

Machine Perception Assignment One

Due Date: Week 10 - Monday 5 October 2020 at 5pm

Kwaramba, Tawana (19476700)

Faculty Of Science And Engineering: School of Civil And Mechanical Engineering

Curtin University

Bentley WA, Australia

19476700@student.curtin.edu.au

Abstract—write at this at the end of your report :)

Index Terms—Histograms, I don't know what this is tbh...

I. TASK ONE: IMAGE HISTOGRAM, HARRIS CORNERS AND SCALE-INVARIANT FEATURE TRANSFORM (SIFT) KEY POINTS

The Harris corner detection algorithm, and the SIFT algorithm can be thought of algorithms which will pick the key features of an image with the pre-dominate implementations of these algorithms detect the corners of the image [2] [3]. Corners can be thought of as regions in an image with large variation of intensities in all directions [2]. Therefore, the SIFT and the Harris algorithm, are algorithms for detecting the corners in an image the key differences is the invariance and variance to certain image transformations [3]. Harris corner detection algorithm is mainly invariant to rotation, and the SIFT algorithm is mainly invariant to scaling and is also invariant to rotations due the orientation assignment stage of the SIFT algorithm [3], which is explored in greater detail in section II subsection ??.

A. Harris corner detection

The Harris corner detection is an algorithm whereby a window function will scan through the image to locate local maximums, these local maximums can be thought as potential candidates for key features detected by the Harris corner detection algorithm [4] [2]. This following behaviour can be modelled by the following equation:

$$E(u, v) = \sum_{(x, y)} w(x, y) [I(x + u, y + v) - I(x, y)]^2 \quad (1)$$

Whereby:

- $w(x, y)$ represents the window which is scanning over each section of the image,
- $I(x + u, y + v)$ represents the intensity neighbouring pixels, and
- $I(x, y)$ represents the intensity of the current pixel.

Thereafter, the function is through taylor expansion to form the following system of equations

$$E(u, v) \approx [u \ v] M \begin{bmatrix} u \\ v \end{bmatrix} \quad (2)$$

Whereby, M is represented by the following equation

$$M = \sum_{(x, y)} w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \quad (3)$$

Thereafter, a plethora of potential corners are found in the image matrices, the Harris algorithm will then go through each window function to determine if the found corner is most likely going to be an actual corner, this process is determined by the following equation [2]:

$$R = \det(M) - k(\text{trace}(M))^2 \quad (4)$$

Where by $\det(M)$ is the multiplication of eigenvalue values of the matrices M, and $\text{trace}(M)$ is the addition of the eigenvalue values of the matrices M [2]. Therefore, if one of the eigenvalue values is significantly greater than the other eigenvalue value those found points are most likely to be an edge, and if the eigenvalue values are large and approximately the same value that point is a corner otherwise, that point is a flat region [2].

Harris corner detection is an algorithm which can be mainly thought of as a rotation invariant algorithm meaning, no matter the angles you rotate a given image, the algorithm is going to detect the same key-features [2] [4]. This is due to that an corner will always going to remain a corner no matter what orientation the image because a corner is just a high intensity pixel, and the algorithm only looks at a pixel and it's neighbouring pixels thus, no matter what orientation the image the pixels will still have the same neighbouring pixels [2]. Albeit, if you scale the image, a pixels neighbouring pixels will be transformed hence, the window function may determine those clusters of pixels as a non-corner when it's a corner, or as a corner when it's a non-corner. Therefore, the conducted experiments should show consistencies through the set of rotate images, and inconsistencies through-out the scaled images.

B. Harris Corner detection: Results - Diamond

- 1) Rotational Results:
- 2) Scaling results:

C. SIFT Results - Diamond

- 1) Rotational results:
- 2) Scaling results:

D. Harris Corner detection: Results - Dugong

- 1) Rotational Results:
- 2) Scaling results:

E. Discussion about produced results

To determine if an image is invariant to a certain transform we have to determine how many corners that image is detecting relative to the performed corners. Hence, the first experiment is to get the returned corners from the Harris corner detector, and the SIFT transform and find the difference relative to the original image for each transform. Therefore, for perfect results we would assume that the difference should be 0 for invariant feature as they should have picked up the same number of corners in the image. Although, to greater consolidate this claim, we must determine if the corners are been picked up in the same region hence, we calculate the histograms of the image in the same channel as the corners, and theoretically for invariant features the histograms must follow perfectly after each other as the detected pixels should lay in the same region. Finally, to determine how well the histograms follow each other we calculate the distance between each transformed histogram relative to the original histogram of the image.

Look at figure 31 we can see that through each rotated image, they is irregularities in the number of key points found in each transformed image. It's natural to conclude that this experiment dis-prove Harris corner detection invariance to rotation. Albeit, we have to consider the manner the image is rotated. As can be seen in figures ??, 33, 34, and 35 you can see when you rotate the image they is an increase of black space around the image, and the boarders of the image may have been picked up as a corner. In figure 32 you see the boarders of the image are detected as a corner but, in the other images the boarders are not detected as a corner. Therefore, with this observation we should expect irregularities in the data produced. Furthermore, revisiting figure 31 we can see the produced histograms follow a clear sinusoidal pattern implying that although they is irregularity between the number of corners found in each transformed image but atleast they is consistency of the number of corners found relative to the angle of rotation of the image. Therefore, we can infer that the Harris corner detection algorithm would produce the same number of corners if each rotated image had a tightly bounded box around the image thus removing the corners picked up by the boarders of the image. Therefore, en-counting for the extra corners found due to the boarders of the image we can infer that the number of corners found for each transformed image will be very similar to the original transformed image. This

same concepts will apply for the images found in the dugong images in section??.

Looking, at the scaled results of the experiment we will expect the same to hold for invariant properties for the image, and the opposite to be true for variance to a specific image transform. Therefore, we should be expecting no clear pattern in the produced histograms for the first experiment, we should see great irregularity in the produced histograms, and we should see great irregularity in the distances between the produced histograms. Therefore, as seen in figures **the key point figures in the report** they is great irregularity in the produced histograms hence implying variance to scaling transformations. The same can be said for SIFT.

En-counting only for the numbers of corners found by the Harris corner detection is not enough to prove the invariance of the algorithm. For the diamond playing card the only two visible colors are white, and red hence we can use the cards characteristic to our advantage by drawing the found corners in the green channel, the same can be said about the dugong image by drawing the found corners onto the red channel as seen in section I-B. The aim of this experiment to see if the found corners are going to be laying in the same color space as the original image hence, for the playing card we should get a spike of green pixels around the cluster of red pixels. Referring, to figure 37 we can see that the histograms overlay over each other for the mid intensities, and they is some deviation in the lower intensities. Deviation in the lower intensities of the image is expected as in some transformed images we're introducing more black pixels into the image because of the method we're using to rotate the image as seen 32, 33, 34, and 35. Therefore, as we can see the produced histograms in figure 37 support the invariance of the Harris corner detection.

