containing a local average corresponds to a neighborhood pixels in it's surrounding ,the neighbors of pixel at each scale determined by the size of the Gaussian kernel and it's standard deviation so that at lower levels of the pyramid more neighbors are taken into account .  
 The purpose of gaussian blur is to remove noises that can disturb the detection by aliasing effect. Smoothing the image will reduce the maximum frequency of the features.

* What is the shape of Gaussian Pyramid matrix?

Answer:

The shape is (6, pixel width, pixel length), for the selected image (6,139,98).



*Figure 1: Gaussian Pyramid*

## Part 1.3 - DoG Pyramid

A Laplacian filter can be approximated by subtraction of two gaussian filter images with different kernel and standard devotion size . The Laplacian filter preform well at emphasize corners and blob-like objects.  
The purpose in the creation of the Laplacian pyramid is to enable the identification of such objects on a different scale



*Figure 2: Laplacian pyramid*

The algorithm for this process is:

𝐷𝑙(𝑥, 𝑦, 𝜎𝑙) = 𝐺(𝑥, 𝑦, 𝜎𝑙−1) ∙ 𝐼(𝑥, 𝑦) − 𝐺(𝑥, 𝑦, 𝜎𝑙) ∙ 𝐼(𝑥, 𝑦) = 𝐺𝑃𝑙−1 − 𝐺𝑃𝑙

## Part 1.4 – Edge Suppression

The main goal is to detect objects in a given image. one of the basic methods to do so is by recognized feature in the image like corners and edges. However, edges are not desirable for feature extraction as they are not as distinctive and do not provide a substantially stable localization for key-points like corners.

to distinct between the two we would like to use the principle curvature ratio in a local neighborhood of a points. One of the methods to do so is by constructing the Hessian matrix for each individual pixel in the image, when the relation between the principle curvature and the Hessian matrix is:

𝐷𝑥𝑥 𝐷𝑥𝑦

𝑇𝑅(𝐻)2

(𝜆𝑚𝑖𝑛 + 𝜆𝑚𝑎𝑥)2

(1) 𝐻 = [𝐷𝑦𝑥 𝐷𝑦𝑦] ; (2) 𝑅 = 𝐷𝑒𝑡(𝐻) =

𝜆𝑚𝑖𝑛

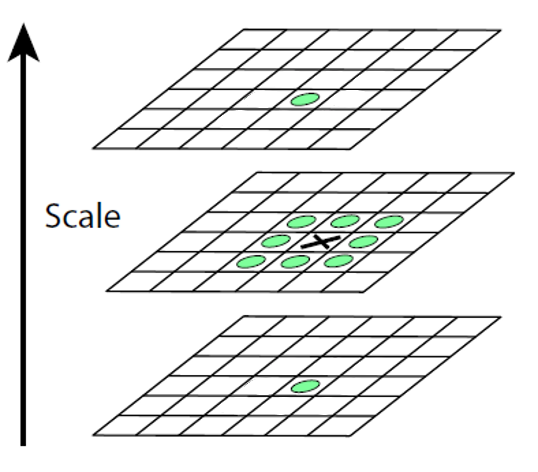
𝜆𝑚𝑎𝑥

The principal curvature reaching is minima when the two eigen values are equal, meaning the curvature is the same in two (perpendicular) directions. just to give an intuition, by looking at an edge, we can see that there is a major difference in one direction and a minor in the other. in order to keep

only the principle curvature related to corner we simply check against a threshold 𝑅 < 𝜃𝑟, In addition, in the unlikely event of a negative determinant we also discard points for which R<0.

## Part 1.5 – Detecting Extrema

To detect corner like ,scale invariant interet points, we would like to choose the highest intensity value pixel both in scale and space. That’s mean we will consider a point's eight neighbors in the space and choose the Biggest intensity pixel out of them and also compare to above and below pixel in scale as showed in Figure 3.



*Figure 3:Extrema points*

By setting threshold 𝜃𝑐 = 0.03 we should remove any point that is local extrema but doesn’t have

Gaussian response magnitude above the threshold. the second threshold use to remove edge like points by discard each point which holds principle curvature ratio bigger then 𝜃𝑟 = 12.

In our implementation of the function we first filter all the points that meet the two thresholds(against contrast and principle curvature),We then check each point along with its ten neighbors (8 in space and 2 more on a scale), and keeping it only if it is the maximum point in it’s surroundings.

We can see the results in Figure 4,5,6.  
in Figure 4 we check the algorithm against simple geometrical shapes, we can see we get a perfect detection of all corners.

In Figure 5,6 we check the algorithm against real image.  
in Figure 5 we can see that the algorithms was able to identify all the coroners and edges of the books in the figure as well as the names that appear on the books.

in Figure 6 which shows a living room ,we can see that the algorithm was able to identify the coroners of the pictures hanging on the wall and across the table, the coroners of the TV, the edges of the fruit bowl and even the reflection of the sides of the chair through the TV.

