

|  |  |
| --- | --- |
| **ASSIGNMENT** | |
| **Course Code** | CSE407A |
| **Course Name** | Computer Vision |
| **Programme** | B. Tech. |
| **Department** | Computer Science & Engineering |
| **Faculty** | Faculty of Engineering & Technology |

#### 

|  |  |
| --- | --- |
| **Name of the Student** | Subhendu Maji |
| **Reg. No** | 18ETCS002121 |
| **Semester/Year** | 7th semester / 2018 batch |
| **Course Leader/s** | Dr. Subarna Chatterjee |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Declaration Sheet** | | | | | | | | |
| Student Name | Subhendu Maji | | | | | | | |
| Reg. No | 18ETCS002121 | | | | | | | |
| Programme | B. Tech. | | | | | Semester/Year | 7th sem /2018 batch | |
| Course Code | CSE407A | | | | | | | |
| Course Title | Computer Vision | | | | | | | |
| Course Date |  | | to | |  | | | |
| Course Leader | Dr. Subarna Chatterjee | | | | | | | |
| **Declaration**  The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of university regulations and will be dealt with accordingly. | | | | | | | | |
| Signature of the Student | |  | | | | | Date |  |
| Submission date stamp  (by Examination & Assessment Section) | |  | | | | | | |
| Signature of the Course Leader and date | | | | Signature of the Reviewer and date | | | | |
|  | | | |  | | | | |

# **Contents**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[**Declaration Sheet** ii](#_Toc88474947)

[**Contents** iii](#_Toc88474948)

[Marking Scheme 4](#_Toc88474949)

[**Question No. 1** 5](#_Toc88474950)

[1.1 Implement the noise reduction for the noisy image Q1\_1.tif and submit your code and the denoised image. 5](#_Toc88474951)

[1.2 Implement the histogram equalization to the input images Q1\_2.tif and Q1\_3.tif and submit your code and the output images. 7](#_Toc88474952)

[1.3 Reduce the salt-and-pepper noise from the input image Q1\_4.tif and submit your code and the output image. 9](#_Toc88474953)

[1.4 Extract the gradient parts from the input image Q1\_5.tif and submit your code and the output image. 10](#_Toc88474954)

[1.5 Conclusion and Recommendation 10](#_Toc88474955)

[**Question No. 2** 11](#_Toc88474956)

[2.1 Introduction to the Segmentation techniques 11](#_Toc88474957)

[2.2 Identify and explain the appropriate pre-processing techniques. 14](#_Toc88474958)

[2.3 Identify and explain the appropriate Segmentation techniques. 18](#_Toc88474959)

[**Question No. 3** 22](#_Toc88474960)

[3.1 Perform pre-processing on the images of the created dataset. 22](#_Toc88474961)

[3.2 Perform segmentation to segment the image. 23](#_Toc88474962)

[3.3 Results and Discussions. 27](#_Toc88474963)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Questions | Marking Scheme | | Marks | | |
| Max Marks | First Examiner Marks | Moderator |
| 1 |  | | | | |
| 1.1 | Implementation of noise reduction filter | 5 |  |  |
| 1.2 | Implementation of histogram equalization filter | 7 |  |  |
| 1.3 | Reduction of the salt-and-pepper noise from the input image | 4 |  |  |
| 1.4 | Extraction of the gradient parts from the input image | 4 |  |  |
| 1.5 | Conclusion and Recommendations | 3 |  |  |
| **Question 1 Max Marks** | | **23** |  |  |
| 2 | 2.1 | Introduction to Segmentation and Creation of Dataset. | 3 |  |  |
| 2.2 | Identification and explanation of the appropriate pre-processing techniques | 5 |  |  |
| 2.3 | Identification and explanation of the appropriate Segmentation techniques | 2 |  |  |
|  | **Question 2 Max Marks** | | **10** |  |  |
| 3 | 3.1 | Pre-processing on the images of the created dataset | **5** |  |  |
| 3.2 | Segmentation of the images. | **7** |  |  |
| 3.3 | Results and Discussions. | **5** |  |  |
|  | **Question 3 Max Marks** | | **17** |  |  |
| **Total Assignment Marks** | | | 50 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Course Marks Tabulation** | | | | |
| **Question** | **First Examiner** | **Remarks** | **Moderator** | **Remarks** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| **Marks (Max 50)** |  |  |  |  |
| **Signature of First Examiner Signature of Moderator** | | | | |

# **Question No. 1**

**Solution to Question No. 1:**

## Implement the noise reduction for the noisy image Q1\_1.tif and submit your code and the denoised image.

As seen in the input image *Q1\_1.tif*, there is a constant repeating noise pattern on the image, this usually happens in the case of periodic noise.

Periodic noise in an image arises typically from electrical or electromechanical interference during image acquisition. This gives rise to regular noise patterns in an image. Frequency domain techniques in the Fourier domain are most effective at removing periodic noise.

To remove periodic noise from an image, we need to remove a particular range of frequencies from that image. This can be done using **Band-reject filter**, in ideal band reject, the centered Fast Fourier Transform (FFT) is filtered by the following function, where  is the lower bound of the frequency band,  is the upper bound of the frequency band, and *)* is the distance between a point *(u, v)* in the frequency domain and the center of the frequency rectangle.

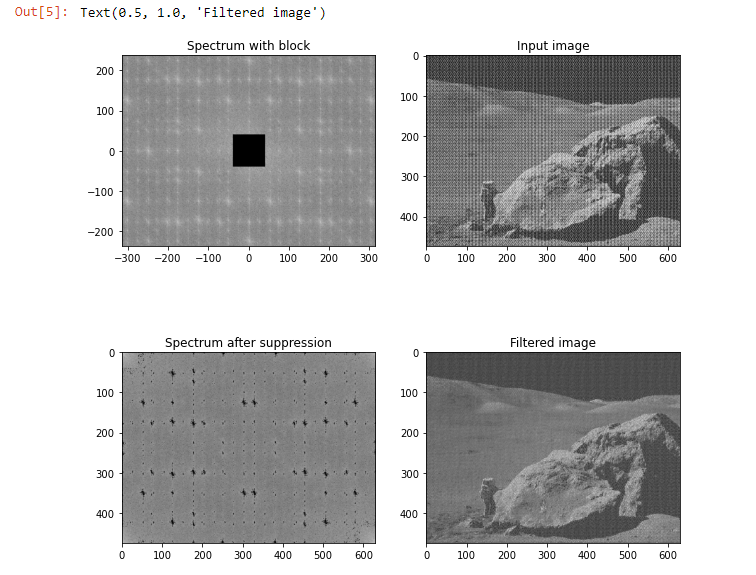
Diagram

Description automatically generated with medium confidence









## Implement the histogram equalization to the input images Q1\_2.tif and Q1\_3.tif and submit your code and the output images.