Furthermore, this is further consolidated through the calculated distances between the transformed image, and the original image. As we can see in figure 31 the distances found in each transformed image is the same. Given, the introduction of greater quantity of black pixels due to the manner of rotation of the image we would expect the histograms to be a fixed distance away from each other which, is demonstrated through figure 31. Therefore, as demonstrated the Harris corner detection rotationally invariant as show through the diamond card.

II. TASK TWO: IMAGE FEATURES

an example on how to reference other sections III

A. part i: the main steps in Liner Binary Pattern

The linear binary pattern (LBP) algorithm is an algorithm which is mainly used for feature extraction in relation to image textures [5].

The main pipeline steps of the LBP algorithm can be summarized as the following:

- Convert the image into a gray-scale image
- Calculate the LBP for each pixel in the image
- analyse the produced LBP's

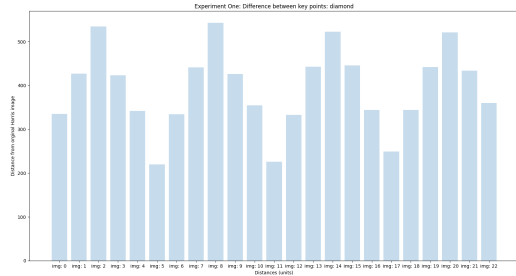


Fig. 1: Difference of keypoints found relative to first image: Harris, Rotated, Diamond

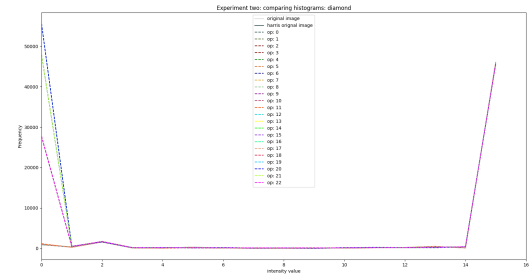


Fig. 2: Comparison of histograms for each transformed image: Harris, Roateted, Diamond

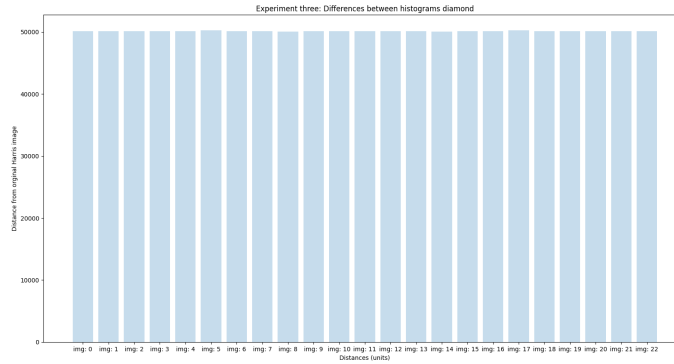


Fig. 3: Difference of distances between histograms relative to original: Harris, Rotated, Diamond

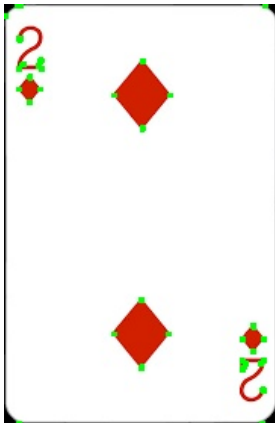


Fig. 4: Diamond harris rotated original produced image

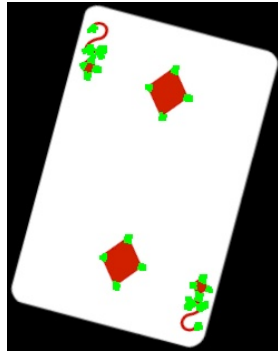


Fig. 5: Diamond harris image 1 rotated produced image

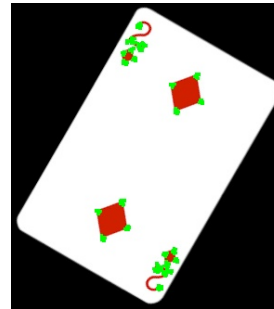


Fig. 6: Diamond harris image 2 rotated produced image

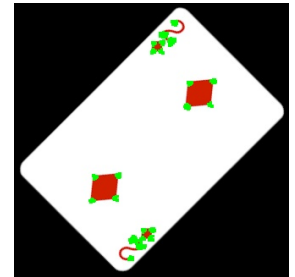


Fig. 7: Diamond harris image 3 rotated produced image

A set of 24 experimental images were produced. Only four displayed in report as displaying all 24 doesn't add great benefit to the reader. To re-view all 24 experimental images please implement the algorithms found in the appendixes.

- Calculate the histograms

Before the linear binary pattern can be performed on an image the image has to be converted into a gray-scale image [5]. The main reason for this step is because all

the color channels are unnecessary because LBP is mainly comparing the intensity of pixels relative to a neighbourhood [5], and including all the color channels will be an extra processing over-head and algorithm complexity as it contains

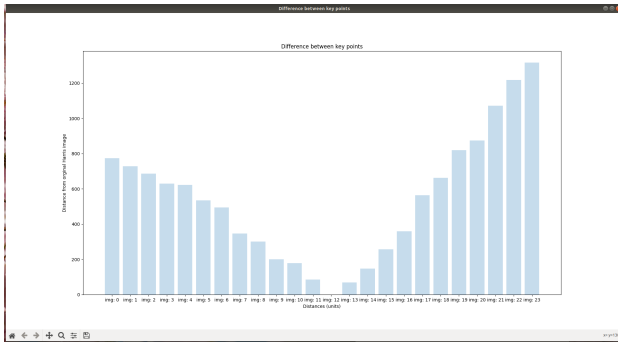


Fig. 8: Difference of keypoints found relative to first image: Harris, Scaled, Diamond

