**IMAGE PROCESSING PROJECT 3**

**NIKHIL CHEZIAN|11027706**

**DATE OF SUBMISSION: 15-03-2023**

**PROFESSOR: milan.gnjatovic@kpu.edu.r**

## AIM

To develop object detection program based on the Histogram of gradients. This program should meet the following requirements:

1. It computes a histogram of oriented gradients for a given image. The input parameters are: grayscale image, number of chain code directions, and grid dimension.

2. It estimates the similarity between two input grayscale images by calculating the cosine similarity between their histograms of oriented gradients.

3. It reads an object represented in image A and detects its occurrences in image B.

## INTRODUCTION:

### SIFT Algorithm

Scale-Invariant Feature Transform (SIFT)—SIFT is an algorithm in computer vision to detect and describe local features in images. It is a feature that is widely used in image processing. The processes of SIFT include Difference of Gaussians (DoG) Space Generation, Keypoints Detection, and Feature Description.

The SIFT features are local and based on the appearance of the object at particular interest points and are invariant to image scale and rotation. They are also robust to changes in illumination, noise, and minor changes in viewpoint.

**Following are the major stages of computation used to generate the set of image features:**

1. **Scale-space extrema detection**: The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.

2. **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.

3. **Orientation assignment:** One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data

that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

4. **Keypoint descriptor:** 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.

**Key benefits of SIFT include**

Features are local, making them resistant to occlusion and clutter (no prior segmentation)

**Uniqueness**: Certain characteristics can be compared to a vast database of things.

**Quantity**: Even little objects can create a large number of characteristics.

**Efficiency:** performance that is nearly real-time

**SIFT** is a very efficient and easy-to-use feature extraction technique with a lot of advantages. It helps to reduce the dimensions of the feature space by removing the redundant features, which highly impact the training of the machine learning used in large scale applications worldwide.

The different elements of an image are identified by the SIFT technique, and these features can then be used to locate related or identical things in other photos. SIFT comprises four stages for computation. This is due to the fact that some SIFT computations are highly expensive. The cascading approach of SIFT reduces the expense of retrieving the keypoints. Only areas that pass a first, less expensive test receive the more expensive operations. A collection of keypoint descriptors is the SIFT algorithm's output.

**CODE EXPLANATION**

This code is an implementation of an object detection algorithm using SIFT (Scale-Invariant Feature Transform) feature matching and HOG (Histogram of Oriented Gradients) feature extraction.

This is a Python code that performs image processing tasks using OpenCV, scikit-image, and scikit-learn libraries. The code performs the following tasks:

1.Loads two images from the specified path

2.Converts both images to grayscale

3.Detects and describes keypoints in both images using SIFT feature detector and descriptor

4.Matches the descriptors using a brute-force matcher and applies a ratio test to filter out false matches

5.Finds the homography matrix between the two images using RANSAC algorithm

6.Draws a bounding box around the detected object in the first image and displays the result

7.Resizes the images to a uniform size

8.Computes the Histogram of Oriented Gradients (HOG) features for both images

9.Calculates the cosine similarity between the two HOG feature vectors

10.The HOG feature extraction method is used to represent the local gradient information of an image. The cosine similarity is a measure of similarity between two vectors, which ranges from -1 to 1. A value of 1 indicates that the two vectors are identical, while a value of -1 indicates that they are completely dissimilar. Therefore, the cosine similarity value between the two HOG feature vectors is a measure of how similar the images are in terms of their texture and shape features.

The code first imports the required libraries - skimage, numpy, cv2 (OpenCV), PIL, sklearn, and OS. Then, it changes the working directory to where the input images are stored. The code reads two input images using cv2.imread() function, converts them to grayscale using cv2.cvtColor() function, and detects and describes the keypoints in both images using the SIFT feature detector and descriptor.

Next, it initializes the feature matcher using the Brute-Force Matcher algorithm and matches the descriptors using the knnMatch () function. Then, it applies the ratio test to filter out false matches and extracts the keypoint locations from the good matches.