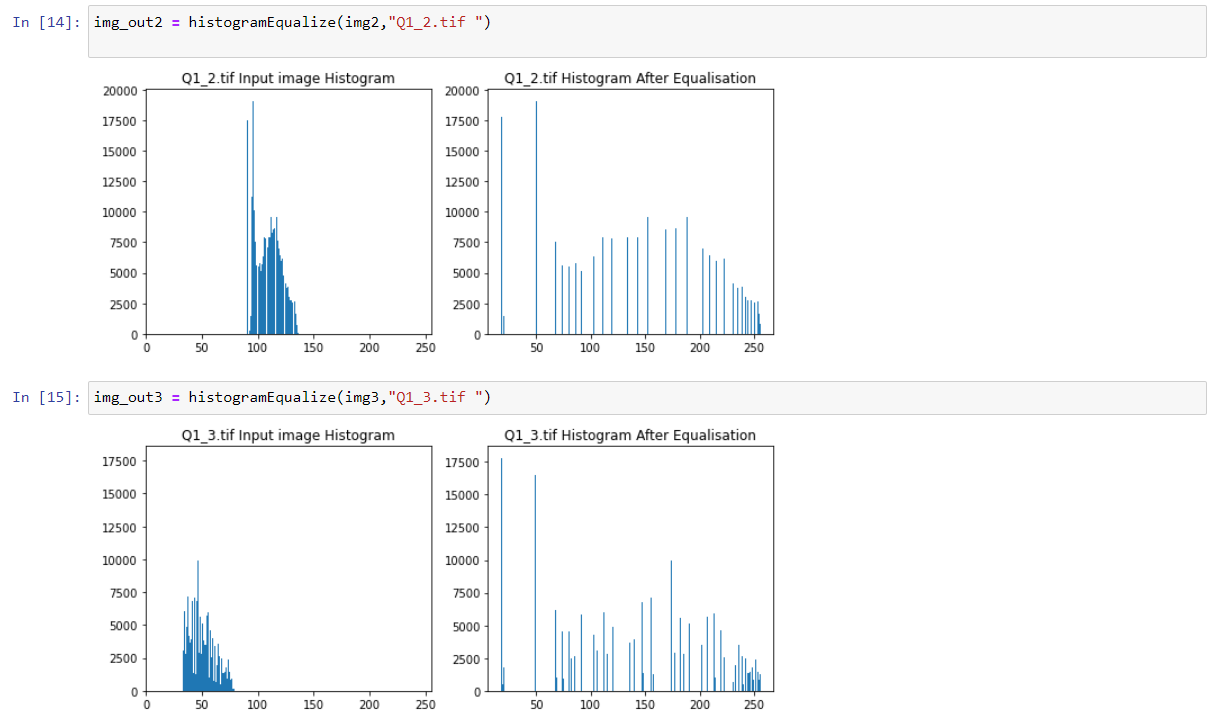
Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the image is represented by a narrow range of intensity values. Through this adjustment, the intensities can be better distributed on the histogram utilizing the full range of intensities evenly. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the highly populated intensity values which use to degrade image contrast.

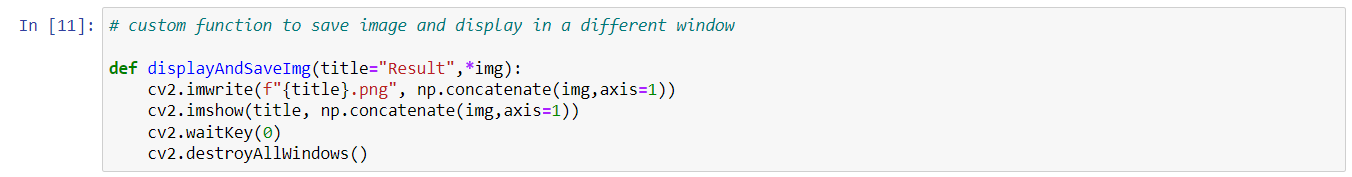
The method is useful in images with backgrounds and foregrounds that are both bright or both dark.

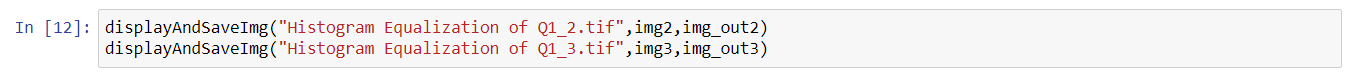
In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are either over or under-exposed.

* A key advantage of the method is that it is a fairly straightforward technique adaptive to the input image and an invertible operator. So, in theory, if the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive.
* A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background noise, while decreasing the usable signal.









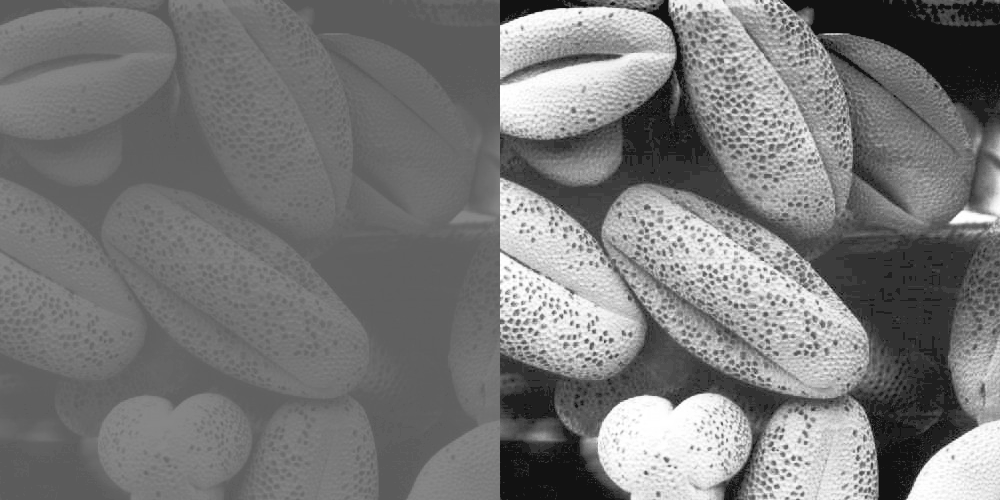


Figure histogram equalized on Q1\_2.tif

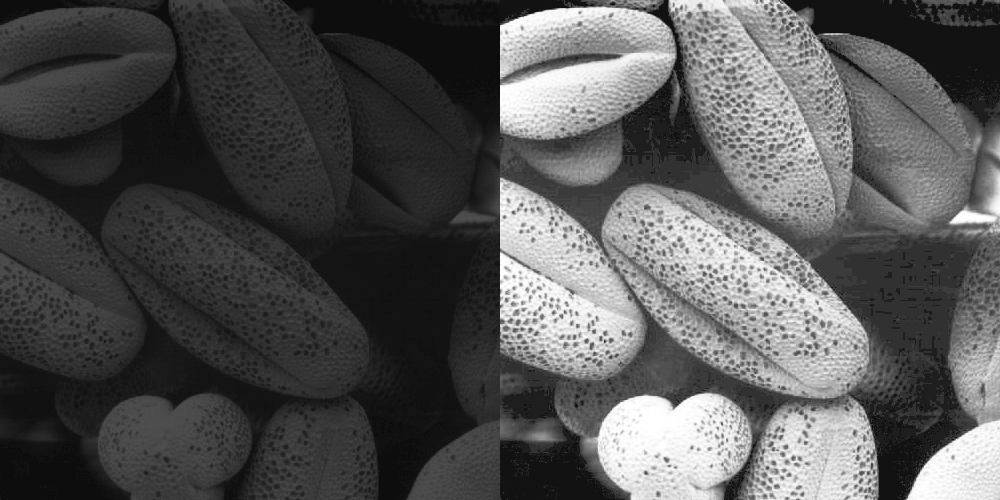


Figure histogram equalized on Q1\_3.tif

## Reduce the salt-and-pepper noise from the input image Q1\_4.tif and submit your code and the output image.

Salt-and-pepper noise is a form of noise sometimes seen on images. It is also known as **impulse noise.** This noise can be caused by sharp and sudden disturbances in the image signal. It presents itself as sparsely occurring white and black pixels. An effective noise reduction method for this type of noise is a **median filter.**



