

Second report delivery

Image recognition of touristic places

Ximena Cervantes Díaz, 276978

Fundamentals of Artificial Intelligence, 281301
October 28, 2019

Project advisor: Dr. José Ignacio Nuñez Varela

Abstract:

For image recognition, using the different algorithms can change the time of processing and the efficiency of machine learning. Nowadays we need to use software that help us to do things automatically, with artificial intelligence. Once we have the best algorithm to image recognition, is time to check how it works and which are the paramethers of the different functions or methods. These paramethers or variables can be changed by the programmer (maybe in a future, the computers can change this automatically), to has better results of the program.

The computer has to be trainned to use these results as a “scale” or measure to recognize the different elements or a whole image. This training stage uses neural networks to save the information previously presented and compare the new information for determinate a result.

Lets see how the algorithms work, adjust the paramethers to be more efficient and choosing the best neural networks.

SIFT

According the definition of SIFT algorithm at the OpenCV documentation, in 2004, D.Lowe of the University of British Columbia made a new algorithm: Scale Invariant Feature Transform (SIFT) in Distinctive Image Features from Scale-Invariant Keypoints which extract keypoints and compute its descriptions.

SIFT has four main steps:

1. Scale-Space Extrema Detection
2. Keypoint Localization
3. Orientation Assignment
4. Local Descriptor Creation

First, the algorithm found the coordinates of keypoints in different scale. For each point, the algorithm assigns an orientation. These results are locally invariable on the image, scale and rotation. Then, it has to be necessary to make a keypoint descriptor.

Collections of keys that agree on a possible model are identified, when 3 or more keys agree is evident in the image with high probability.

Scale-Space Extrema Detection

This step takes to use Laplacian of Gaussian (LoG) method to find potential interest points that are invariant to scale and orientation.

LoG small definition

The first stage of keypoint detection is to identify locations that can be repeatably assigned under differing views of the same object.

It has been shown by Koenderink (1984) and Lindeberg (1994) that under a variety of Scale reasonable assumptions the only possible scale-space kernel is the Gaussian function.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Here is where we can take an image and blur it a little. The second step is to calculate the laplacian derivatives and this is going to locate edges and corners on the image. That edges and corners are used for finding keypoints. At this part, the second order derivatives (laplacian) is extremely sensitive to noise, that why the blur smoothes it out the noise and stabilizes this derivatives.

Keypoint Localization

At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability

Orientation assignment

One or more orientations are assigned to each keypoint location based on local image gradient directions.

Local descriptor creation

The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

SURF

The dimension of feature description vector in SIFT algorithm is too high and it will be a trouble to computation in the practice application. That's why Herbert Bay proposed an algorithm which greatly enhanced the accuracy of feature detection and reduce the matching time, this algorithm is the Speed Up Robust Features (SURF).

SURF algorithm uses box filters and a fast Hessian operator to approximate the Laplace detector, but this requires integral images which demand extra storage and computation.

The proposed SURF descriptor is based on SIFT algorithm on similar properties but with a lighter complexity. First it is necessary to fix a reproducible orientation based on information from a circular region around the keypoint or the interest point; computation of a descriptor that has

information is represented by this circle, which contains a keypoint and this will has information of the neighbourhood of keypoint look like in the right scale.

Hessian Matrix

The Hessian matrix of a function is a matrix that contains the second-order derivatives of the function. Depends on two variables such that the notation must include which variables the Hessian is calculated with respect to.

The SURF algorithm pushes the approximation of an integral images and box type filters which are used for convolution decreasing the computational time required. Is used here at the SURF algorithm for the selection of the characteristic scale and because its determinand is a measure for detecting structures.

The Hessian matrix at scale σ is defined as:

$$H(x, y, \sigma) = \begin{pmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{pmatrix}$$

Where L_{xx} is the convolution of the Gaussian second order derivative with the image I at location (x, y)

Comparision of SIFT and SURF algorithms

SIFT test algorithm

```
# Artificial Intelligence SIFT algorithm
# Original author: docs.opencv.org
# Used for the proyect of Image Recognition by Ximena Cervates Díaz

# For execution time testing, is necessary to have a measure of time, this library
gave us the necessary method
import time

start_time = time.time()

# Implement the library of OpenCV
import cv2
# Using pandas and numpy libraries
import pandas as pd
import numpy as np
```