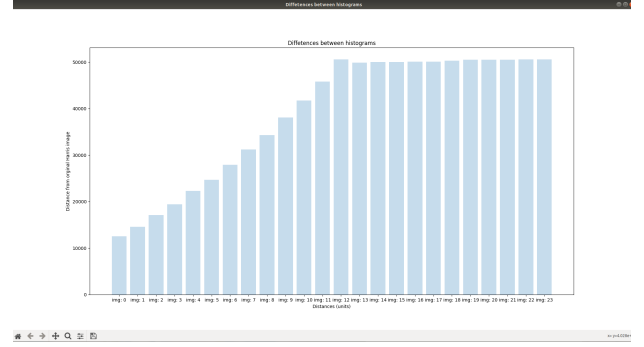


Fig. 9: Difference between the distances of histograms: Harris, scaled, Diamond

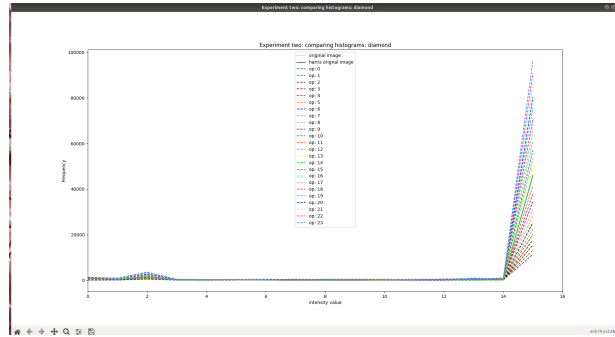


Fig. 10: All produced histograms: Harris, Scaled, Diamond

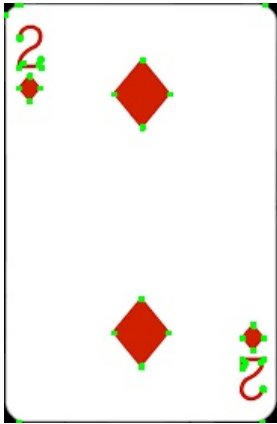


Fig. 11: Harris scaled original produced image: Harris, Scaled, Diamond

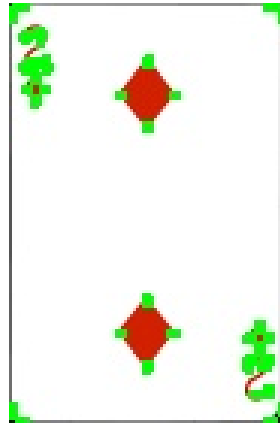


Fig. 12: Diamond arris image 1 scaled produced image

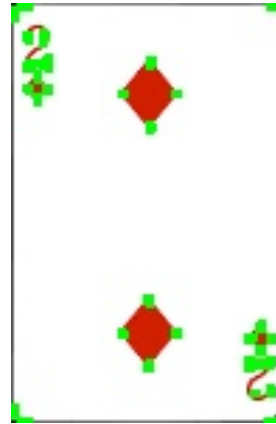


Fig. 13: Diamond harris image 2 scaled produced image

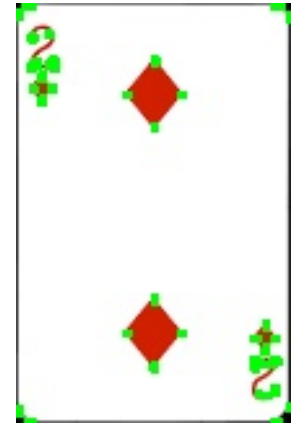


Fig. 14: Diamond harris image 3 scaled produced image

A set of 24 experimental images were produced. Only four displayed in report as displaying all 24 doesn't add great benefit to the reader. To re-view all 24 experimental images please implement the algorithms found in the appendixes.

two extra dimensions in image processing [7].

The next step in the algorithm is to calculate the LBP for each neighbouring cell. The algorithm will initialise a function window of $n \times n$ size which will scan through the image, and at each group of cells the linear binary pattern will compare the pixel value with the surrounding pixel values [5]. The algorithm will start at any set point inside the function window

and label the pixels in a clock-wise or a counter clockwise direction [5] [8]. If the current pixel is greater than the middle pixel the pixel would be labelled 1, otherwise it will be labelled 0 the labelling is also applicable in the other way thus for greater pixels label the pixel 1 otherwise, label the pixel 0. [8] [5].

Thereafter the algorithm will determine decimal represen-

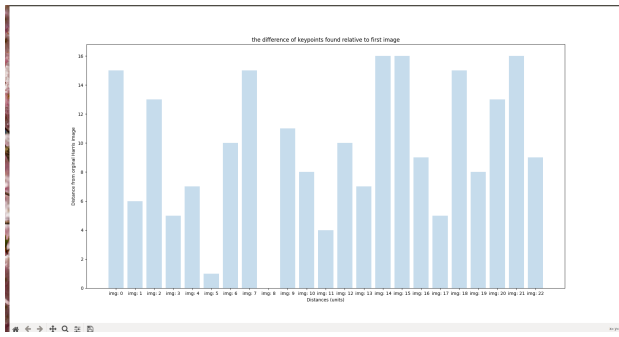


Fig. 15: Difference of keypoints found relative to first image: SIFT, Rotated, Diamond

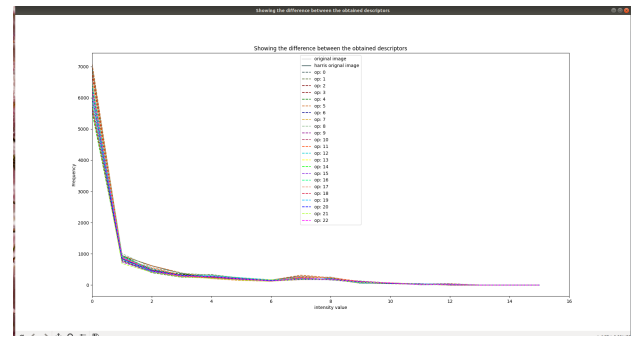
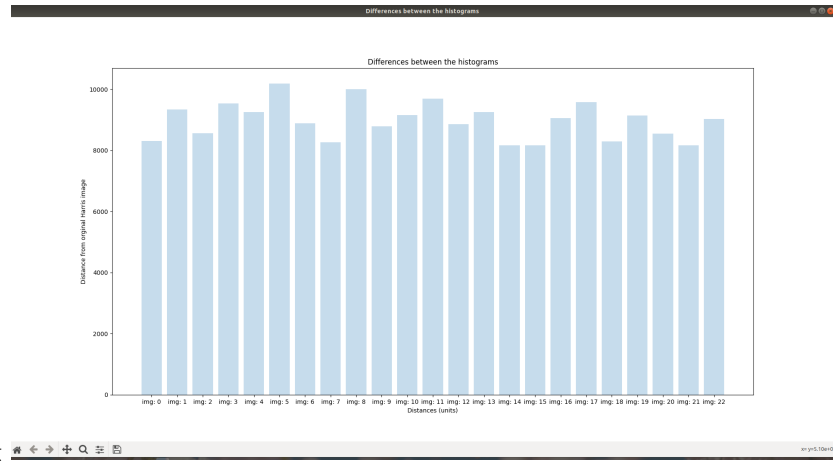


Fig. 16: Comparison of histograms for each transformed image: SIFT, Roateted, Diamond



differences between histograms.png

Fig. 17: Difference of distances between histograms relative to original: SIFT, Rotated, Diamond