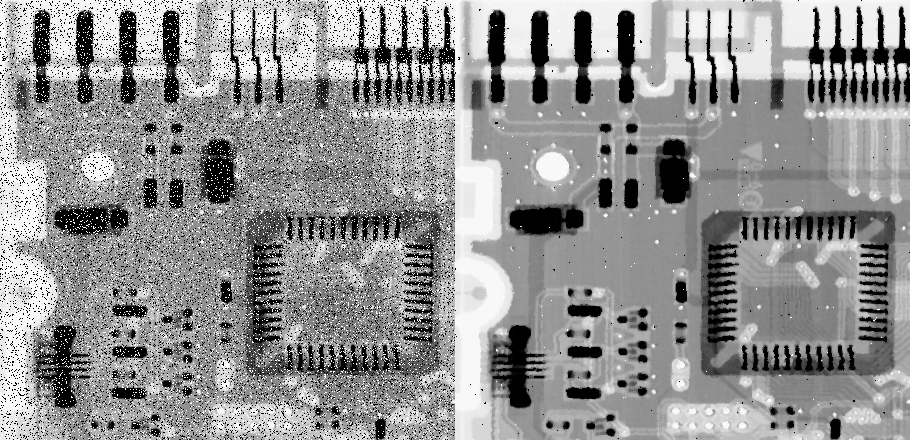


Figure Reduced Salt and Pepper noise using Median filter from Q1\_4.tif

## Extract the gradient parts from the input image Q1\_5.tif and submit your code and the output image.

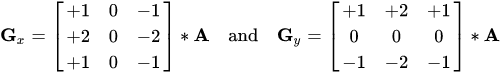
An image gradient is a directional change in the intensity or color in an image. The gradient of the image is one of the fundamental building blocks in image processing.

For example, the Canny edge detector uses image gradient for edge detection. The most common way to approximate the image gradient is to convolve an image with a kernel, such as the **Sobel operator** or **Prewitt operator**.

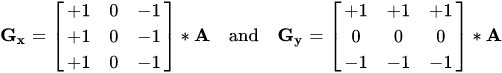
**Sobel and Prewitt Filter**

The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. If we define A as the source image, and and are two images which at each point contain the horizontal and vertical derivative approximations respectively, the computations are as follows:

**Sobel Operator**



**Prewitt Operator**



**Final gradient**



**Gaussian Smoothing**

Gaussian smoothing is commonly used with edge detection. Most edge-detection algorithms are sensitive to noise; the 2-D Laplacian filter, built from a discretization of the Laplace operator, is highly sensitive to noisy environments.





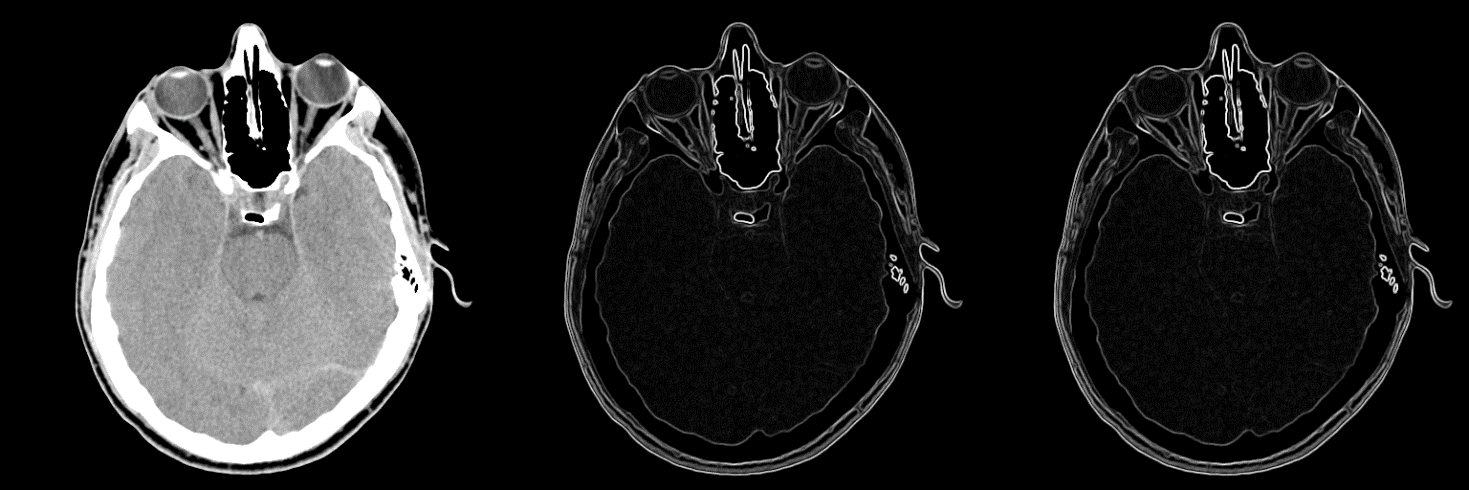


Figure Sobel and Prewitt filter applied on Q1\_5.tif to extract gradient parts

## 1.5 Conclusion and Recommendation

As it can be seen in the image *Q1\_1.tif*, there is a constant repeating noise pattern on the image, this usually happens in the case of periodic noise. To remove periodic noise from an image, we need to remove a particular range of frequencies from that image. This can be done using **Band-Reject filter.**

In section 1.2, we were asked to implement histogram equalization. Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the image is represented by a narrow range of intensity values. We can see in (Figure 1, Figure 2) histogram equalized images have balanced contrast.

In section 1.3, we were asked to reduce Salt and Pepper noise which can effectively remove by median filter.

In section 1.4, we were asked to extract gradient parts of an image. The most common way to approximate the image gradient is to convolve an image with a kernel, such as the **Sobel operator** or **Prewitt operator**.

Hence, we have implemented band reject filter, histogram equalizer, median filter, Sobel and Prewitt filter successfully. Although all the filters are working as expected but there is a delay in getting the result. This is due to the reason we used Python 3.9 (comparatively slower than some other programming language) for the implementation. We can get a faster result using MATLAB.

# **Question No. 2**

**Solution to Question No. 2:**

## 2.1 Introduction to the Segmentation techniques

Image segmentation is the process of dividing an image (a digital image) into multiple disjoint regions, where pixels in the same region share similar properties or belong to the same object. Image segmentation changes the representation of an object, making it meaningful and easier to analyze.

Image segmentation has variety of applications.