COMPUTER SCIENCE DEPARTMENT - COLLEGE OF ENGINEERING -UASLP

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE - **2ND REPORT**

```
# Import the matplotlib library
import matplotlib.pyplot as plt
from scipy.misc.pilutil import imread, imresize

pd.set_option('display.max_rows', 5)
# Change the name of the image file and the extension
image_file = 'uaslp1.jpeg'

img = cv2.imread(image_file)
plt.imshow(img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray)
# Create a SIFT Object
sift = cv2.xfeatures2d.SIFT_create(50)

# Get the Key Points from the 'gray' image, this returns a numpy array
kp, des = sift.detectAndCompute(gray, None)
print( len(kp) )

# Now we drawn the gray image and overlay the Key Points (kp)
# The keypoints marks are circles which are of red color
img2 = cv2.drawKeypoints(gray, kp, None, (255,0,0),4)

#The program prints at the console the number of seconds that has passed since we
run and compile the script
print("---- %s seconds ----" % (time.time() - start_time))

# Plot it to the screen, looks a little small
plt.imshow(img2),plt.show()
```

SURF test algorithm

```
# Artificial Intelligence SURF algorithm
# Original author: docs.opencv.org
# Used for the project of Image Recognition by Ximena Cervates Díaz

# For execution time testing, is necessary to have a measure of time, this library
gave us the necessary method
import time

start_time = time.time()

# Implement the library of OpenCV and matplotlib
import cv2 as cv
import matplotlib.pyplot as plt

#Here the program reads the image file and put it in a variable
# Change the name of the image file and the extension
```

