

# Image Processing Python Lab - COLOR & HISTOGRAMS

---

Here is a guide to the different parts of the lab:



- This symbol means that there is a question related to it. Your answer to the specified question should go in the yellow box below it.



- This symbol means that there is some additional or important information relating to the lab or specific exercise.



- This symbol means that the following text is part of the overarching story you follow, giving you more context into what you are doing and how it is applicable in real life situations.

## Objectives

- Implement histogram equalization
  - Determine when to use histogram equalisation
  - Determine the difference between color spaces and convert from one to the other
  - Segment parts of the image based on color
- 

## Convert to PDF

First try to convert to PDF via LaTeX:

- First you need to make sure that you have LaTeX installed (this goes for whatever operating system you are using). You can go to <https://www.latex-project.org/get/> and get the proper distribution for your operating system.
- After doing that, you can go to where xelatex.exe is found (this should be in AppData\Local\Programs\MiKTeX\miktex\bin\x64\xelatex.exe if you are getting MiKTeX for Windows, or the equivalent on other any other OS). If you are having a hard time finding the folder, you can use "where xelatex" as a command for Windows, the find command for Mac and Linux or just use the file explorer.

- After making sure you have xelatex setup, you can try the "PDF via LaTeX" in the "File" -> "Download as" menu in jupyter notebook

If the above option does not work you can follow these instructions:

- Go into "File" -> "Print Preview"
- Make sure the generated PDF is readable then you can use Ctrl+P or Command+P (on MacOS) to Print Page, where you select to save to PDF rather than printing (each system has a different interface).
- Before submitting make sure that all answers/code/plots are clearly visible.

If none of the methods work, contact a TA during labs and ask about further steps.



You are working as a CSI Image Processing expert in Delft. One night while your neighbours are away you hear a window breaking and shortly after you see a van storm off. Not long after you see your colleagues arrive at the scene and after some investigation they brief you in that there has been a robbery at the house two doors down. There seems to be some surveillance footage available from a nearby street camera but its quality is questionable at best. Lead detective O'Hara has requested you at the precinct immediately as after contacting the house owners assistant officer Reily was informed that some very valuable jewels have been stolen from the property. Begrudgingly you change into your working clothes, grab your morning coffee and head to the office.

---

After arriving at your desk you turn on your 3 working monitors and slap number 4 to make the image there stable as it has been giving you some troubles lately. In front of you is the footage from the camera in the neighbourhood. Your first task now is to "enhance" the image and make objects in it more visible as with the current quality the detectives can barely distinguish anything.

Fortunately for you your assistant Jeeves was already at the office sorting previous case files and he managed to sift through the materials before you arrived. He has selected several frames which he thinks might be useful to analyse and improve.

(Remember that you are the expert here and if you think that the frames selected by Jeeves are not suitable feel free to select your own from the video and work on them)

**i** In the folder of the Lab Notebook you are also given several folders with images to accompany the exercises you're about to start. In most of the exercises the path to the image is already specified and you only need to add the name of the file you want to work with. Additionally in some code cells you'll be given code that you can use as is. Please do not change the given functions signature (that includes the names, inputs and return values).

You can read the image(s) into the notebook using the command `plt.imread("path/to/file.jpg")`