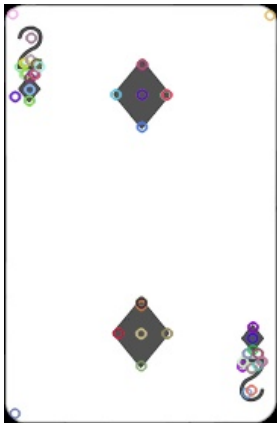


Fig. 18: Diamond SIFT rotated original produced image

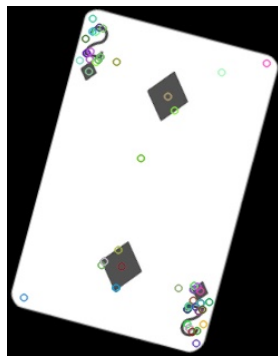


Fig. 19: Diamond SIFT image 1 rotated produced image

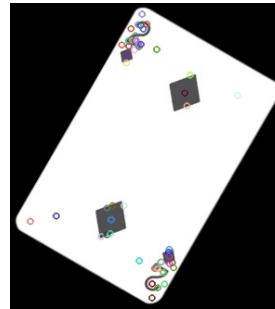


Fig. 20: Diamond SIFT image 2 rotated produced image

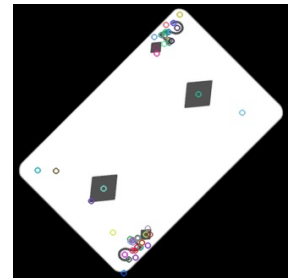


Fig. 21: Diamond SIFT image 3 rotated produced image

A set of 24 experimental images were produced. Only four displayed in report as displaying all 24 doesn't add great benefit to the reader. To re-view all 24 experimental images please implement the algorithms found in the appendixes.

tation of the produced binary pattern for the centre pixel. The algorithm will start from the same position as the labelling had began, transverse the window in the same direction as the

labelling had done [5]. This number is stored in an output LBP matrix [5]. This process is repeated indefinitely till all the pixels of the original image are mapped to an output LBP

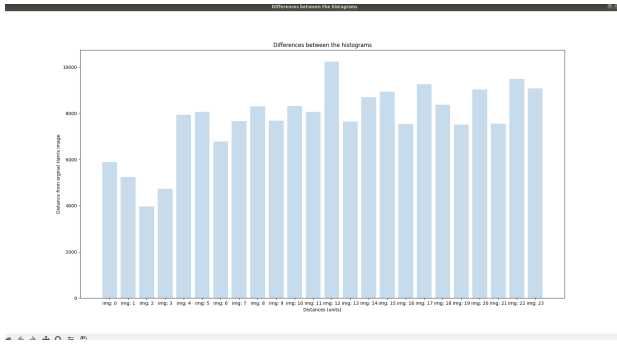


Fig. 22: Difference of keypoints found relative to first image: SIFT, Scaled, Diamond

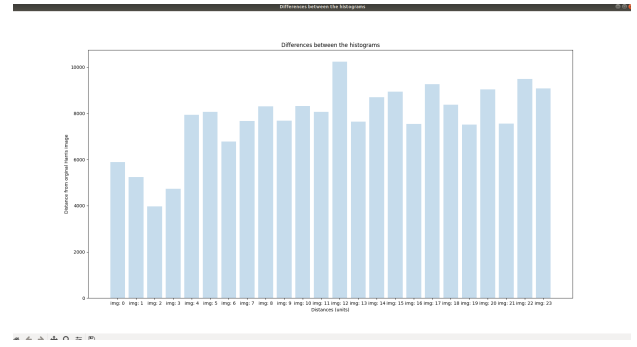


Fig. 23: Difference between the distances of histograms

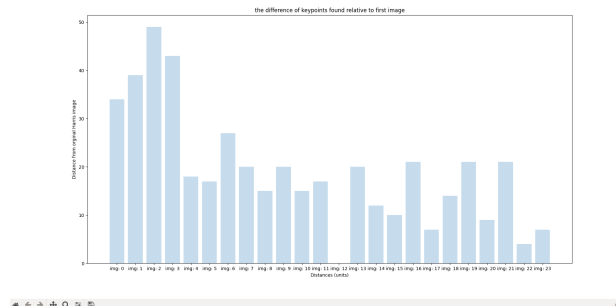


Fig. 24: All produced histograms: SIFT, Scaled, Diamond

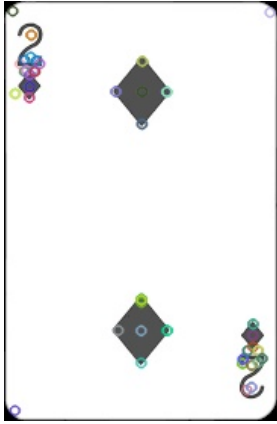


Fig. 25: Harris scaled original produced image: Harris, Scaled, Diamond

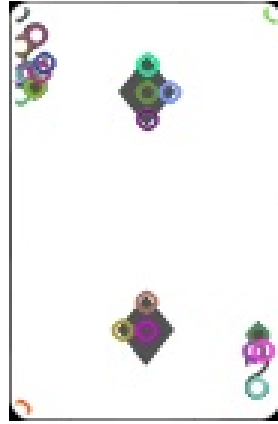


Fig. 26: Diamond SIFT image 1 scaled produced image



Fig. 27: Diamond SIFT image 2 scaled produced image

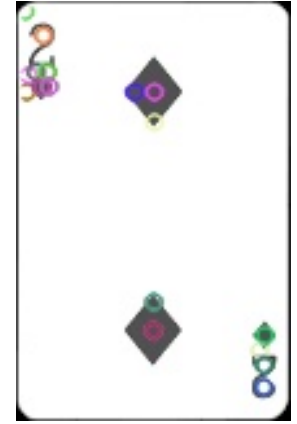


Fig. 28: Diamond SIFT image 3 scaled produced image

A set of 24 experimental images were produced. Only four displayed in report as displaying all 24 doesn't add great benefit to the reader. To re-view all 24 experimental images please implement the algorithms found in the appendixes.

matrix [5]. Consequently, the process will form a replica image of the original image although, this outputted matrix will be more descriptive about the textures found in the image [5].