COMPUTER SCIENCE DEPARTMENT - COLLEGE OF ENGINEERING -UASLP

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE - **2ND REPORT**

```
img = cv.imread('uaslp1.jpeg',0)
# Create SURF object
# Is important to set the Hessian in values between 30 000 and 1 000

surf = cv.xfeatures2d.SURF_create(20000)
# Find keypoints and descriptors directly
kp, des = surf.detectAndCompute(img,None)

len(kp)

# Check present Hessian threshold
print( surf.getHessianThreshold() )

# Again compute keypoints and check its number.
kp, des = surf.detectAndCompute(img,None)
print( len(kp) )

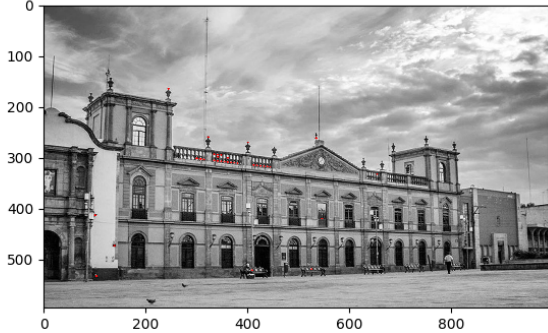
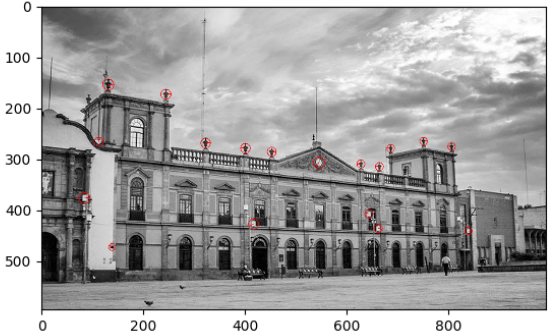
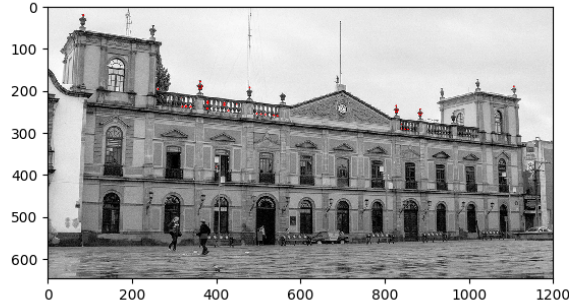
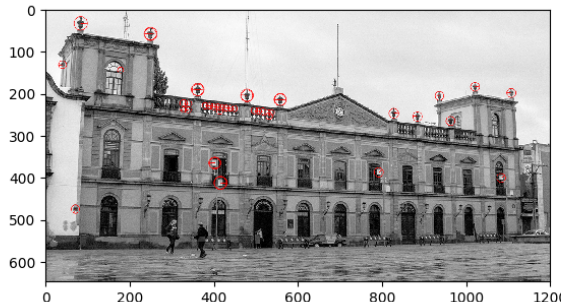
# The keypoints marks are circles which are of red color
img2 = cv.drawKeypoints(img,kp,None,(255,0,0),4)

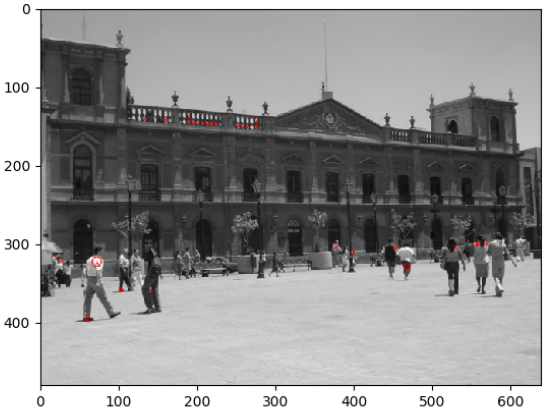
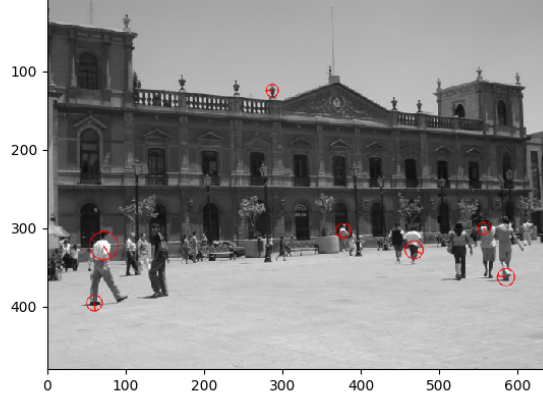
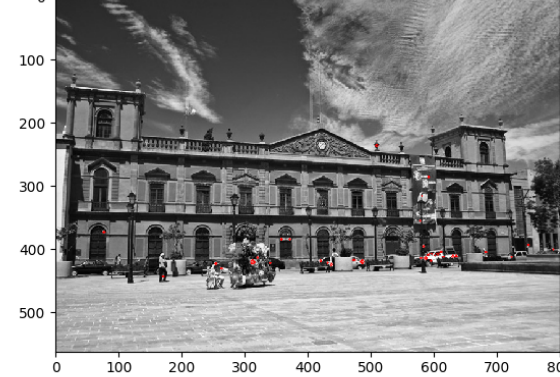
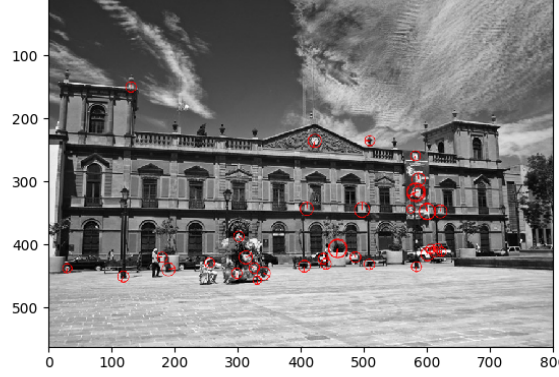


# The program prints at the console the number of seconds that has passed since we
run and compile the script
print("--- %s seconds ---" % (time.time() - start_time))

plt.imshow(img2),plt.show()
```

Applying both algorithms comparission

With a number of 50 keypoints as a limit at both algorithms (SIFT doesn't change the number of keypoints), with a value of 20 000 of Hessian Threshold in case of SURF algorithm. The number of the decimals at the time part was considered by 4 numbers after the dot. The image used bellow is the *Central Building of the Autonomus University of San Luis Potosi*.

	SIFT algorithm	SURF algorithm
1	 <p>Execution time: 3.6020 seconds Number of keypoints: 50 Percentage of useful keypoints: 90%</p>	 <p>Execution time: 1.7395 seconds Number of keypoints: 18 Percentage of useful keypoints: 83.33%</p>
2	 <p>Execution time: 3.9979 seconds Number of keypoints: 50 Percentage of useful keypoints: 98%</p>	 <p>Execution time: 1.8863 seconds Number of keypoints: 43 Percentage of useful keypoints: 97%</p>