* Medical Imaging

Image segmentation is applied extensively in medical applications to locate tumors and other pathologies, measure tissue volume, diagnosis and surgery planning, simulation, and navigation.

* Recognition

Image segmentation is applied in face recognition to identify multiple faces in an image, fingerprint recognition to highlight the contours and edges of a fingerprint and in iris recognition.

* Object detection

Image segmentation is applied in pedestrian detection, to identify the pedestrians on a road, brake light segmentation, and locating objects in satellite images.

Image segmentation is a mid-level processing technique used to analyze the image and can be defined as a processing technique used to classify or cluster an image into several disjoint parts by grouping the pixels to form a region of homogeneity based on the pixel characteristics like gray level, color, texture, intensity and other features. The main purpose of the segmentation process is to get more information in the region of interest in an image which helps in annotation of the object scene. Image segmentation aims at domain-independent partition of the image into a set of visually distinct and homogeneous regions with respect to certain properties. The main goal of segmentation is to clearly differentiate the object and the background in an image.

If represents an image, then the image segmentation is simply division of into subregions such that

and is governed by following set of rules:

1. is a connected set, .
2. for all i and j
3. = True for .
4. = False for adjoint regions, and

Where, is a logical predicate.

The rules described above mentions about continuity, one-to-one relationship, homogeneity and non-repeatability of the pixels after segmentation respectively.

Dataset creation is a long and drawn-out process which involves collection of the data samples (such as images, or any other required data) and to label the images in an appropriate manner which depends on the problem that is being tackled. Dataset creation plays an important role in segmentation algorithms that involve neural networks. Errors during the creation of a dataset significantly impacts the performance of the segmentation model, often leading to disastrous consequences.

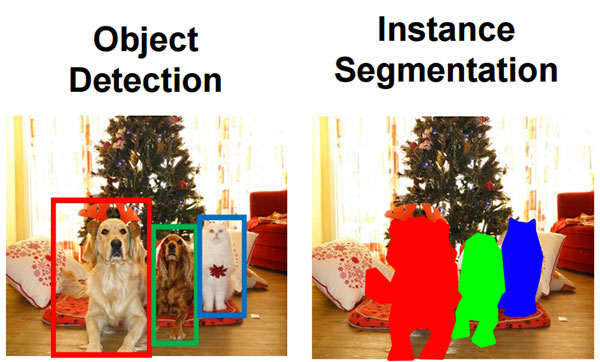


Figure Example of Segmentation of an image

## 2.2 Identify and explain the appropriate pre-processing techniques.

Image preprocessing techniques help to prepare the image for segmentation. Almost always, the image to segment that is obtained in a production environment are not as “perfect” as the ones that are available during testing or deployment. Often, the images received will have noise, objects in the image will not be aligned properly, the image can be blurry, etc.

The primary focus on pre-processing techniques in the training phase of a neural network-based segmentation algorithm is to augment the dataset and try to represent real world scenarios during training.

Image processing could be simple tasks like image resizing. In order to feed a dataset of images to a convolutional network, they must all be the same size. Other processing tasks can take place like geometric and color transformation or converting color to grayscale and many more.

Data preprocessing techniques might include:

1. Convert color images to grayscale to reduce computation complexity

In certain problems we’ll find it useful to lose unnecessary information from our images to reduce space or computational complexity.

For example, converting our colored images to grayscale images. This is because in many objects, color isn’t necessary to recognize and interpret an image. Grayscale can be good enough for recognizing certain objects. Because color images contain more information than black and white images, they can add unnecessary complexity and take up more space in memory (Remember how color images are represented in three channels, which means that converting it to grayscale reduces the number of pixels that need to be processed).

1. Standardize images

One important constraint that exists in some machine learning algorithms, such as CNN, is the need to resize the images in our dataset to a unified dimension. This implies that our images must be preprocessed and scaled to have identical widths and heights before fed to the learning algorithm.

1. Data augmentation

Another common pre-processing technique involves augmenting the existing dataset with perturbed versions of the existing images. Scaling, rotations and other affine transformations are typical. This is done to enlarge our dataset and expose the neural network to a wide variety of variations of our images. This makes it more likely that our model recognizes objects when they appear in any form and shape. Here’s an example of image augmentation applied to a butterfly image:

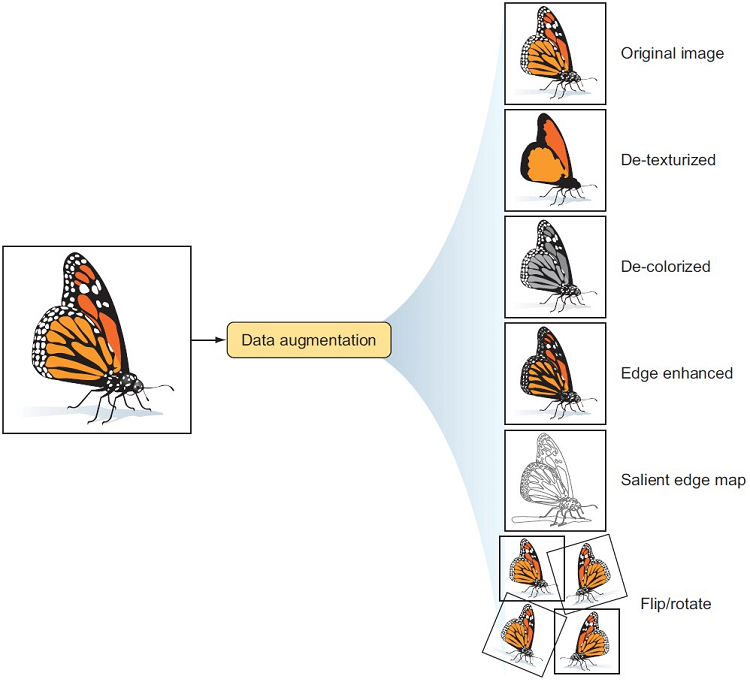


Figure Data augmentation example on a butterfly image

1. Other techniques

Many preprocessing techniques can be used to get our images ready to train the machine learning model. In some projects, we might need to remove the background color from our images to reduce the noise. Other projects might require that we brighten or darken our images. In short, any adjustments that we need to apply to our dataset are considered a sort of preprocessing. And we’ll select the appropriate processing techniques based on the dataset at hand and the problem we’re solving. That builds our intuition of which ones we need when working on our own projects.

Identifying the correct preprocessing and augmentation steps most useful for increasing model performance requires a firm understanding of the problem, data collected, and production environment. What may work well in one situation is not appropriate in all others.

Thus, considering techniques and why each may be valuable enables informed decisions. In this post, we’ll surface considerations and provide recommendations that are generally best.

1. Resize

Changing the size of an image sounds trivial, but there are considerations to take into account.

Many model architectures call for square input images, but few devices capture perfectly square images. Altering an image to be a square call for either stretching its dimensions to fit to be a square or keeping its aspect ratio constant and filling in newly created “dead space” with new pixels. Moreover, input images may be various sizes, and some may be smaller than the desired input size.

1. Orientation

When an image is captured, it contains metadata that tells our machines the orientation by which to display that input image relative to how it is stored on disk. That metadata is called its EXIF orientation, and inconsistent handling of EXIF data has long been a bane of developers everywhere.