**i** Some code cells and exercises have some code already provided for you to use. The steps for the exercises you need to do in the code cells are marked with comments. Your code should go under the corresponding comment line describing the specific step. You can treat those lines as "TODO" items.

```
In [ ]: #####
# Importing various modules into PYTHON. These will be used throughout this Jupyter Notebook
#####

# If you're having problems with the imports uncomment the following lines to install the libraries
# !pip install opencv-python
# !pip install numpy
# !pip install pickle-mixin
# !pip install matplotlib
# !pip install imutils

# import matplotlib for data visualisation
import matplotlib.pyplot as plt

# import NumPy for better matrix support
import numpy as np

# import Pickle for data serialisation
import pickle as pickle
```

```
# import cv2 and imutils for image processing functionality
import cv2
import imutils
```

```
In [ ]: # Some plotting functions to make your work easier

# Displays a given RGB image using matplotlib.pyplot
def plotImage(img, title, cmapType=None):
    # Display image
    if (cmapType):
        plt.imshow(img, cmap=cmapType, vmin=0, vmax=255)
    else:
        plt.imshow(img, vmin=0, vmax=255)
    plt.title(title)
    plt.show()

# ! Use [img] that has only one channel !
# Displays the given image and its histogram below it
def plotImageAndHistogram(img, title, cmapType=None):
    # Display image
    if (cmapType):
        plt.imshow(img, cmap=cmapType, vmin=0, vmax=255)
    else:
        plt.imshow(img, vmin=0, vmax=255)
    plt.title(title)
    plt.show()
    # Display histogram
    plt.hist(img.flatten(), bins=256)
    plt.show()
```

```
In [ ]: # Specifying the path to the images
path = "resources/lab_01/other/"

# Read and show the image that you have chosen

# Note: Reading and displaying an image using cv2 is also possible.
# However within the jupyter notebook it causes problems
```

```
In [ ]: #####
# Boilerplate code for selecting your own frame from the video
#####
```

```
cap = cv2.VideoCapture(path + "dutch_plates_cropped.mp4")

# Choose a frame to work on
frameN = 42

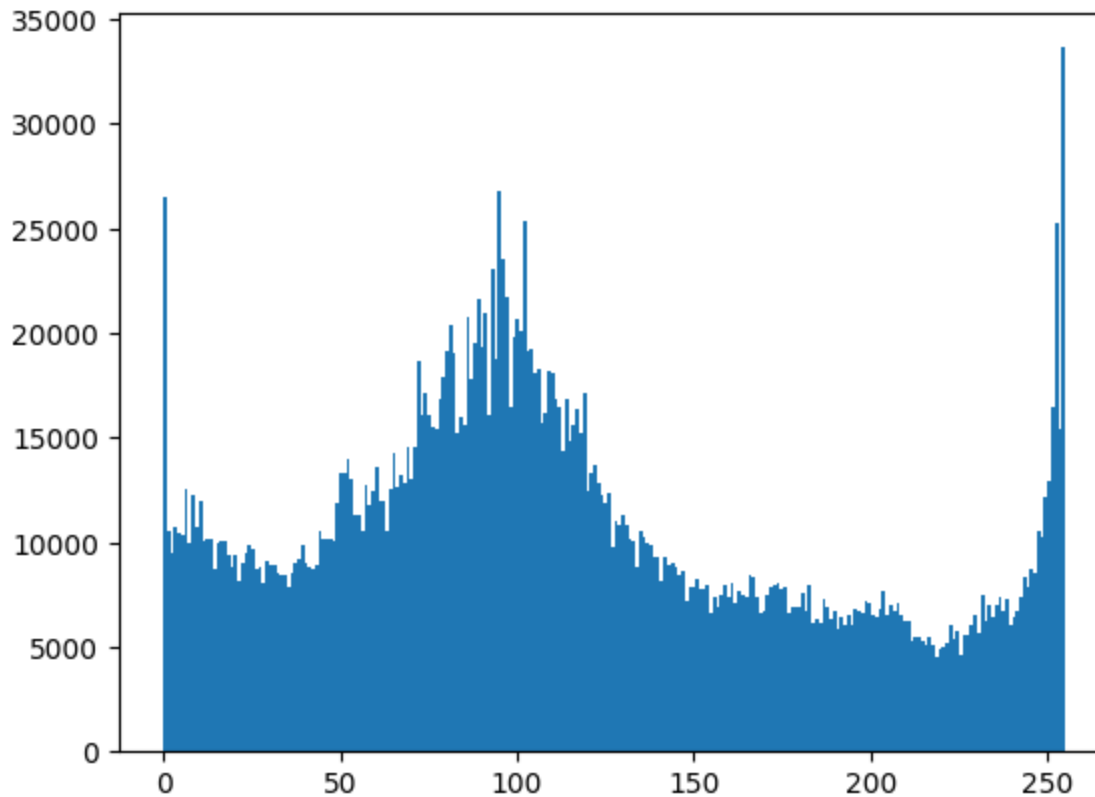
for i in range(0, frameN) :

    # Read the video frame by frame
    ret, frame = cap.read()
    # if we have no more frames end the loop
    if not ret:
        break

# When everything done, release the capture
cap.release()
# cv2.destroyAllWindows()

# Show your frame
plotImageAndHistogram(frame, "Selected")
```





## PYTHON HANDS-ON Assignment I.1: Enhance the contrast of the image

### Completion requirements for this assignment:

- ☐ Implement the 4 ways of contrast improvement
- ☐ Show and explain the pros and cons of histogram equalization

The image you're given of the robbery seems like it could use some contrast enhancements. You cannot clearly see how many people are being taped or what kind of car is on the tape. To fix this you're tasked to try multiple different contrast enhancement techniques to determine which works best in making the image more readable. Work with the 'robbery\_low\_contrast.jpg' in 'resources/lab\_01/exercise\_01' directory.

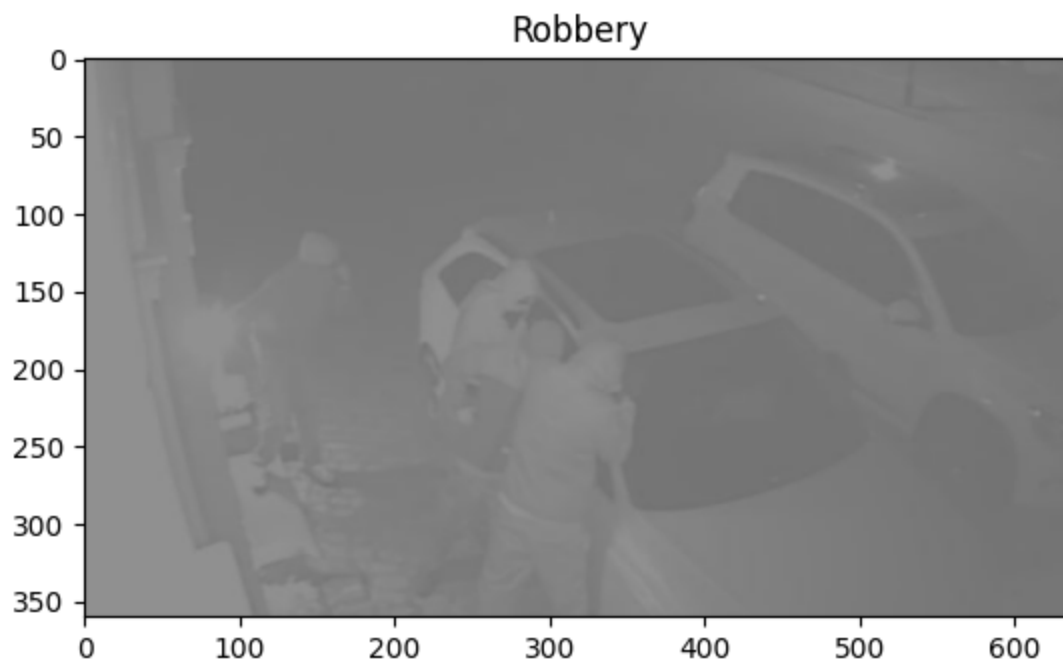
For this first part you'll need to display a chosen image and its histogram