3	 <p>Execution time: 4.6889 seconds Number of keypoints: 50 Percentage of useful keypoints: 74%</p>	 <p>Execution time: 1.5580 seconds Number of keypoints: 7 Percentage of useful keypoints: 14%</p>
4	 <p>Execution time: 3.3915 seconds Number of keypoints: 50 Percentage of useful keypoints: 10%</p>	 <p>Execution time: 1.6462 seconds Number of keypoints: 38 Percentage of useful keypoints: 23.33%</p>
5	 <p>Execution time: 3.5550 seconds Number of keypoints: 50 Percentage of useful keypoints: 80%</p>	 <p>Execution time: 1.7199 seconds Number of keypoints: 12 Percentage of useful keypoints: 83.33%</p>

As we can appreciate, both algorithms take different keypoints. SIFT algorithm in this case take the a lower ratio of the keypoint and is slowly compared to SURF algorithm, which is more precise than SIFT results. This is a simple comparission, obviously the values that influenced the analysis are the execution time, the ratio of keypoints and the number or percentage of the keypoints which are pointing at the object of interest (in this example is the central building of the university) but this percentage not means that these are the ideal keypoints, it is necessary to discard these points that are in similar coordinates (spliced points).

For SIFT algorithm, the average percentage of keypoints located at the object of interest is 70.4%, for SURF is 60.2%. SIFT algorithm result “right” than SURF algorithm in this 5 cases but we have to remember that this is a small amount of test items which is not ideal to make professional tests or inquiries.

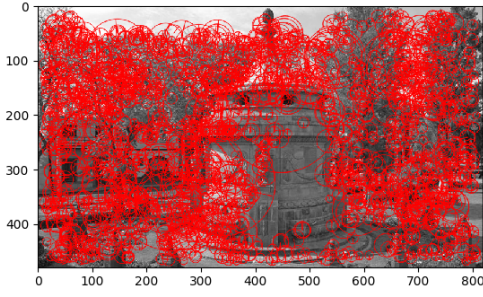
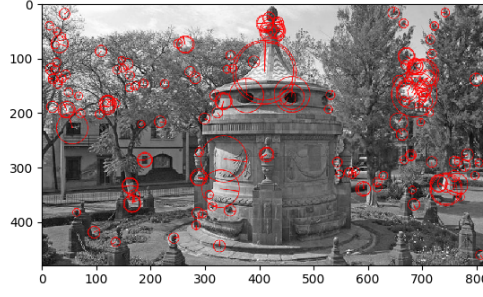
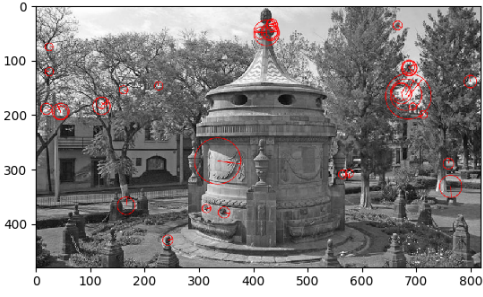
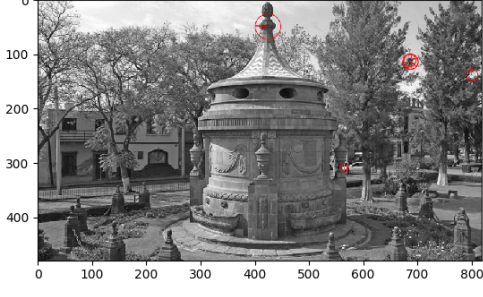
Once that the SURF algorithm has been choosen for be used, is time to pick a value to define the Hessian Threshold and use the result of the ideal keypoints for the neural network.

SURF testing

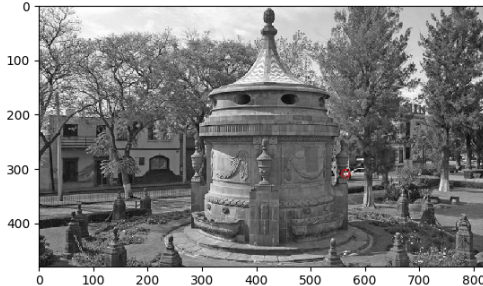
According to the aftermentioned SURF algorithm, let’s make some test to know the number of keypoints and execution time to determinate the ideal Hessian Threshold.

For the same image of the *Caja del Agua* monument, the result of the test adjusting the Hessian Threshold with $\Delta = 6\ 000$ starting at 1 000 value. The decimal at the seconds part are taken by 4 digits.