This applies to models, too: if we’ve created annotated bound boxes on how we perceived an image to be oriented but our model is “seeing” that image in a different orientation, we’re training the model completely wrong!

1. Grayscale

Color changes are an example of image transformations that may be applied to all images (train and test) or randomly altered in training only as augmentations. Generally, gray scaling is a color change applied to all images. While we may think “more signal is always better; we should show the model color,” we may see timelier model performance when images are gray scaled. Color images are stored as red, green, and blue values, whereas grayscale images are only stored as a range of black to white. This means for CNNs, our model only needs to work with one matrix per image, not three.

1. Random Flips

Randomly mirroring an image about its x- or y-axis forces our model to recognize that an object need not always be read from left to right or up to down. Flipping may be illogical for order-dependent contexts, like interpreting text.

1. Random Rotations

Rotating an image is particularly important when a model may be used in non-fixed position, like a mobile app. Rotating can be tricky as it, too, generates “dead pixels” on the edges of our images and, for bounding boxes, requires trigonometry to update any bounding boxes.

1. Random Exposure

Adjusting image brightness to be randomly brighter and darker is most applicable if a model may be required to perform in a variety of lighting settings. It’s important to consider the maximum and minimum of brightness in the room.

1. Random Noise

Adding noise to images can take a variety of forms. A common technique is “salt and pepper noise,” wherein image pixels are randomly converted to be completely black or completely white. While deliberately adding noise to an image may reduce training performance, this can be the goal if a model is overfitting on the wrong elements.

## 2.3 Identify and explain the appropriate Segmentation techniques.

Following are the primary types of image segmentation techniques:

1. Thresholding Segmentation

The simplest method for segmentation in image processing is the threshold method. It divides the pixels in an image by comparing the pixel’s intensity with a specified value (threshold). It is useful when the required object has a higher intensity than the background (unnecessary parts).

According to the different threshold values, we can classify thresholding segmentation in the following categories:

* Simple Thresholding

In this method, you replace the image’s pixels with either white or black.

* Otsu’s Binarization

In Otsu binarization, you calculate the threshold value from the image’s histogram if the image is bimodal. This process is quite popular for scanning documents, recognizing patterns, and removing unnecessary colors from a file.

* Adaptive Thresholding

1. Edge-Based Segmentation

Edge-based segmentation is one of the most popular implementations of segmentation in image processing. It focuses on identifying the edges of different objects in an image. This is a crucial step as it helps you find the features of the various objects present in the image as edges contain a lot of information you can use.

There are many edge-based segmentation methods available. We can divide them into two categories:

* Search-Based Edge Detection

Search-based edge detection methods focus on computing a measure of edge strength and look for local directional maxima of the gradient magnitude through a computed estimate of the edge’s local orientation.

* Zero-Crossing Based Edge Detection

Zero-crossing based edge detection methods look for zero crossings in a derivative expression retrieved from the image to find the edges.

1. Region-Based Segmentation

Region-based segmentation algorithms divide the image into sections with similar features. These regions are only a group of pixels and the algorithm find these groups by first locating a seed point which could be a small section or a large portion of the input image.

After finding the seed points, a region-based segmentation algorithm would either add more pixels to them or shrink them so it can merge them with other seed points.

Based on these two methods, we can classify region-based segmentation into the following categories:

* Region Growing
* Region Splitting and Merging

1. Watershed Segmentation

In image processing, a watershed is a transformation on a grayscale image. It refers to the geological watershed or a drainage divide. A watershed algorithm would handle the image as if it was a topographic map. It considers the brightness of a pixel as its height and finds the lines that run along the top of those ridges.

1. Clustering-Based Segmentation Algorithms

Clustering algorithms are unsupervised algorithms and help you in finding hidden data in the image that might not be visible to a normal vision. This hidden data includes information such as clusters, structures, shadings, etc.

As the name suggests, a clustering algorithm divides the image into clusters (disjoint groups) of pixels that have similar features. It would separate the data elements into clusters where the elements in a cluster are more similar in comparison to the elements present in other clusters.

* K-means Clustering
* Fuzzy C Means

1. Neural Networks for Segmentation

AI is used to analyze an image and identify its different components such as faces, objects, text, etc. Convolutional Neural Networks are quite popular for image segmentation because they can identify and process image data much quickly and efficiently.

# **Question No. 3**

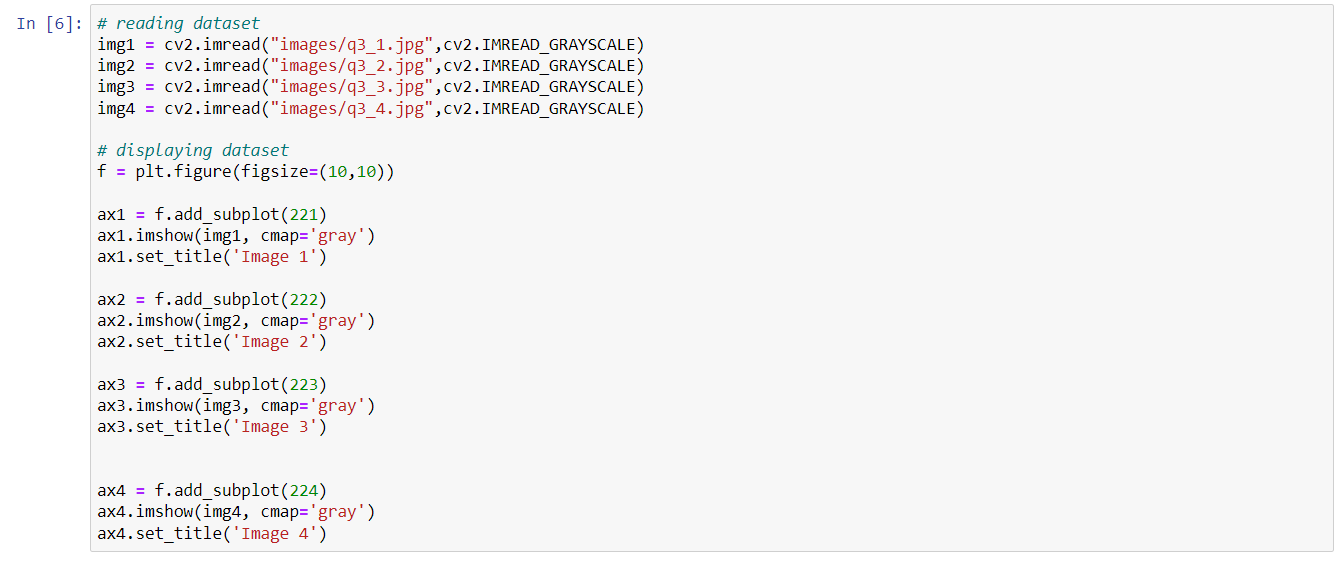
**Solution to Question No. 3:**

## 3.1 Perform pre-processing on the images of the created dataset.

Dataset created:

* Image 1: Image of two dices
* Image 2: Potrait of a lady
* Image 3: Image of a hand sign
* Image 4: Image of road sign with a background

As we can in Image 3 and 4 (Figure 7) the conrast between background and foreground is very low. The contrast can be improved by performing histogram equalization.