Finally, the computation of histograms is done to the output matrix array in relation to the size of the window function hence if a 3 by 3 neighbourhood with 8 surrounding pixels this will yield 256 patterns ($2^8 = 256$) thus, being able

to form a histogram with a minimum value of 0, and a maximum value of 255 [5]. Therefore, forming a final feature vector of the image [5]. This is the main steps of the LBP algorithm, although they exists other implementations of the LBP algorithm such as the one proposed by Ojala et al [5]

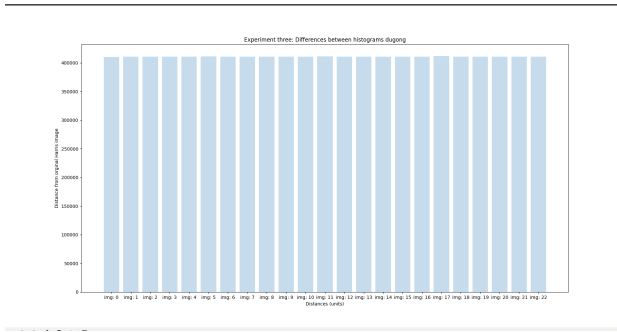


Fig. 29: Difference of keypoints found relative to first image: Harris, Rotated, Diamond

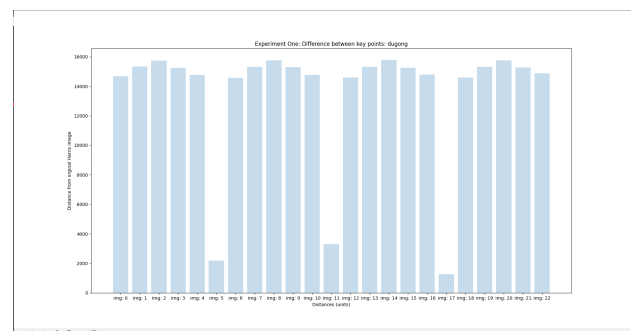


Fig. 30: Comparison of histograms for each transformed image: Harris, Roateted, Diamond

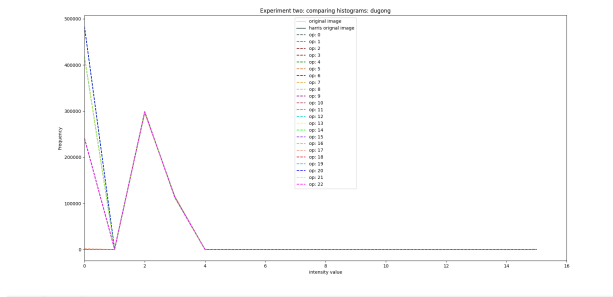


Fig. 31: Difference of distances between histograms relative to original: Harris, Rotated, Diamond



Fig. 32: Diamond harris rotated original produced image

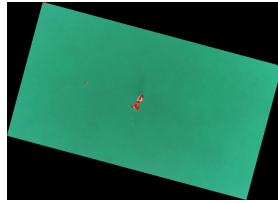


Fig. 33: Diamond harris image 1 rotated produced image

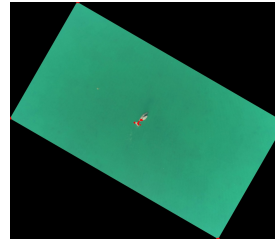


Fig. 34: Diamond harris image 2 rotated produced image

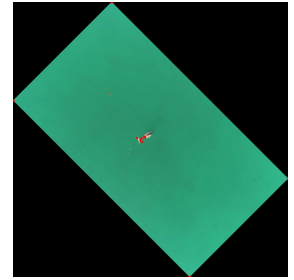


Fig. 35: Diamond harris image 3 rotated produced image

A set of 24 experimental images were produced. Only four displayed in report as displaying all 24 doesn't add great benefit to the reader. To re-view all 24 experimental images please implement the algorithms found in the appendixes.

B. part i: Benefits of LBP

- The ability to capture fine grained details in an image [5]. Thus making it suitable to accurately identifying the location of a cell [6].
- Computationally fast, as it's operations are done on a gray-scale image [7] [8] [10].
- few algorithms which gives you information about the texture of an image, as most algorithms give your information about corners and edges [8].

C. part i: Disadvantages of LBP

- cannot capture fine grained details at varying scales, this is only possible for a 3 by 3 function window [5]

[10]. Although, they has been an LBP implementation algorithm proposed by Ojala et al which overcomes this problem [5].

- For the case of cell identification, it greatly identifies many regions incorrectly relative to other feature detection algorithms [6].
- Due to that the algorithm's performs calculations based on a gray-scale image, the algorithm is susceptible to lighting variation [9].
- the algorithm is variant to image rotation [10].
- the algorithm is sensitive to noise [10]
- the algorithm is good for offering textual information of the whole image although, it lacks the ability to provide

between keypoints found.png

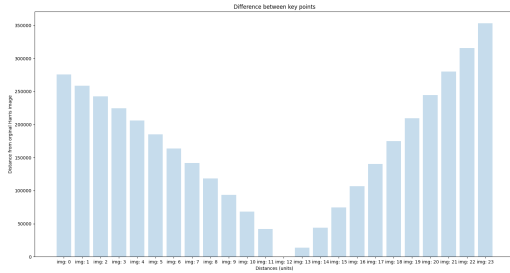


Fig. 36: Difference of keypoints found relative to first image: Harris, Rotated, Diamond

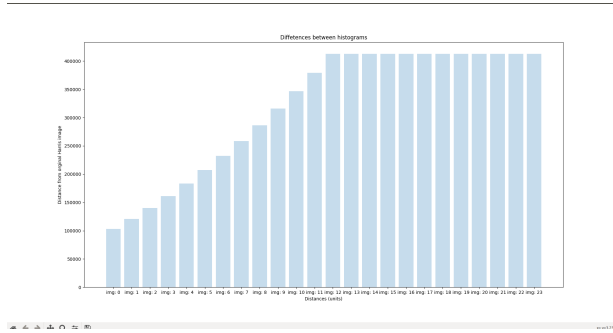


Fig. 38: Difference of keypoints found relative to first image: Harris, Rotated, Diamond



Fig. 39: Diamond harris scaled original produced image



Fig. 40: Diamond harris image 1 scaled produced image

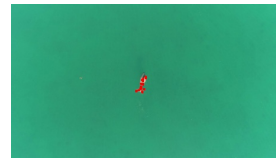


Fig. 41: Diamond harris image 2 scaled produced image

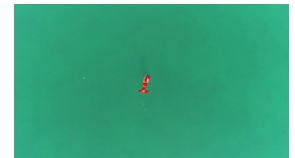


Fig. 42: Diamond harris image 3 scaled produced image

A set of 24 experimental images were produced. Only four displayed in report as displaying all 24 doesn't add great benefit to the reader. To re-view all 24 experimental images please implement the algorithms found in the appendixes.

local textural information [10].

D. part i: the main steps in histogram of gradients (HOG)

The HOG algorithm is typically used for feature detection for the applications of people, animal, and vehicle detection [8].

The main steps for HOG can be summarised as the following:

- Pre-processing
- Calculation of the gradient of the image
- Calculation of the HOG histograms for the neighbouring

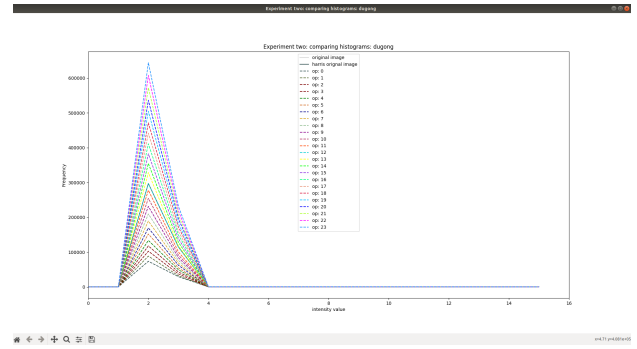


Fig. 37: Comparison of histograms for each transformed image: Harris, Rotated, Diamond

pixels

- Voting process between gradient direction, and gradient magnitude.
- normalisation of the histograms through a 16x16 function window
- calculation of the HOG feature gradient.