The code then finds the homography between the two images using cv2.findHomography() function, which is used to transform the corners of the second image to the original image using cv2.perspectiveTransform() function. Then, it draws a bounding box around the detected object in the original image and displays the image.

After that, the code resizes the images to a uniform size of (100,100) and computes the HOG feature vector for both images using hog () function. Finally, it computes the cosine similarity of the two HOG feature vectors using cosine_similarity () function and displays the result

**RESULTS:**


**HOG feature vector for image1:**

**[3.53553391e-01 3.53553391e-01 3.53553391e-01 3.53553391e-01**

**3.53553391e-01 3.53553391e-01 3.53553391e-01 3.53553391e-01**

**3.53553391e-01 3.53553391e-01 3.53553391e-01 3.53553391e-01**

**3.53553391e-01 3.53553391e-01 3.53553391e-01 3.53553391e-01**

3.65506419e-01 3.65506419e-01 3.65506419e-01 3.65506419e-01

3.65506419e-01 3.65506419e-01 3.65506419e-01 2.54627973e-01

3.66713810e-01 2.42171164e-01 3.66713810e-01 3.66713810e-01

3.66713810e-01 3.66713810e-01 3.66713810e-01 3.66713810e-01

3.53553391e-01 3.53553391e-01 3.53553391e-01 3.53553391e-01

3.53553391e-01 3.53553391e-01 3.53553391e-01 3.53553391e-01

3.61647762e-01 3.61647762e-01 3.61647762e-01 3.61647762e-01

3.61647762e-01 3.61647762e-01 3.61647762e-01 2.90648020e-01

5.70738301e-01 3.65761043e-01 2.33078535e-01 2.45490829e-01

2.23569956e-01 2.09450645e-01 7.93094831e-02 5.70738301e-01

4.05057560e-01 4.05057560e-01 2.15141224e-01 2.36106249e-01

4.05057560e-01 2.78585635e-01 4.05057560e-01 4.05057560e-01

5.23581115e-01 3.14622217e-01 3.41347763e-01 3.41169669e-01

1.51486277e-01 1.84001644e-01 2.51035368e-01 5.23581115e-01

5.82022330e-01 1.66456716e-01 2.36088361e-01 3.14421469e-01

1.71879010e-01 2.10210504e-01 2.57803705e-01 5.82022330e-01

4.82689081e-01 4.13558830e-01 1.33213439e-01 2.55192419e-01

3.70297229e-01 3.17355723e-01 2.05640249e-01 4.82689081e-01

5.24535522e-01 3.77388179e-01 2.93891124e-01 1.99309769e-01

3.95308958e-01 1.38575076e-01 7.57267010e-02 5.24535522e-01

6.28155388e-01 4.40860139e-01 6.83130873e-02 3.60449962e-02

8.67167131e-02 2.27988185e-02 4.97841457e-02 6.28155388e-01

6.80897515e-01 6.02676195e-02 8.88785152e-02 7.46973691e-02

5.39406248e-02 6.70772505e-02 2.19629063e-01 6.80897515e-01

6.68596660e-01 2.85791950e-01 1.20385126e-01 4.32628948e-02

3.47296706e-02 5.69211938e-02 5.89028384e-02 6.68596660e-01