COMPUTER SCIENCE DEPARTMENT – COLLEGE OF ENGINEERING -UASLP
FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE – **2ND REPORT**

Number	Result image	Hessian Threshold	Keypoints	Time in seconds
1		1 000	2138	8.0491
2		6 000	164	1.8276
3		11 000	36	1.6285
4		16 000	6	1.6416

COMPUTER SCIENCE DEPARTMENT – COLLEGE OF ENGINEERING -UASLP
FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE – **2ND REPORT**

5		21 000	1	1.6250
---	---	--------	---	--------

As we can see, this is not always for the same Hessian Threshold, there are many different images that needs to be processed... It's necessary some heuristic to have some control over the values. The ideal rank is 20-50 keypoints at the image. So let's change the SURF test algorithm to implement this heuristic.

```
# Artificial Intelligence SURF algorithm
# Original author: docs.opencv.org
# Used for the proyect of Image Recognition by Ximena Cervates Díaz

# For execution time testing, is necessary to have a measure of time, this library
gave us the necessary method
import time

start_time = time.time()

# Implement the library of OpenCV and matplotlib
import cv2 as cv
import matplotlib.pyplot as plt

#Here the program reads the image file and put it in a variable
img = cv.imread('cajal.jpg',0)

# Variable of Hessian Threshold
hess = 20000

# Create SURF object
# Is important to set the Hessian in values between 30 000 and 1 000

surf = cv.xfeatures2d.SURF_create(hess)
# Find keypoints and descriptors directly
kp, des = surf.detectAndCompute(img,None)

len(kp)
print( len(kp) )
# Check present Hessian threshold
```

COMPUTER SCIENCE DEPARTMENT – COLLEGE OF ENGINEERING -UASLP

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE – **2ND REPORT**

```
print( surf.getHessianThreshold() )

# Check if the keypoints value are between 20 and 50, if they are not, this loop
will increase or decreases the value with 3000 of delta
if len(kp) > 50:
    while(len(kp) > 50):
        hess += 3000
        surf.setHessianThreshold(hess)
        kp, des = surf.detectAndCompute(img,None)

elif len(kp) < 20:
    while(len(kp) < 20):
        hess -= 3000
        surf.setHessianThreshold(hess)
        kp, des = surf.detectAndCompute(img,None)

print( surf.getHessianThreshold() )
# Again compute keypoints and check its number.
print( len(kp) )

img2 = cv.drawKeypoints(img,kp,None,(255,0,0),4)

print("---- %s seconds ----" % (time.time() - start_time))

plt.imshow(img2),plt.show()
```

A SURF keypoint detection is based on a Hessian matrix determinant, which retrieves points in the image with sufficient contrast that makes them easy to localise and track. Since the aforementioned threshold is set prior to the keypoint detection process, it is hard to predict how many keypoints are going to be detected in the processed image. Setting the threshold too high and acquiring a small number of relevant features negatively impacts navigation accuracy. On the other hand, too low threshold produces an excessive number of features which do not contribute to the quality of navigation but hamper the ability of the system to match them to the map in real time. The saliency threshold needs to be continuously adapted to the images that are being processed, this is why the SURF test algorithm uses the while loop, to change the values until they are similar to the rank 20-50 values.

To select the best keypoints is not a lot of documentation on books or documents relatives to this topic.

The first restriction is ratio test. Only strong matches will be included as good ones, using the distance between the first matching candidate and the second one as a quality indication. During the matching process, each keypoint from input image has a set of keypoint candidates from the tentative frame of the map.

Each of these candidate keypoints has a descriptor so the distance between both descriptors may be computed to rank all the candidate keypoints.

Final test of SURF algorithm

Implementing the last mentioned heuristic, this is a table of result of those 15 images of a specific building (*Central Building of the Autonomus University of San Luis Potosí and Caja del Agua*)

References

- 1.
2. David G. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, vol. 60, no 2, 2004, p. 91–110 - “<http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf>”
3. Hjørungnes, A. (2011). *Complex-valued matrix derivatives: with applications in signal processing and communications*. Cambridge University Press.
4. Li, K., Li, J., Liu, Y., & Castiglione, A. (Eds.). (2016). *Computational Intelligence and Intelligent Systems: 7th International Symposium, ISICA 2015, Guangzhou, China, November 21-22, 2015, Revised Selected Papers* (Vol. 575). Springer.
5. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91-110.

6. Mazal, J. (Ed.). (2019). *Modelling and Simulation for Autonomous Systems: 5th International Conference, MESAS 2018, Prague, Czech Republic, October 17-19, 2018, Revised Selected Papers* (Vol. 11472). Springer.
- 7.