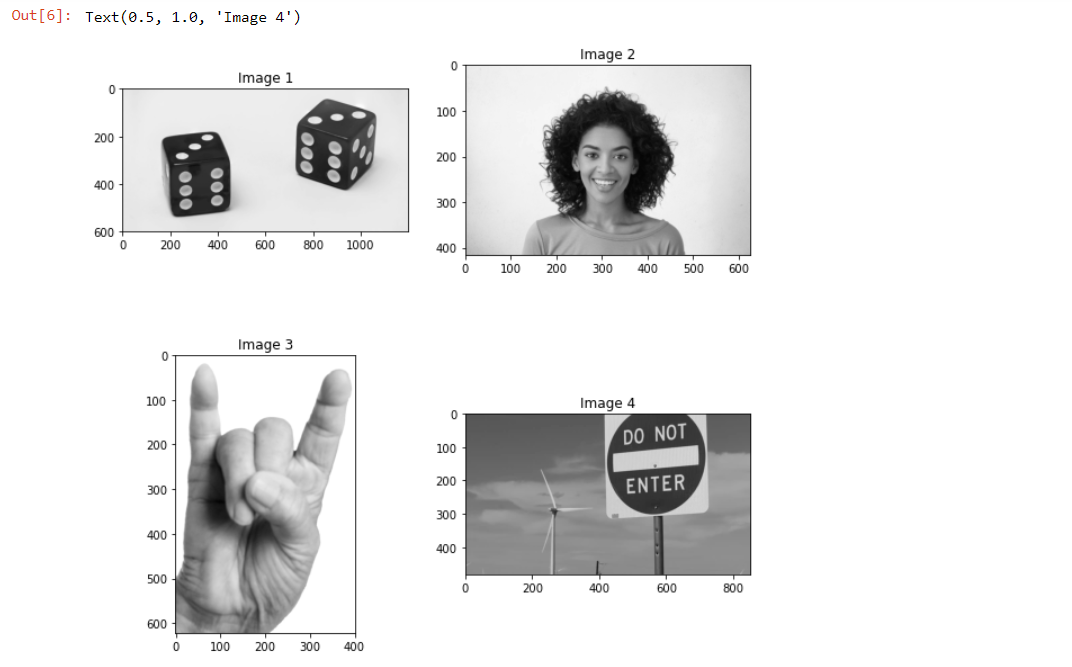
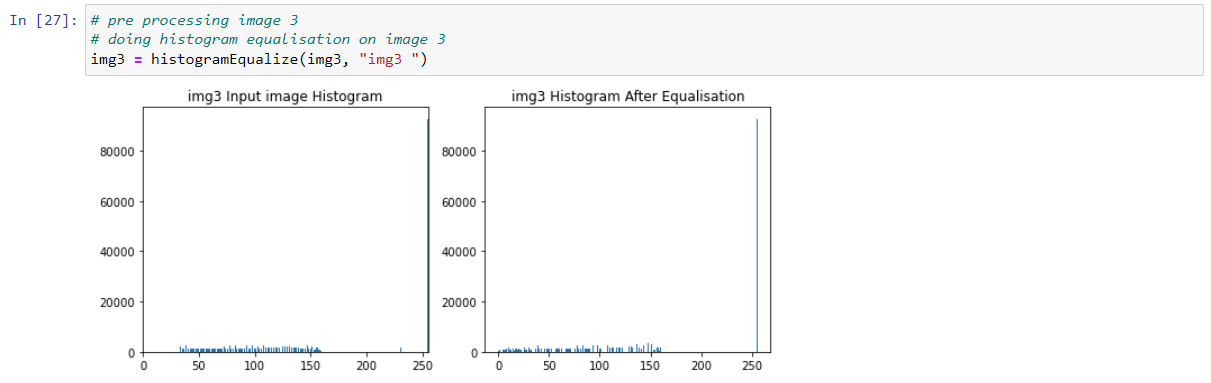
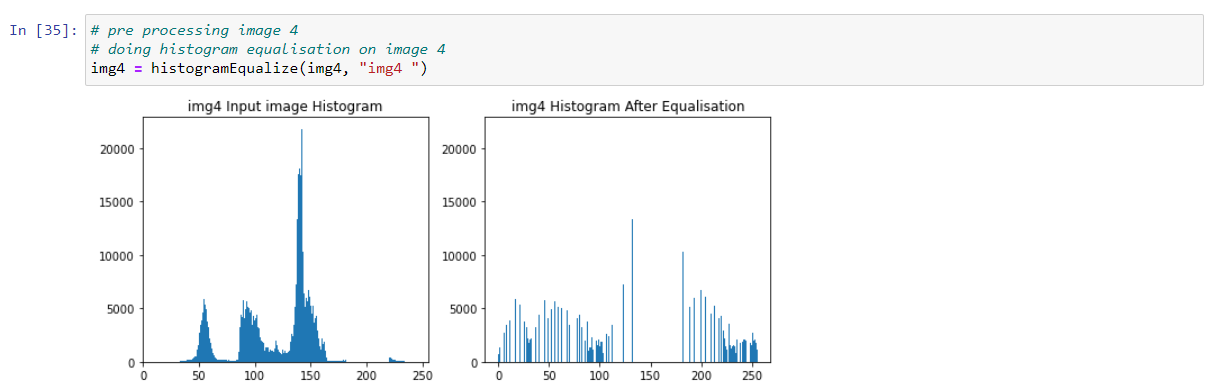