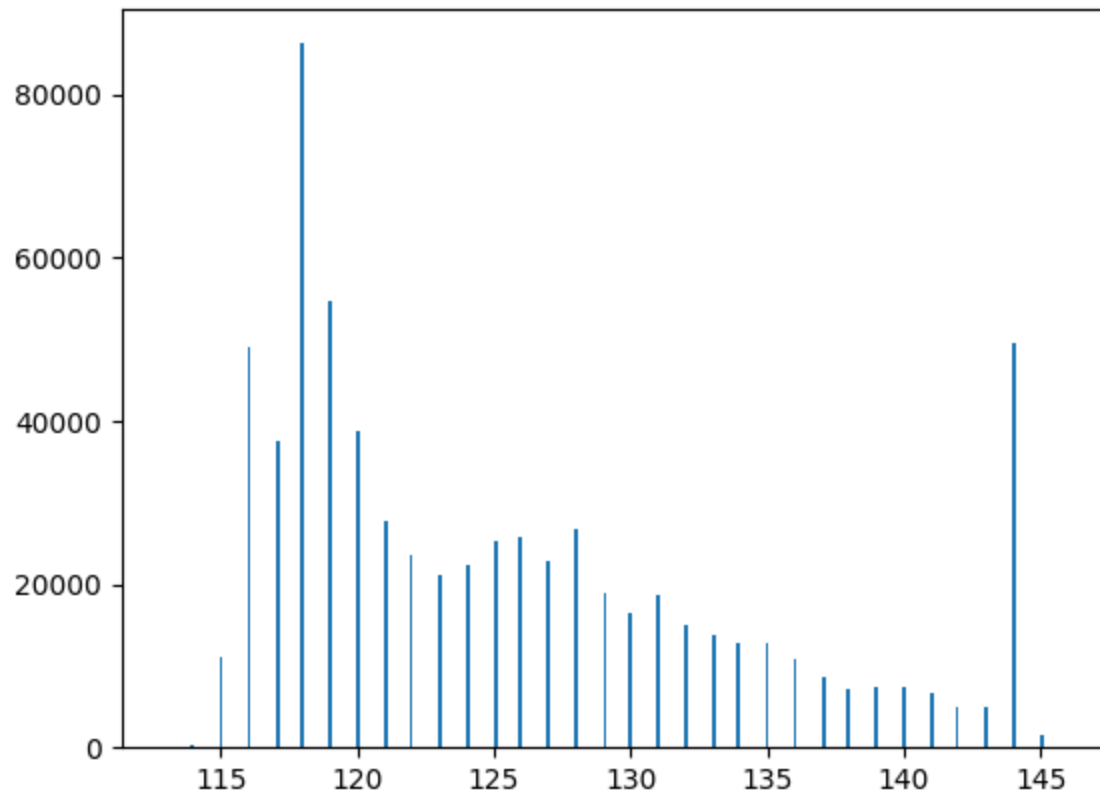
**i** You can use read the input image also using `cv2.imread('myimage.jpg')` - Notice that OpenCV reads the images as **BRG instead of RGB**

**i** Make sure that images are always copied to `np.array` variable type before doing any mathematical operations on the images.

```
In [ ]: # Load image
path = "resources/lab_01/exercise_01/"

# Display image and histogram
image = cv2.imread(path + "robbery_low_contrast.jpg")
plotImageAndHistogram(image, "Robbery")
```





Out[ ]: (360, 640, 3)

## 1.1.1 Set the global variables Contrast Improvement - Pixel Multiplication

Apply the point operation of contrast enhancement by multiplying pixel values by a certain contrast factor. Since we're aiming to increase the contrast think about if the contrast factor should be  $>1$ ,  $<1$  or  $=1$ .

```
In [ ]: # Define a function to do the point operation
def contrastImprovementPixelMultiplication(img, contrastFactor, title = None):
    # Define point operation formula

    # Do point operation on pixels
    image = np.clip(img * contrastFactor, 0, 255).astype(np.uint8)
    plotImageAndHistogram(image, title, 'gray')
    return image
```