Firstly, before the applying the HOG algorithm the user has to ensure that the image meets all the requirements for the HOG algorithm. These requirements include that the image has to have a ratio of 1 to two in relation to width and height, any dimension can be used but the original HOG algorithm was designed to identify pedestrians hence a dimension of 64 x 128 was used [12]. Therefore, it may be required to crop out the patch which needs to be analysed and to be scaled [12]. In the original implementation of the algorithm gamma correction was included in this step although, the performance benefits are minor hence, for simplicity this step is typically ignored [12].

Secondly, the vertical and horizontal gradients of the image are calculated by taking the convolution of the image by

the following kernels $[-1, 0, 1]$, and $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ respectively [12]

[11]. In this step of the algorithm it can be noticed that the algorithm would have removed unnecessary information such as constant coloured background but would've highlighted the shapes of objects in an image [12]. Thereafter, once vertical and horizontal gradient matrices are created, the children matrices of gradient magnitude, and gradient angle are calculated whereby gradient magnitude is the euclidean distance $g = \sqrt{g_x^2 + g_y^2}$, and $\theta = \tan^{-1}(\frac{g_y}{g_x})$ [12] [11] [6]. The gradient magnitude and angle will be calculated for each channel for the respective color spaces used [12] [11]. Therefore, each pixel in an image will have their respective gradient magnitude, and angle assigned to them in separate matrices [12]. The proposed manner to store the gradient angle is to have them as signed angles hence for the angles been represented from unsigned angles of 0° to 360° the angles will be represented from 0° to 180° instead as this have been proven to work better [12] [11].

Thirdly, the calculation of the histograms of the image are partaking. The image is going to be divided into equal $n \times n$ cells, the size of n will be determined by dimension of the original image and the design choice, if the image had a dimension of 64x128 then 8x8 cells would be used for this step [12]. Representing, the image in this manner will make the representation of the image more compact, and makes the algorithm more robust against noise as individual gradients may have noise but the noise will be offset in the $n \times n$ function window.

Fourthly, the voting between the gradient magnitude and angle matrix is done into a 9 bin histogram [12]. The indexes of the bin array represent the gradient angles in increments of 20 degrees thus, the histogram will range from 0° to 160°, and the indexes will be assigned the gradient magnitude corresponding to a gradient angle index [12] [11]. Therefore,

if at the fifth index of the angle matrix has a value of 140°, and at the fifth index of the magnitude matrix they is a value of 69, 69 will be assigned to the seventh index of the histogram as that represents 140° [11] [12]. In the case, the angle sits between two histogram boundaries, the magnitude is split into those two indexes in proportion to how much that angle is in a bin index hence, if the ninth index has an angle of 50 and a magnitude of 180°, 90° will be placed into 2nd index and the remaining 90° will be placed into the 3rd index [12] [11]. This will create the feature descriptor for the image [12] [11].

Fifthly, the produced histogram is going to be normalized to ensure robustness to lighting [11] [12]. The histogram is normalised by taking the euclidean distance of each pixel in the respective color space hence for a BGR color space each cell will be normalised by the following $\sqrt{B^2 + G^2 + R^2} = X$ then each pixel at that respective index in the color space will be divided by the euclidean distance hence $\frac{Pixel/value}{X}$ [12] [11].

Finally, the calculation of the HOG feature gradient is done by taking all the individual histograms calculated and joining them together to form a giant vector which will represent the HOG feature vector [11] [12].

E. part i: the advantage of HOG

- powerful descriptors produced for natural scenes and object recognition [8]
- The HOG algorithm is invariant to changes in lighting (photometric) [8]
- The HOG algorithm is invariant to geometric changes hence, if you can reflect, transpose, scale, etc the object and the HOG algorithm will still detect the respective algorithm this is typically referred to as pose [8].
- address the common problem of objects obstruction another object, by in-front or behind an object of no-interest this is typically referred too partial occlusions [8].
- Although, the HOG algorithm isn't effective in capturing all the objects of an image, the algorithm will have less false positive relative to other algorithms [6].

F. part i: the disadvantages of HOG

- variant to rotation hence, rotation the image will cause the HOG algorithm to produce completely different results. [8].
- computationally expensive as you have to calculate the gradient, and magnitude for an image in all channels [12].

G. part i: the main steps in SIFT

The main steps of the SIFT algorithm can be summarised as the following.

- Determining extrema in produced scale spaces
- Locating the key-points in the scale space
- Assigning orientation labels to the found key-points in the space
- Formation of key point descriptors
- key point matching is done with all the key-points.

An image which has combination of larger corners, and smaller corners an algorithm can't use the same function window size to find all those corners as larger function windows are more suitable to find larger corners, and smaller function windows are more suitable to find smaller corner therefore, to combat against this you will use scale-space filtering [3]. Scale-space filtering is the process of creating the same images with different scales and performing a Laplacian of Gaussian (LoG) which is the convolution of a Laplacian kernel with the image matrix [3] **site the LoG thing as well**. LoG is performed on the image matrix with varying threshold values (σ) lower sigma values will detect smaller corners better, and high threshold values will detect larger corners better [3]. This process will find the local maxima of pixel intensities (corners) across the image, and images in different scale spaces which is typically referred to as octaves [3]. Convolution by LoG is an extremely computationally demanding and be very costly for processing power therefore, an approximation is used which is referred to as difference gaussians (DoG) [3]. LoG/DoG will result in a matrix which has the second derivative of each pixel hence, the algorithm will note all points which are local minimums or local maximums ($\frac{d^2y}{dx^2} = 0$) [3]. Then after, the algorithm will compare the found local maximums at each octave with the octave above it and below it, and if the local maximums coincide with one another that pixel is a potential key-point [3].

Secondly, the list of potential key-points must be refined to find actual key points [3]. This refinement process is done through the application of Taylor series expansion through each scale space identify the actual corners in the image by filtering the corner through a contrast thresh-hold hence, if it's below the threshold the corner is going to be rejected [3]. The consequent application of DoG, is that DoG will have a high response time for edges hence, edges need to be filtered by comparing the eigenvalue of the point of interest with the edge threshold, if the eigenvalue is greater than the edge threshold that point is discarded [3]. After these process the remaining points will be strong points of interest [3].