Figure Dataset images

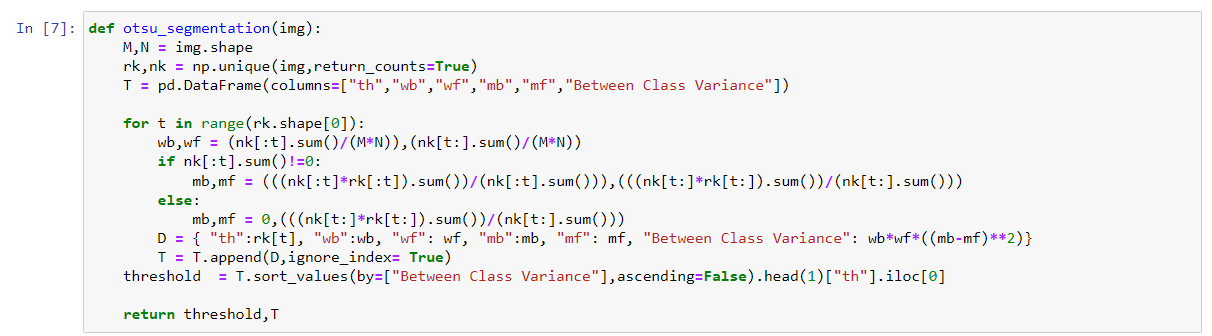


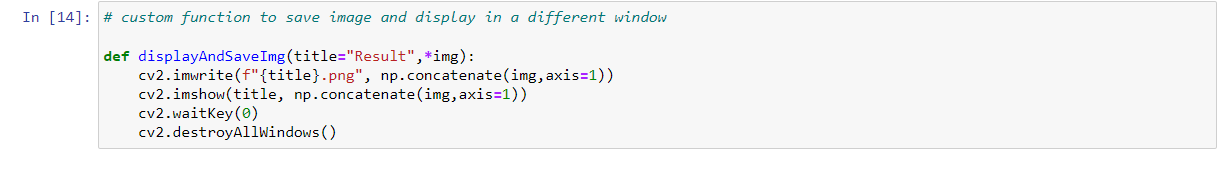


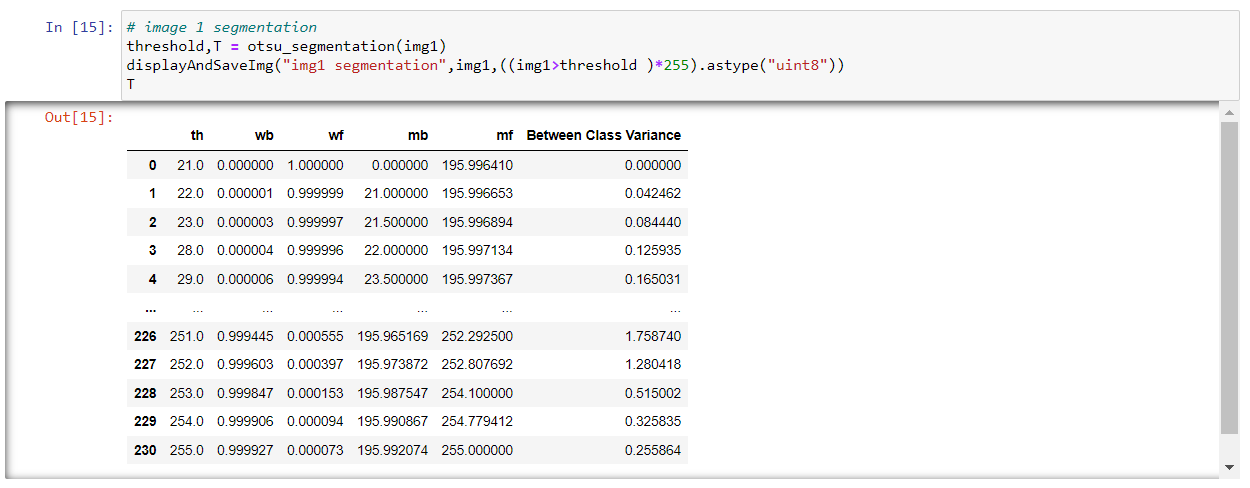
## 3.2 Perform segmentation to segment the image.

We are using thresholding segmentation, specifically Otsu Thresholding for segmentation.

**Otsu's thresholding** method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e., the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.