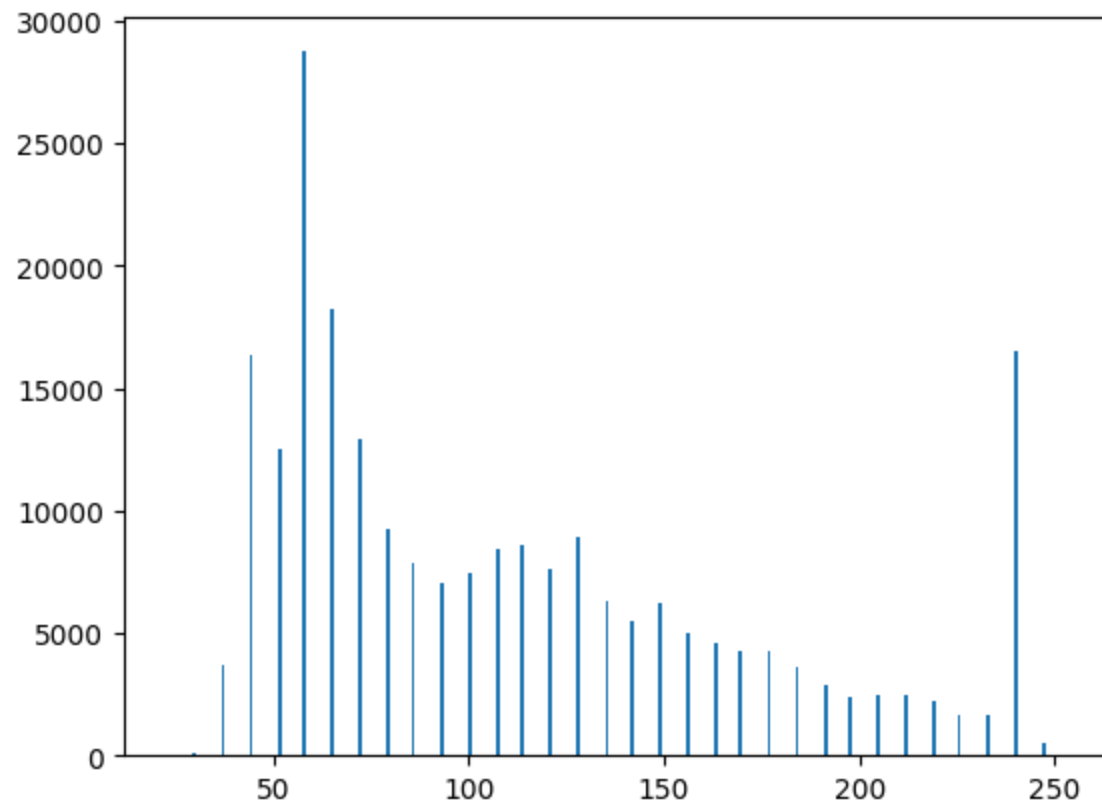
```
# Load image
path = "resources/lab_01/exercise_01/"

# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

mean = np.mean(gray)
# Change contrast using point operation
cont_gray = contrastImprovementPixelMultiplication(gray, 7, "Contrasted")
# Plot the image and histogram

#1 doesnt change the img
# smaller than 1 will make it more
```





## I.1.2 Contrast Improvement - Formula

Apply contrast improvement using this formula,  $(I(x,y) - 127) * \text{contrast} + 127$



*Remember to work with integers when applying the formula*

```
In [ ]: # Define a function to do the point operation
def contrastImprovementFormula(img, contrastFactor, title = None):
    # Define point operation formula

    # Do point operation on pixels
    image2 = ((img - 127) * contrastFactor) + 127
    plotImageAndHistogram(image2, title, 'gray')
```

```
return image2

# Load image
path = "resources/lab_01/exercise_01/"

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

mean = np.mean(gray)

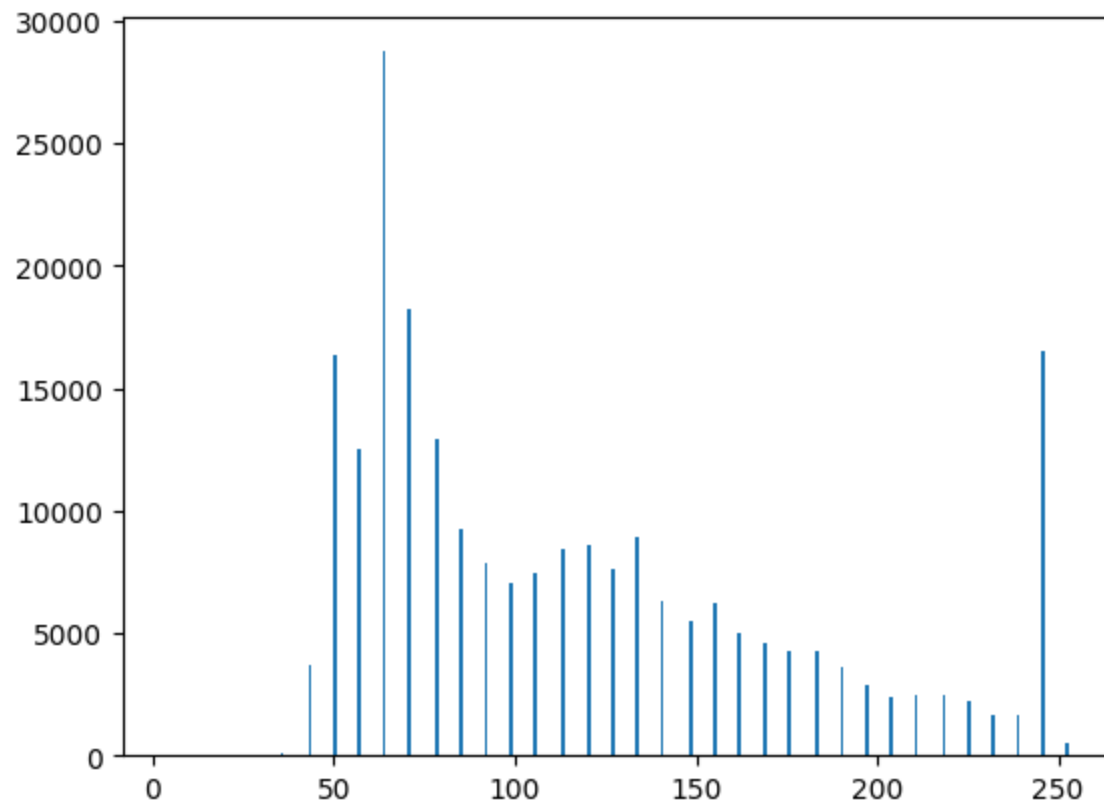
cont_gray = contrastImprovementFormula(gray, 1, "Contrasted")

# Convert to grayscale

# Change contrast using formula

# Plot the image and histogram
```





### 1.1.3 Contrast Improvement - Contrast Stretching

Apply contrast stretching in practice using the linear transform function  $f(i) = a(i + b)$ . Make sure to find appropriate values to  $a$  and  $b$ .