Thirdly, the strong points of interest are assigned an orientation dependent on their octave level and characteristics this is to make SIFT invariant to rotation although this is only applicable for angles up to 30° [3].

Fourthly, to create robustness against illumination changes, and rotation a 16 by 16 descriptor is created around the key points which were found in the image in which 8 bin orientation histograms histograms are created [3].

Finally, for images in the octave the algorithm will match key points in relation the adjacent scaled image by identifying the nearest neighbour [3]. A ratio between between the closest-distance to the second closest distance is take if that's greater than 0.8 that key point is rejected this eliminates a great proportion of the false matches.

H. part i: the advantages of SIFT

The advantages of SIFT are the following:

- It' scale invariant [8]

- invariant to translation, rotation and re-scaling to a certain degree [8]
- poor stability for small affine changes [13]
- Can robustly identify objects even among clutter, and partial occlusion [13].
- affine invariance [13]

I. part i: the disadvantage of SIFT

The disadvantages of SIFT are the following:

- it's very computationally expensive [8].

J. Common and different components between SIFT, HOG, and LBP

The LBP algorithm is the only algorithm out of the three algorithms which describes texture of the image, as SIFT looks for corners in an image, and HOG finds the outlines of the object. LBP is relative simple algorithm, and does its calculations on a gray-scale image hence, it's computationally fast while SIFT, and HOG are slower. SIFT will be the slowest algorithm out of the three because of the extra calculations done in each octave, and the performance of LoG/DoG. SIFT is the only algorithm which is a feature detector, and feature descriptor. The LBP, HOG, and SIFT algorithm are all features descriptors with different implementations. SIFT's descriptor is a descriptor is for orientation, LBP's descriptors is for the binary pattern, and HOG's descriptor is for the gradients of the image. All algorithm's perform calculations on neighbouring pixels. HOG and LBP are typically used in conjunction with other algorithms to boost the performance of other algorithms while SIFT is typically a stand alone algorithm. SIFT is the only algorithm which does comparisons with images of different scales. HOG and SIFT require the scaling of their images to initialise their algorithm.

III. TASK THREE

A. Results for Diamond playing card

B. Results for Dugong

Don't forget to include the areas found by each object extracted

Don't forget to insert the tables for the area found for each object in here as well

C. Results for Dugong

don't forget to insert the results for the area found by the algorithm

don't forget to insert to insert the results for the area found by the algorithm here

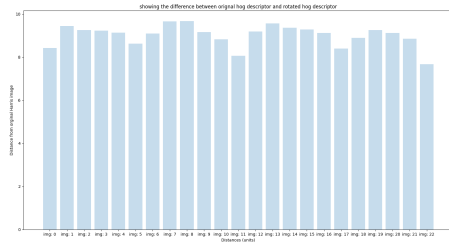


Fig. 43: Diamond: comparison between the descriptors found for original image, and rotated images

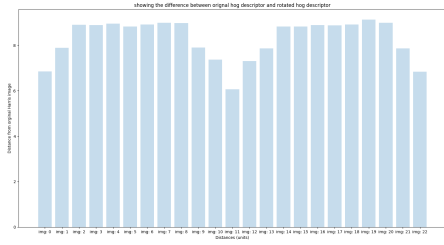


Fig. 45: Diamond: comparison between the descriptors found for original image, and rotated images

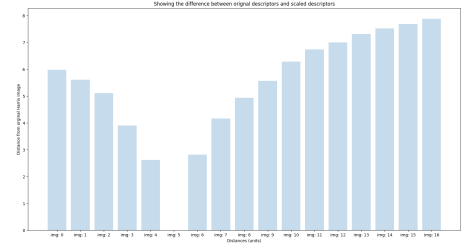


Fig. 44: Diamond: comparison between the descriptors found for original image, and scaled images

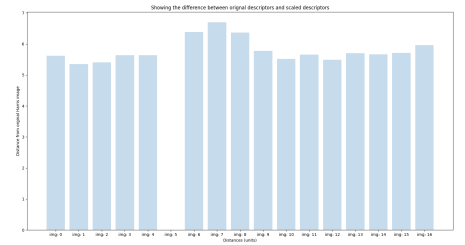


Fig. 46: Diamond: comparison between the descriptors found for original image, and scaled images

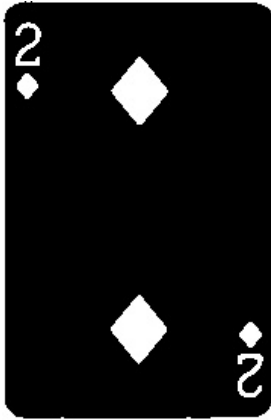


Fig. 47: Diamond: Separated fore-ground and background

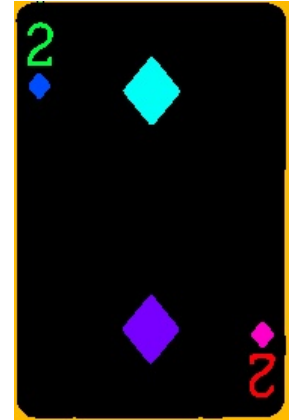


Fig. 48: Diamond: the labelling of each object found in the image

IV. TASK FOUR: IMAGE SEGMENTATION WITH K-MEANS

- A. Diamond: Image Segmentation: with HSV color space
- B. Diamond: Image Segmentation: with BGR color space
- C. Diamond: Image Segmentation: with contouring
- D. Dugong: Image Segmentation: with HSV color space
- E. Dugong: Image Segmentation: with BGR color space
- F. Dugong: Image Segmentation: with contouring