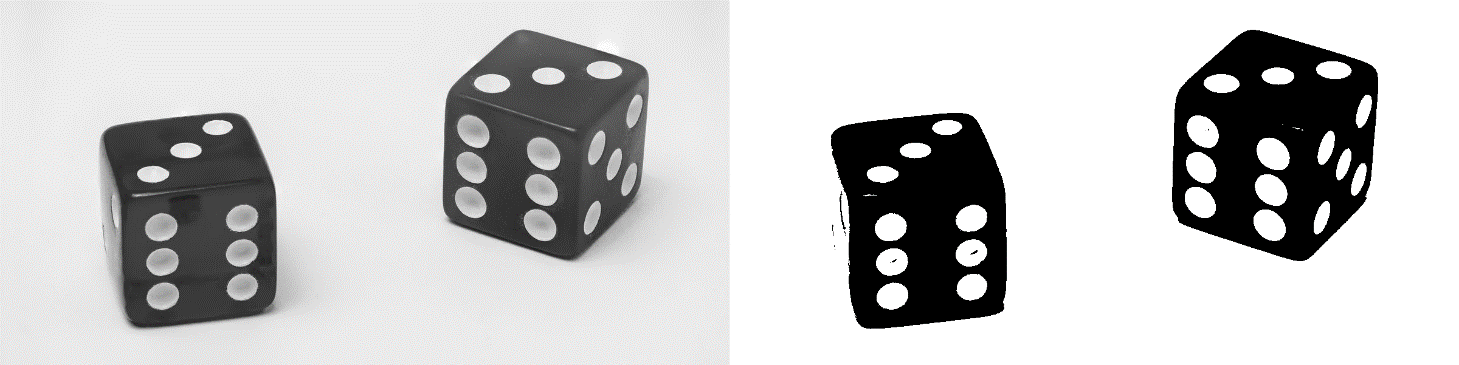


Figure Image1: Input image on the left and segmented image on the right

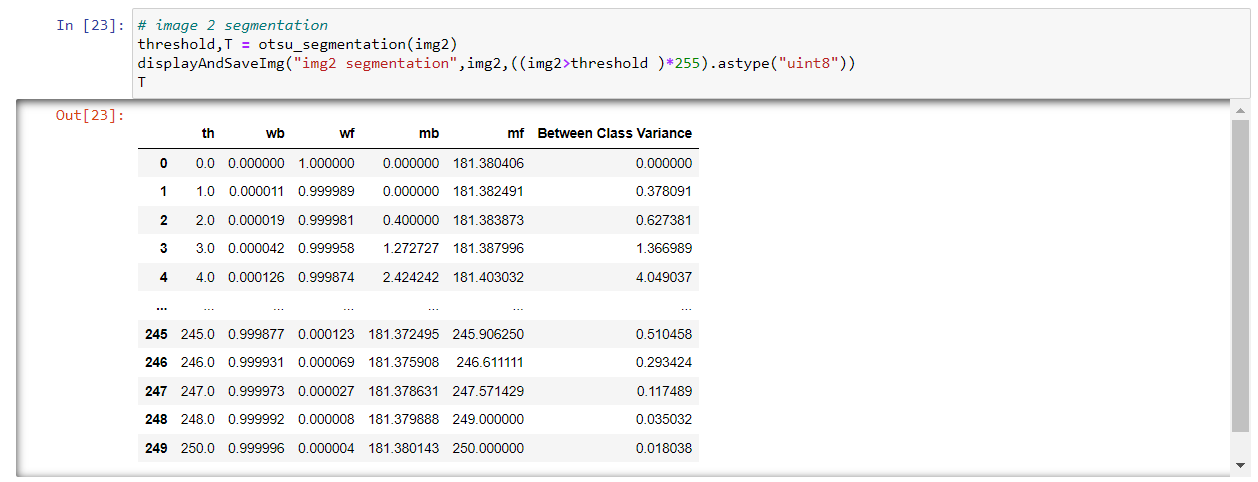




Figure Image2: Input image on the left and segmented image on the right

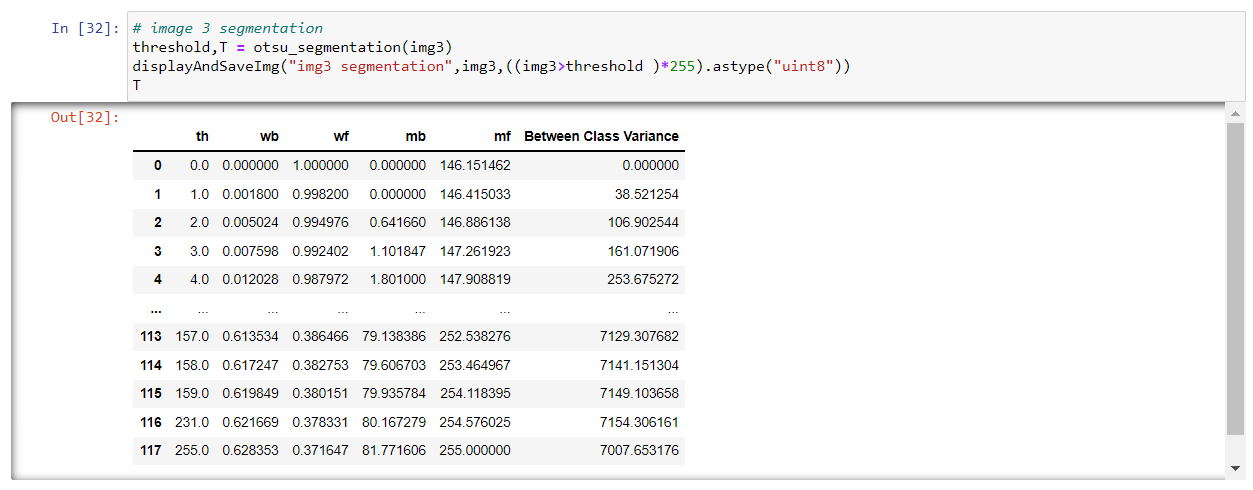
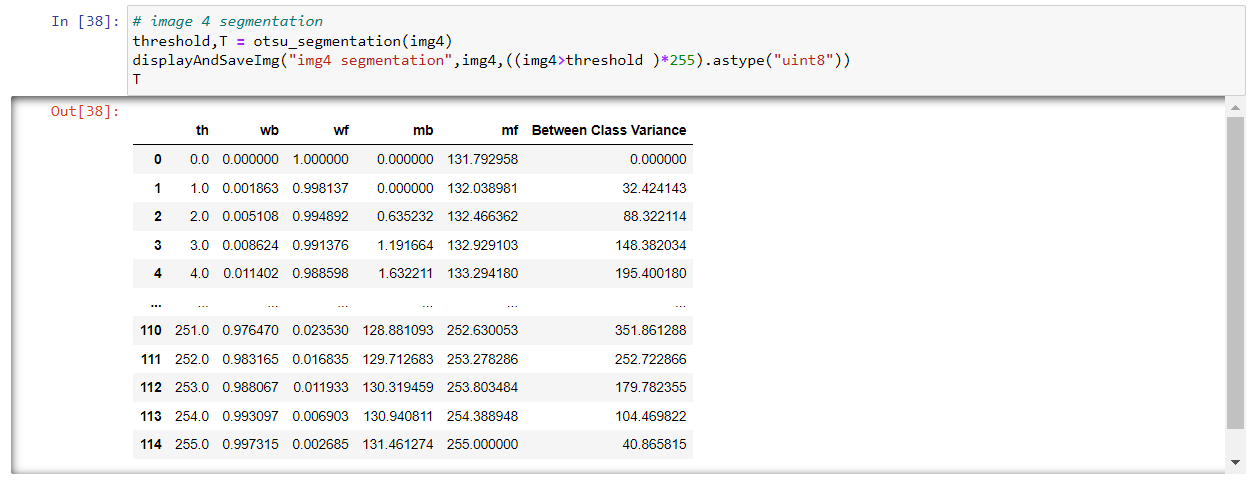




Figure Image3: Segmented image without histogram equalized



Figure Image3: Otsu Segmentation on histogram equalized image



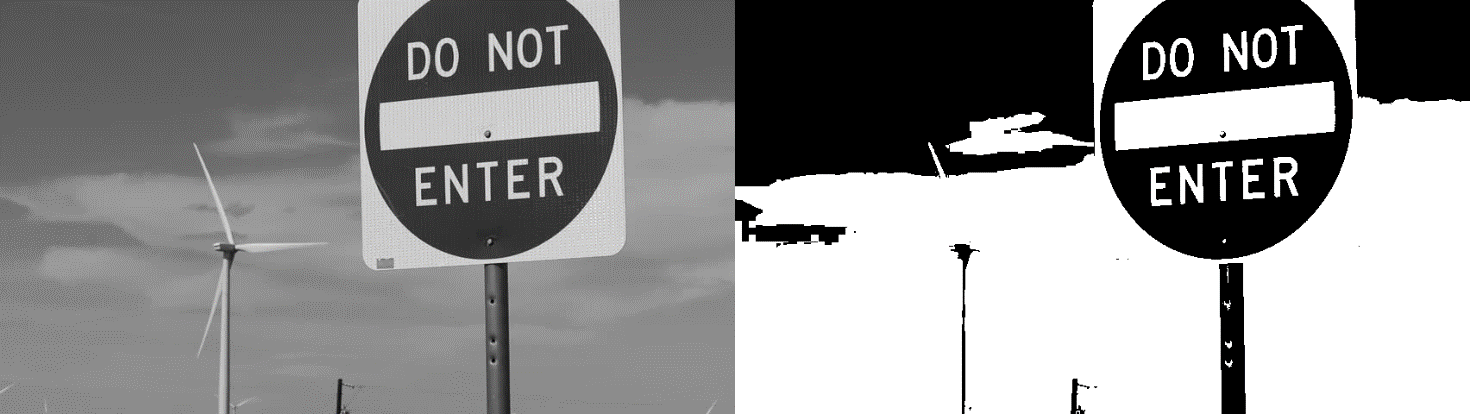


Figure Image4: Segmented image without histogram equalized



Figure Image4: Otsu Segmentation on histogram equalized image

## 3.3 Results and Discussions.

Image segmentation is the process of dividing an image (a digital image) into multiple disjoint regions, where pixels in the same region share similar properties or belong to the same object. Image segmentation changes the representation of an object, making it meaningful and easier to analyze.

We can see for image 1 (Figure 9), the image of two dices is segmented properly without any complex preprocessing. We can clearly detect the regions of the dices and its numbers shown.

For image 2 (Figure 10), the image of a portrait of a girl is segmented. Segmentation algorithm has divided the dark and light regions for easy analyzation.

For image3 (Figure 11), the image of a hand sign is segmented. We can see without any pre processing the segmentation give a bad result and difficult to recognize the hand outline/region. After we processes with histogram equalization (Figure 12) we can see a much better result in segmenting the only the hand region.

For image4 (Figure 13), the image of a road sign with background is segmented. We can see without any pre-processing the segmentation gives a moderately good result but in some places like the background clouds, it is very hard to distinguish. After we processes with histogram equalization (Figure 14) we can see more segmentation in background clouds. Overall, a much better result that before.

**Bibliography**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. <https://en.wikipedia.org/wiki/Image_segmentation>
2. <https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/>
3. <https://freecontent.manning.com/the-computer-vision-pipeline-part-3-image-preprocessing/>
4. <https://en.wikipedia.org/wiki/Histogram_equalization>
5. <https://en.wikipedia.org/wiki/Sobel_operator>