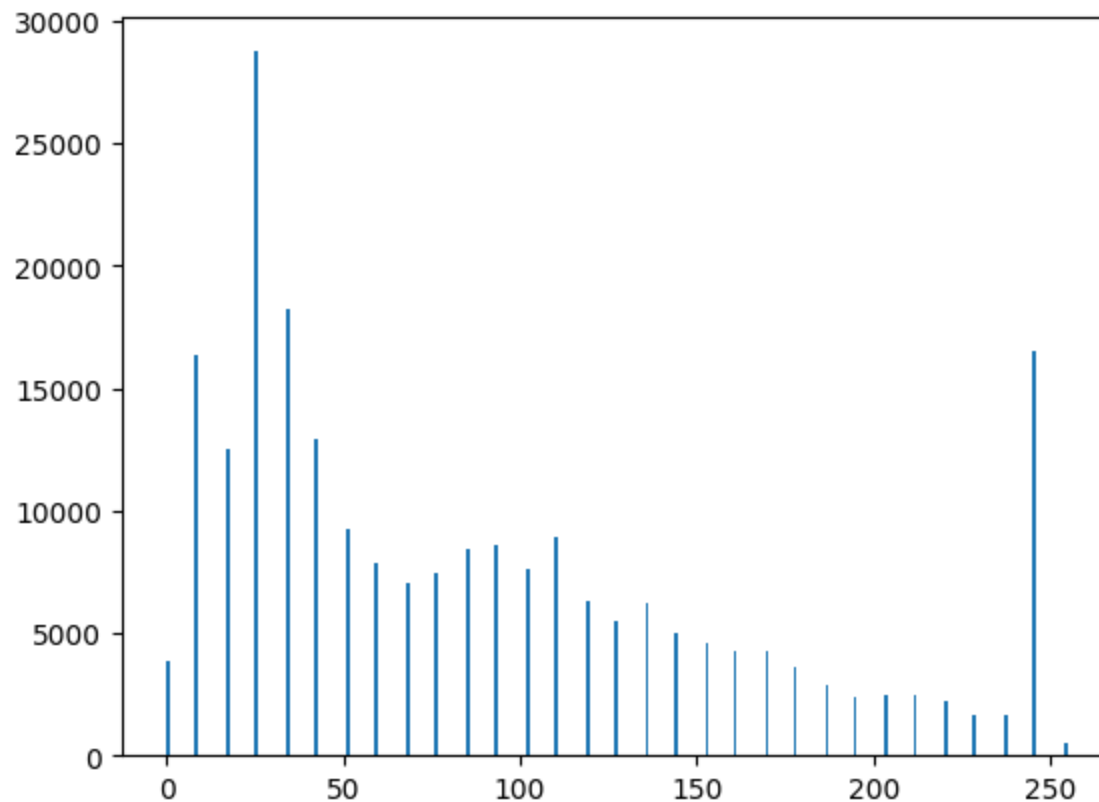
```
In [ ]: def contrastImprovementContrastStretching(img, a, b, title = None):
    # Define operation formula
    img = a * (img + b)
    # Do point operation on pixels
    img = np.clip(img, 0, 255).astype(np.uint8)
    return img

# Load image
path = "resources/lab_01/exercise_01/"
```

```
# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Apply contrast stretching function
a = 255/30
b = -115
image2 = contrastImprovementContrastStretching(gray, a, b, "gray")
plotImageAndHistogram(image2, "image", "gray")

# Plot the image and histogram
```