Fig. 49: Dugong: Separated fore-ground and back-ground

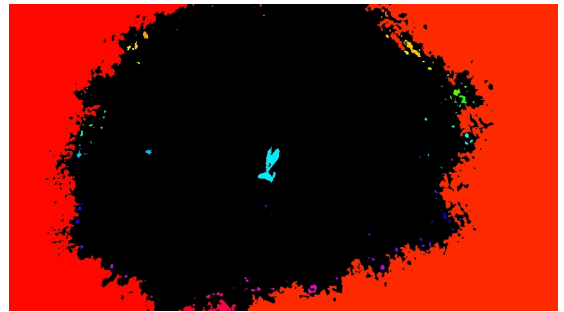


Fig. 50: Dugong: the labelling of each object found in the image

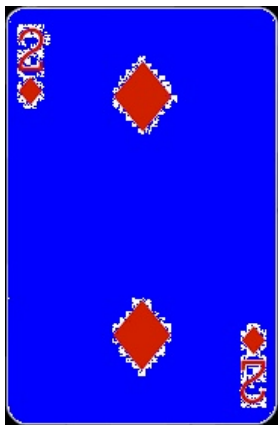


Fig. 51: Diamond: 1st cluster found by k-means

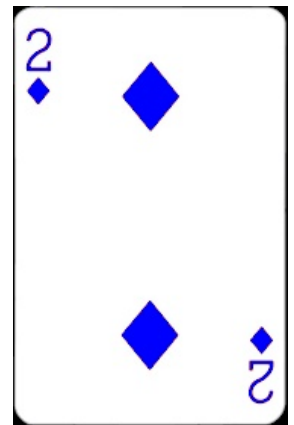


Fig. 52: Diamond: 2nd cluster found by k-means

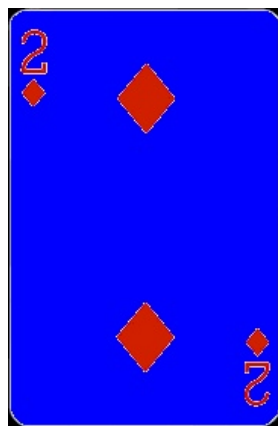


Fig. 53: Cluster found by the BGR color space

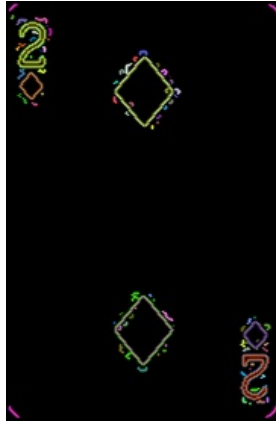


Fig. 54: Image segmentation done by contouring



Fig. 55: Dugong: 1st cluster found by k-means

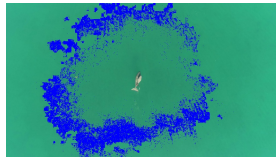


Fig. 56: Dugong: 2nd cluster found by k-means

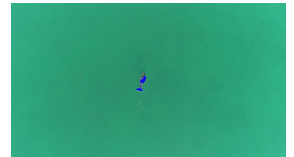


Fig. 57: Dugong: 3rd cluster found by k-means

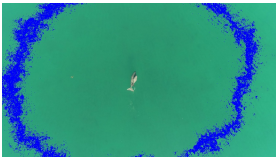


Fig. 58: Dugong: 1st cluster found by k-means

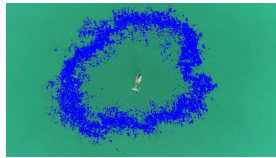


Fig. 59: Dugong: 2nd cluster found by k-means

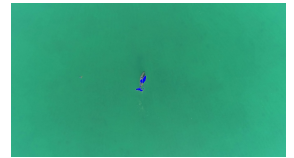


Fig. 60: Dugong: 3rd cluster found by k-means



Fig. 61: Dugong: Image segmentation done by contouring

APPENDIX

APPENDIX A

TASK 1 CODE PRINT OUT

APPENDIX B

TASK 2 CODE PRINT OUT

APPENDIX C

FULL CODE PRINT OUT WITH SUPPORTING FUNCTIONS

A. *Main.py*

REFERENCES

- [1] Dr. S. An. 2020. Machine Perception Lecture 03 [PowerPoint slides] Available: https://learn-ap-southeast-2-prod-fleet01-xythos.s3.ap-southeast-2.amazonaws.com/5dc3e34515a0e/4348643?response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27lecture03_feature_detection.pdf&response-content-type=application%2Fpdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20200927T060000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=AKIAYDKQORRYZBCCQFY5%2F20200927%2Fap-southeast-2%2Fs3%2Faws4_request&X-Amz-Signature=bcf107d5e76759efe45687b516ac725158189de0af558d0a7a798df1723f4299
- [2] A. Mordvinste and K. Abid. "Harris Corner detection". OpenCV-Python Tutorials. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html (retrieved Sept. 27, 2020).
- [3] OpenCV. "Introduction to SIFT (Scale-Invariant Feature Transform)". OpenCV-Open source Computer vision. https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html
- [4] C. Harris and M. Stephens. 1988. A Combined Corner And Edge Detector. Plessey Reasearch Roke Manor, UK. [Online]. Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.434.4816&rep=rep1&type=pdf>
- [5] A. Rosebrock. December, 2015. "Local Binary Patterns with Python & OpenCV". pyimagesearch. <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>
- [6] S. Ozturk, and B.Akdemir. "Comparison of HOG, MSER, SIFT, FAST, LBP and CANNY features for cell detection in histopathological images " in Helix the scientific Explorer. Feb 2018. [Online]. Available: <http://helix.dnares.in/wp-content/uploads/2018/05/3321-3325.pdf>
- [7] Kanan C, Cottrell GW (2012) Color-to-Grayscale: Does the Method Matter in Image Recognition? PLoS ONE 7(1): e29740. doi:10.1371/journal.pone.0029740
- [8] Dr. S. An. 2020. Machine Perception Lecture 04 [PowerPoint slides] Available: https://learn-ap-southeast-2-prod-fleet01-xythos.s3.ap-southeast-2.amazonaws.com/5dc3e34515a0e/6301343?response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27lecture04_feature_extraction_2019%25281%2529.pdf&response-content-type=application%2Fpdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20201005T060000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=AKIAYDKQORRYZBCCQFY5%2F20201005%2Fap-southeast-2%2Fs3%2Faws4_request&X-Amz-Signature=c3d82f1e880c0787db8eb858336ae4e82df4d25ad29ea5a9563494a6f57cc6d4
- [9] C. H. Chan. "Multi-scale Local Binary Pattern Histogram for Face Recognition". The Universtiy of Surrey, Surrey, Guildford, U.K. gu27xh. September 2008. [Online]. Available: https://www.researchgate.net/profile/Chi-Ho-Chan/publication/224936218_Multi-scale_Local_Binary_Pattern_Histograms_for_Face_Recognition/links/00b495225c98186d23000000/Multi-scale-Local-Binary-Pattern-Histograms-for-Face-Recognition.pdf
- [10] . Z. Tang, Y. SU, M. J. Er, F. Qi, L. Zhang, and J. Zhou. "A local Binary pattern based texture descriptors for classification of tea leaves." in Neurocomputing. December 2014. [Online]. Available: <https://reader.elsevier.com/reader/sd/pii/S0925231215006475?token=53A42C9FF9C438422CB044F70C414EA9C464364C8C5A90D9A510E9D71B152E4E0ABB5302B>
- [11] L. Weng. Oct, 2017. "Object Detection for Dummies Part 1: Gradient Vector, HOG, and SS. Lil'Log. <https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html>
- [12] S. Mallick. "Histogram of Oriented Gradients". learn OpenCV. December 2016. <https://www.learnopencv.com/histogram-of-oriented-gradients/>
- [13] D. G. Lowe. "Distinictive Image Features form Scale-Invariant Keypoints". The University of British Columbia. Vancouver . B.C. Canada. January 2004. [Online]. Available: <https://www.cs.ubc.ca/lowe/papers/ijcv04.pdf>