6.97719228e-01 8.47548157e-02 2.80771600e-03 5.40313680e-02

1.00362871e-01 5.56350795e-04 7.86898295e-02 6.97719228e-01

6.80761142e-01 1.47907690e-01 7.52459484e-02 1.24494722e-01

7.63063237e-02 5.47238130e-02 1.45854752e-01 6.80761142e-01

6.13647745e-01 4.78099551e-01 2.37646427e-02 2.63012814e-02

1.28882381e-01 1.11407067e-02 1.73899143e-02 6.13647745e-01

6.43299844e-01 2.08203262e-01 6.78371724e-02 1.56666275e-01

1.56552839e-01 2.34031988e-02 2.73458074e-01 6.43299844e-01

7.05569357e-01 6.63373380e-03 3.68434772e-02 3.08192629e-02

3.60492134e-02 1.84904121e-02 1.87362214e-02 7.05569357e-01

6.59093667e-01 1.21328736e-01 4.12634889e-02 8.95373167e-02

3.05524850e-01 6.74583524e-02 9.40996397e-02 6.59093667e-01

6.80356093e-01 1.94363593e-03 3.68778980e-03 1.51442234e-02

2.71991261e-01 1.29575733e-03 1.87714200e-03 6.80356093e-01

6.89860355e-01 5.94814500e-02 3.82210430e-02 4.42036551e-02

1.48258918e-01 3.92539926e-02 1.33082471e-01 6.89860355e-01

6.55566894e-01 2.28599186e-01 1.89624478e-02 2.16983824e-01

1.80512091e-01 6.23978714e-02 6.54739878e-02 6.55566894e-01

6.31692599e-01 1.54931407e-01 1.27494805e-01 1.77408944e-01

1.57305372e-01 1.48493942e-01 2.88792123e-01 6.31692599e-01

5.09185714e-01 2.93542560e-01 2.05893944e-01 2.13461860e-01

2.61173670e-01 1.57289876e-01 4.63014589e-01 5.09185714e-01

5.84835562e-01 1.56572917e-01 1.92617016e-01 2.33365511e-01

3.55907152e-01 1.95408394e-01 1.87094985e-01 5.84835562e-01

6.31321106e-01 5.29650270e-02 1.54414100e-01 1.94538228e-01

3.37653640e-01 1.18402208e-01 1.01706082e-01 6.31321106e-01

5.88341865e-01 2.79263447e-01 1.37295825e-01 1.09766953e-01

3.42338513e-01 1.49572776e-01 2.43419441e-01 5.88341865e-01

5.21855629e-01 6.77830746e-02 1.67473412e-01 2.63000947e-01

5.21855629e-01 1.92693114e-01 2.09900173e-01 5.21855629e-01

3.92687054e-01 3.10944527e-01 2.93287022e-01 3.92687054e-01

3.92687054e-01 2.15129287e-01 3.92687054e-01 3.92687054e-01

3.80416587e-01 3.26159864e-01 3.80416587e-01 3.80416587e-01

3.80416587e-01 2.41352568e-01 3.34342314e-01 3.80416587e-01

3.63791585e-01 2.71274362e-01 3.63791585e-01 3.63791585e-01

3.63791585e-01 3.63791585e-01 3.63791585e-01 3.63791585e-01

3.72944997e-01 2.10883490e-01 3.72944997e-01 3.72944997e-01

3.72944997e-01 3.72944997e-01 3.47851012e-01 3.72944997e-01

3.99893087e-01 1.85948361e-01 3.99893087e-01 3.99893087e-01

3.99893087e-01 2.49863892e-01 3.21587994e-01 3.99893087e-01

3.68756325e-01 2.60909421e-01 3.68756325e-01 3.68756325e-01

3.68756325e-01 3.68756325e-01 3.40644845e-01 3.68756325e-01]