### I.1.4 Contrast Improvement - Histogram Equalization

Now you'll need to apply histogram equalization. Parts of the code have already been provided as well as some function suggestions. Feel free to refer to the numpy documentation (<https://numpy.org/doc/>) for more clarification on what each function does.

```
In [ ]: def contrastImprovementHistogramEqualization(img, title = None):  
        # Create your own histogram array (a suggested function is np.bincount())  
  
        hist,bins = np.histogram(img.flatten(),256,[0,255])  
  
        hist = hist / len(img.flatten())  
        # Convert to cumulative sum histogram
```

```
cumulativeHistogramArray = hist.cumsum()

# Creating a mapping lookup table
transformMap = np.floor(255 * cumulativeHistogramArray).astype(np.uint8)

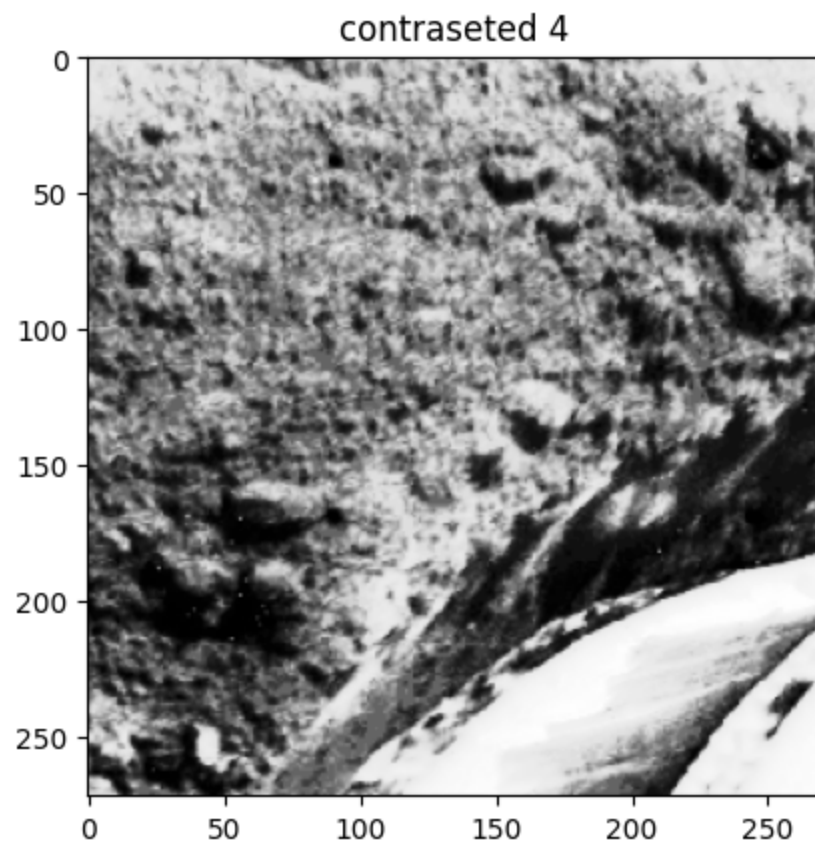
# Flatten image into 1D List
imgflat = img.flatten()
# Transform pixel values to equalize
imgflat = [transformMap[x] for x in imgflat]
# Write back into an image
img = np.reshape(imgflat, img.shape)
return img

# Load image
photo = cv2.imread(path + "slidePic.png")

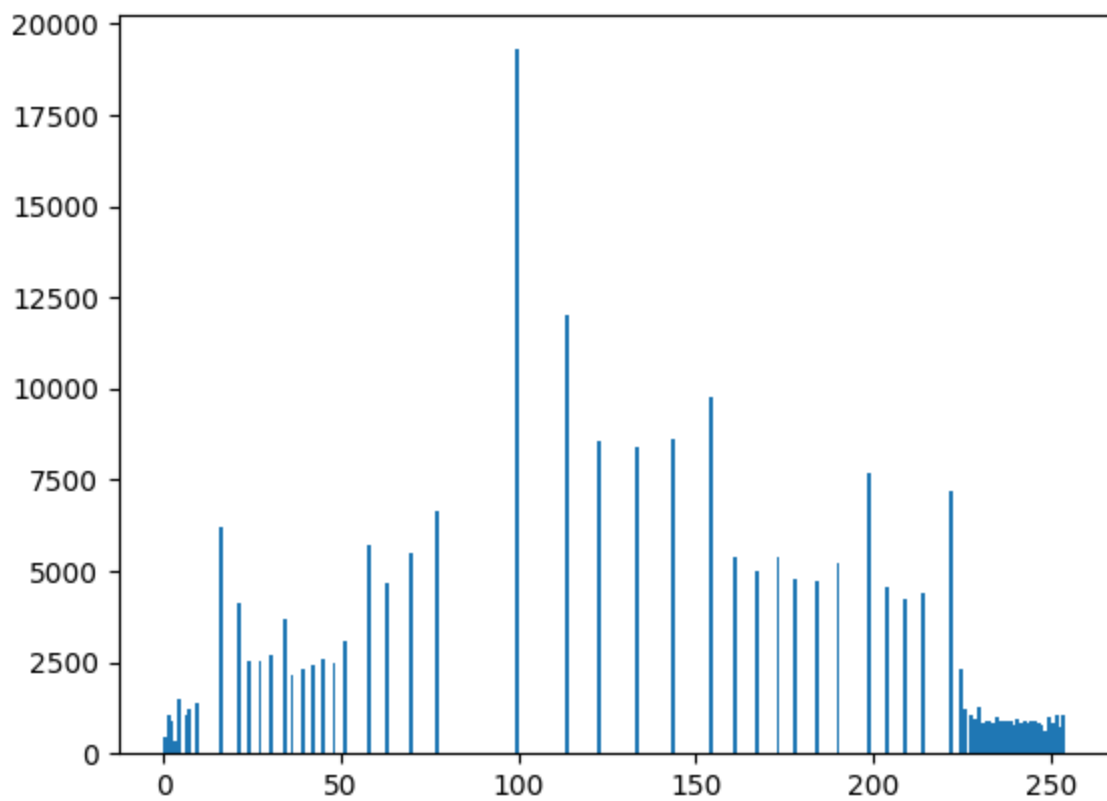
path = "resources/lab_01/exercise_01/"
i = contrastImprovementHistogramEqualization(photo)
# Plot original image and histogram
plotImageAndHistogram(i, "contraseted 4", "gray")
# Apply histogram equalization function

path = "resources/lab_01/exercise_01/"
i = contrastImprovementHistogramEqualization(image)
# Plot original image and histogram
plotImageAndHistogram(i, "contraseted 4", "gray")

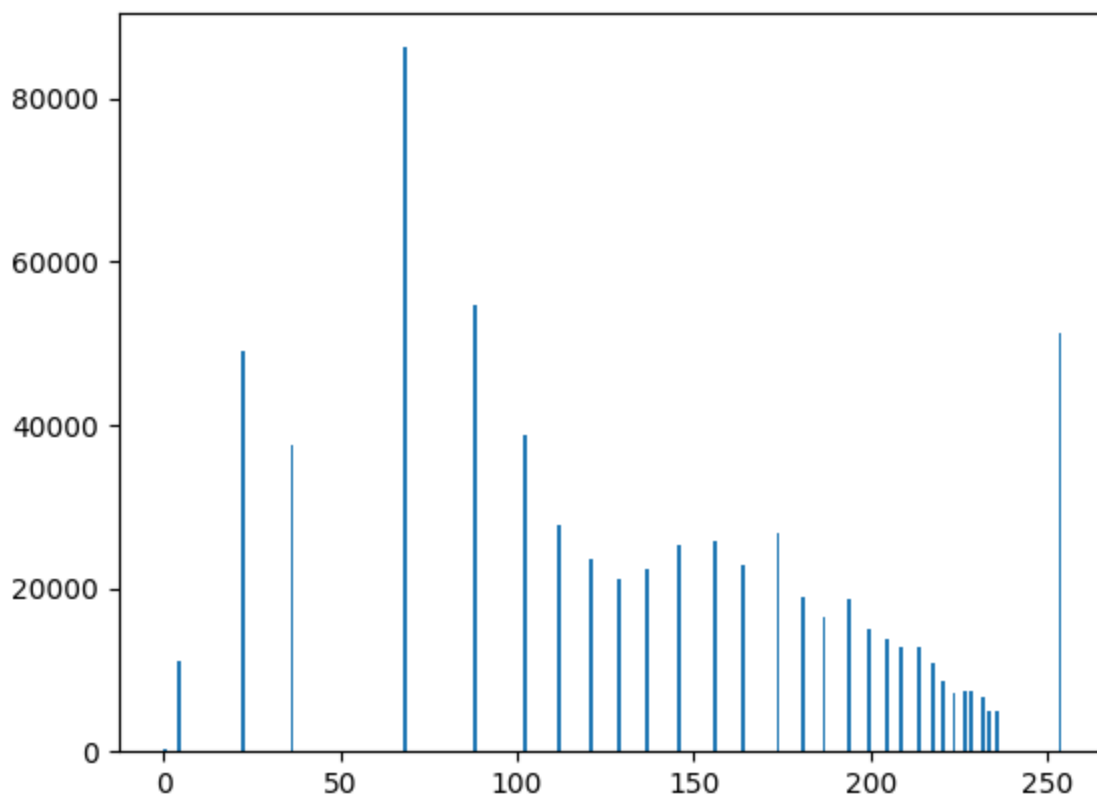
# Plot the image and histogram
```