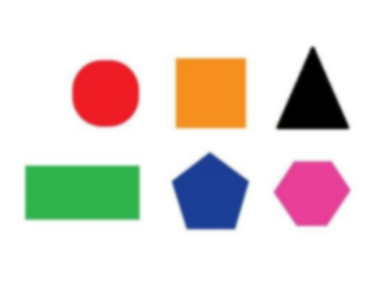
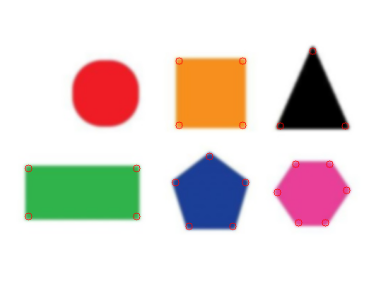
****

Figure 4: sainty check image.





'

Figure 5: real image-1



Figure 6: real image-2

How can we improve the results?

Answer:

* Optimize the value of 𝜃𝑟 and the value of 𝜃𝑐𝑜𝑛𝑡𝑟𝑎𝑠𝑡.
* smoother the image with more filter before applying detection .

# Part 2 – BRIEF Descriptor

in this section we would like to use the brief Descriptor , after extracting all the key points that holds most of the information feature points in the image we would like to descript each key point region with a descriptor. The selected descriptor called "BRIEF", is a vector that a n-bits long where each bit is the result of the following simple test:

𝜌(𝑃; 𝑥, 𝑦) ≔ {1, 𝑖𝑓 𝑝(𝑥) < 𝑝(𝑦)

0, 𝑜𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒.

The descriptor gets as input a flatten patch which (width\*width) = 92, and compare the following pairs x,y which created randomly (discussed later).

The out put is n-bits × 1 of one-hot array.

This way we create a "DNA" like array of the area around the key-point for each key-point found before.

Later we use those arrays to determine if two key -points in deferent images (containing the same object) are the same point.

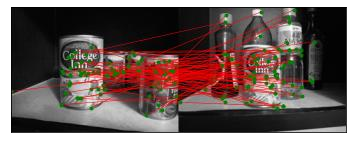
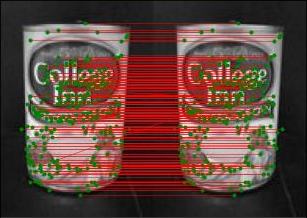
At last we present the two images with the matching key-points.

## Part 2.1 – Creating a set of BRIEF test

In the function we made, the implementation of this algorithm was to chose 256 pair with the index number of each between 0-80, randomly.

* We generated only one set of pairs for all the comparison we made.

## Part 2.2 – compute the BRIEF descriptor



At this part we picked up all the pixels (81) surrounding each key-point. The pixels were collected as a n-bits(256) long vector.

Later we applied the simple test, described before, to create the descriptor.

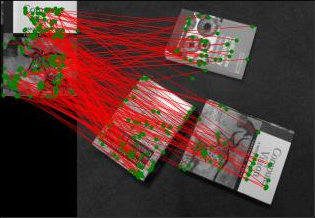
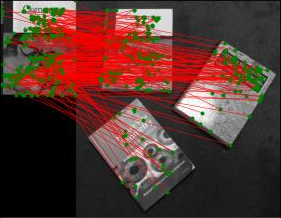
## Part 2.4 – Check point: Descriptor matching

Later we implemented the matching between two descriptor vectors to find similar features in two images with the same objects but in a different orientation, or different point of view.

* First we apply the matching function on the same images to get the following results (as expected)
* Later, we took the same object (chicken broth), and matched it from 2 different POV:

*Figure 6: chicken broth comparison*

* Later, we took the same object (CV\_book), and matched it from 3 different POV:

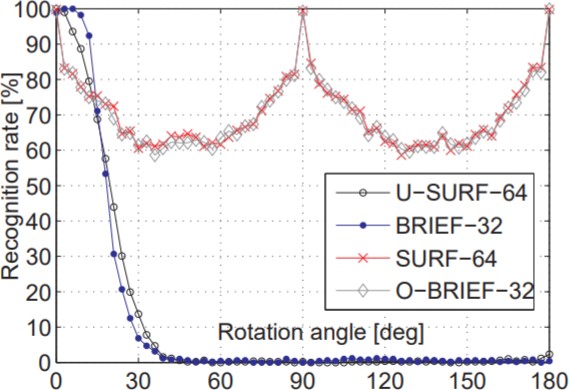


*Figure 7: no rotation (left img) with rotation(right img)*

As we can see from the above comparison, there is a poor result with the right image (with rotation).

This result is supported by [BRIEF: Binary Robust Independent Elementary Features⋆]

fig 8 presents the recognition rate as varies rotation angle. We can see a big drop with rotation, bigger than 30°.



*Figure 8: recognition rate as fnction of rotation angle*