**HOG feature vector for image2:**

**[0.35355339 0.35355339 0.35355339 0.35355339 0.35355339 0.35355339**
**0.35355339 0.35355339 0.38415903 0.33156563 0.31028534 0.38415903**
**0.38415903 0.34718654 0.28798631 0.38415903 0.36293482 0.27919206**
**0.36293482 0.36293482 0.36293482 0.36293482 0.36293482 0.36293482**
**0.35355339 0.35355339 0.35355339 0.35355339 0.35355339 0.35355339**
**0.35355339 0.35355339 0.35902657 0.35902657 0.35902657 0.35902657**
**0.35902657 0.31256916 0.35902657 0.35902657 0.3632606 0.31258916**
**0.3324732 0.3632606 0.3632606 0.3632606 0.3632606 0.3632606**
**0.35950093 0.30872899 0.35950093 0.35950093 0.35950093 0.35950093**
**0.35950093 0.35950093 0.4828798 0.4828798 0.22760144 0.4828798**
**0.41165527 0.04349231 0.08856462 0.26359734 0.56096929 0.2203536**
**0.18239917 0.20241399 0.2312201 0.31437188 0.30909207 0.56096929**
**0.56701534 0.17012899 0.35460873 0.19819211 0.31485971 0.16496193**
**0.19148576 0.56701534 0.429019 0.13994517 0.07020087 0.429019**
**0.42879947 0.23534867 0.429019 0.429019 0.48406882 0.48406882**
**0.1162668 0.28015178 0.4329942 0.0459141 0.124246 0.48406882**
**0.50532178 0.50532178 0.23166382 0.27337779 0.33401396 0.19776673**
**0.24338806 0.38856338 0.52388127 0.52388127 0.05125448 0.20345249**
**0.36287436 0.02491576 0.01807485 0.52388127 0.59829015 0.17246975**
**0.23819093 0.22612453 0.26169881 0.2042652 0.19045794 0.59829015**
**0.60930786 0.20280954 0.18629239 0.16162638 0.28700137 0.17430285**
**0.20682589 0.60930786 0.57323399 0.07812298 0.04493216 0.06399552**
**0.03651639 0.02564015 0.57323399 0.57323399 0.61688481 0.48660371**
**0.02749441 0. 0.02454071 0. 0.02765696 0.61688481**
**0.64491491 0.64491491 0.16007545 0.08327088 0.22373541 0.14387803**
**0.15116825 0.20494162 0.47442043 0.47442043 0.10687382 0.47339465**
**0.28326218 0.09391064 0.013949 0.47442043 0.45121382 0.1295881**
**0.18278888 0.45121382 0.45121382 0.31971362 0.18221606 0.45121382**
**0.60107751 0.14373254 0.29359222 0.12326023 0.21404133 0.26947094**
**0.19218443 0.60107751 0.57638813 0.28230575 0.20256139 0.03970183**
**0.40531222 0.2200296 0.02363354 0.57638813 0.70703424 0.00199017**
**0.00755215 0. 0.01157038 0.00199017 0.00251738 0.70703424**
**0.48182939 0.0600489 0.12181026 0.48182939 0.48182939 0.21082362**

0.09204091 0.48182939 0.49024623 0.00791455 0.09615722 0.49024623

0.49024623 0.31189277 0.07639532 0.4081099 0.46867708 0.17355333

0.25906056 0.46867708 0.46867708 0.11217554 0.10747409 0.46867708

0.59949534 0.24947552 0.23618607 0.13855251 0.22431733 0.23452137

0.19665567 0.59949534 0.56402494 0.00784193 0.0469331 0.15128598

0.02191886 0.14140548 0.56402494 0.56402494 0.66324162 0.27827263

0.04454751 0.1982398 0.03784223 0.00342059 0.00763031 0.66324162

0.35355339 0.35355339 0.35355339 0.35355339 0.35355339 0.35355339

0.35355339 0.35355339 0.47630175 0.09270339 0.10871498 0.47630175

0.47630175 0.12475065 0.23784655 0.47630175 0.52293779 0.11738012

0.42423985 0.38374864 0.19343196 0.19791002 0.18832758 0.52293779

0.51212162 0.15332648 0.26915416 0.31542579 0.36022476 0.20660743

0.32797559 0.51212162 0.48054914 0.11626409 0.15207323 0.48054914

0.48054914 0.12033396 0.15863872 0.48054914 0.4830035 0.18489481

0.11258176 0.4830035 0.4830035 0.06247689 0.12675315 0.4830035 ]

Cosine similarity of the two HOG feature vectors: 0.9083657446383477

REFERENCES :


1. https://openai.com/blog/chatgpt

2. https://www.educative.io/answers/what-is-sift

3. https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

4. https://github.com/arcticpenguin/ObjectDetection

5.https://www.youtube.com/watch?v=thcB1NcorV8

6. SIFT Chunk code extracted from mjt3fin.py—J.Chowdhury


Thank You Professor, Have a nice day.