## Questions section

? (type your answer in the yellow field below) *Showcase the pros and cons of histogram equalisation:*

- *Provide 1 example images where it would be a preferred technique and explain why.*
- *Provide 1 image where you would use a different contrast manipulation method and explain why histogram equalisation is worse*

Type your answer here.

1. frist image is an example where it works well. The reason it works well is because we have a lot of color set together and we add diffence and contrast between smaller changes.

2. Second image works less well because we have more variety in the color, the distribution is more spread out, meaning that we don't add much new information when equalizing.

## PYTHON HANDS-ON Assignment I.2: Color analysis and manipulation



A college police officer, Barry Steel, while cycling noticed a car that roughly matches the description given of the one from the robbery. Unfortunately the car passed him by quite quickly and he couldn't get a good look at it. Fortunately, however, all police bikes are equipped with a camera and so officer Steel decided to come to you with the footage and images hoping you'd be able to figure something out.

The police station has an old image processing program that takes in the images of cropped plates and tries to find matches in the database. They used to do this task of cropping manually. But since you joined the team, they asked you if you could build a system that does the cropping automatically.

### Completion requirements for this assignment:

- ☐ Convert images into different color spaces
- ☐ Segment a part of an image based on its color/hue
- ☐ Crop out the licence plate based on the previous segmentation

One possible way to do this is using color segmentation since the Dutch plates usually have a distinct yellow color. To build such a system you know you'll need the following steps:

- View the image in different color space to see how the colors are distributed over the channels & to determine which color space is the more feasible to work with.
- View the histogram of each color space channel to instigate how the colors are distributed over each channel
- Segment the plates in the image based on the color.

- Crop the image after segmentation by determining the corner location of the plate.

## I.2.1 Converting to different color spaces and channels


Convert an image to different color spaces and work with different color channels. Convert the image into the following color spaces:


- RGB
- HSI
- Grayscale


and determine the yellow color in each space (if possible) by visually inspecting the histograms of the channels:


- Hue
- Saturation
- Intensity

Finally, write the most appropriate color range you determined (and specify in which color space it is)

 You can read about some mathematical features of colors in [this book](https://staff.fnwi.uva.nl/th.gevers/pub/CIP06.pdf)(<https://staff.fnwi.uva.nl/th.gevers/pub/CIP06.pdf>) on page 5

 You can use read the input image also using `cv2.imread('myimage.jpg')` - Notice that OpenCV reads the images as **BRG instead of RGB**

 To Convert the BRG image to RGB, this function can be used `cv2.cvtColor(img, cv2.COLOR_BGR2RGB)`

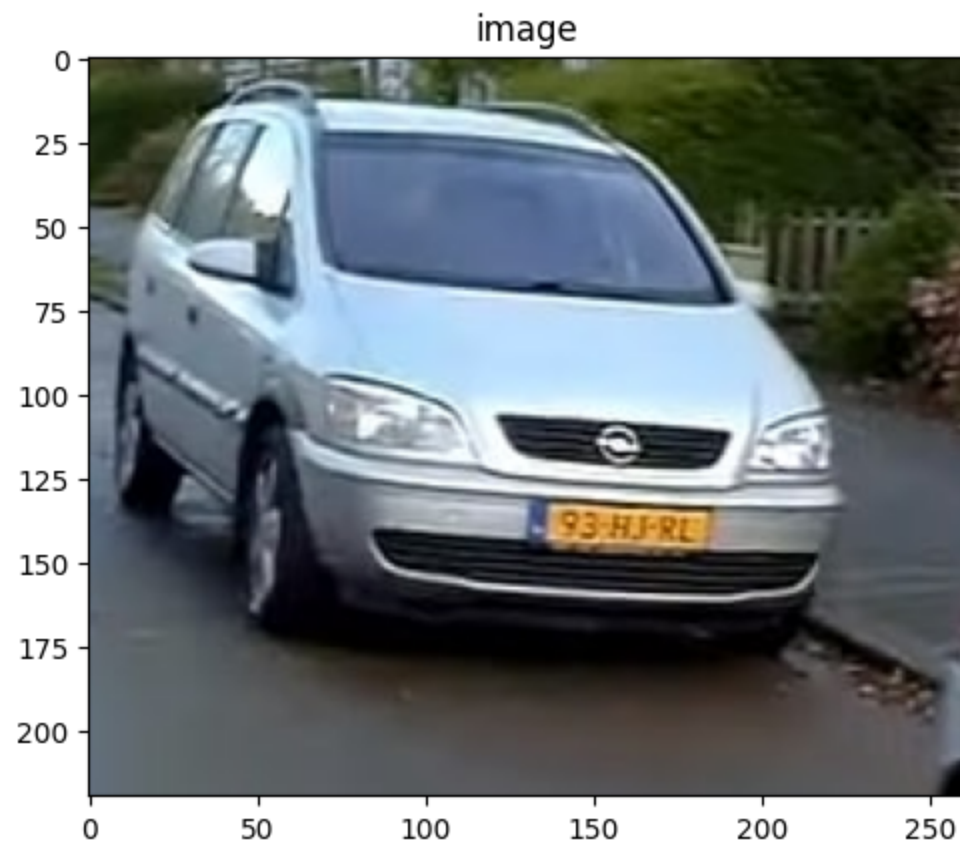
 To Convert the RGB image to HSV, this function can be used `cv2.cvtColor(img, cv2.COLOR_RGB2HSV)`

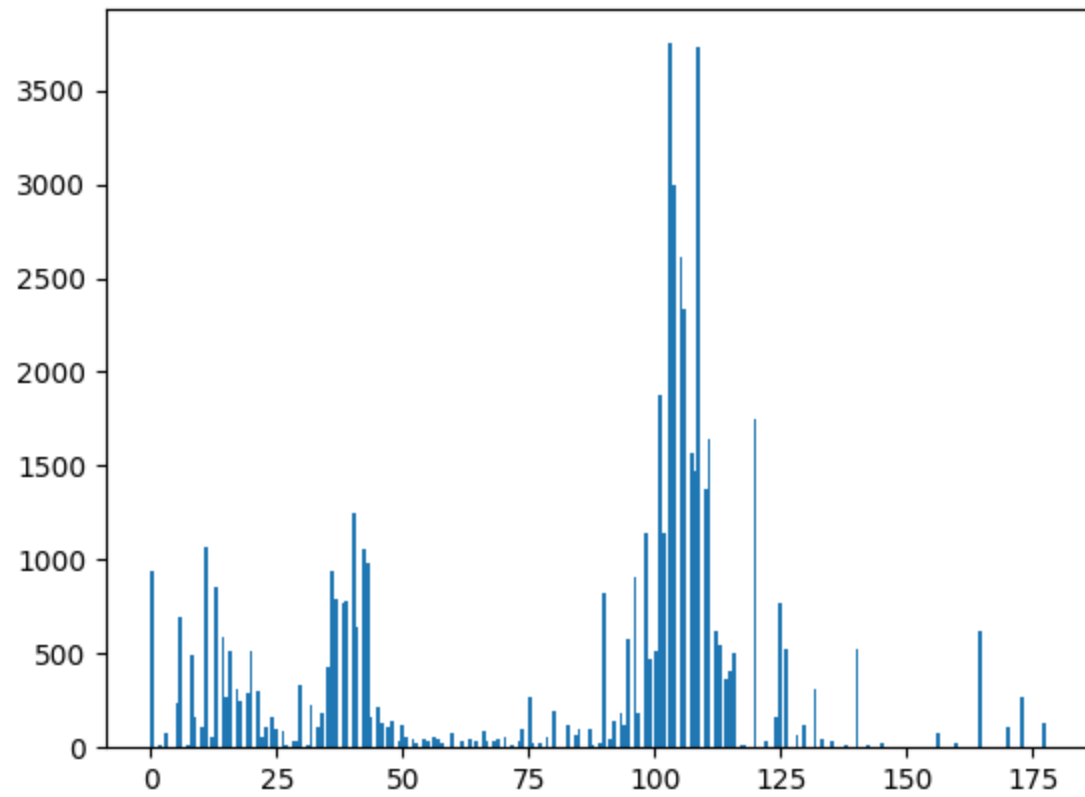
Make sure to set up `matplotlib.pyplot` if you're displaying an image that isn't in RGB

```
In [ ]: # Load image
path = "resources/lab_01/exercise_02/"
image = cv2.imread(path + "single_car_plate.jpg")
# Display image
# Convert to RGB (if it isn't)
imageRGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plotImage(imageRGB, "image")
# Convert to HSV/HSI
imageHSV = cv2.cvtColor(imageRGB, cv2.COLOR_RGB2HSV)

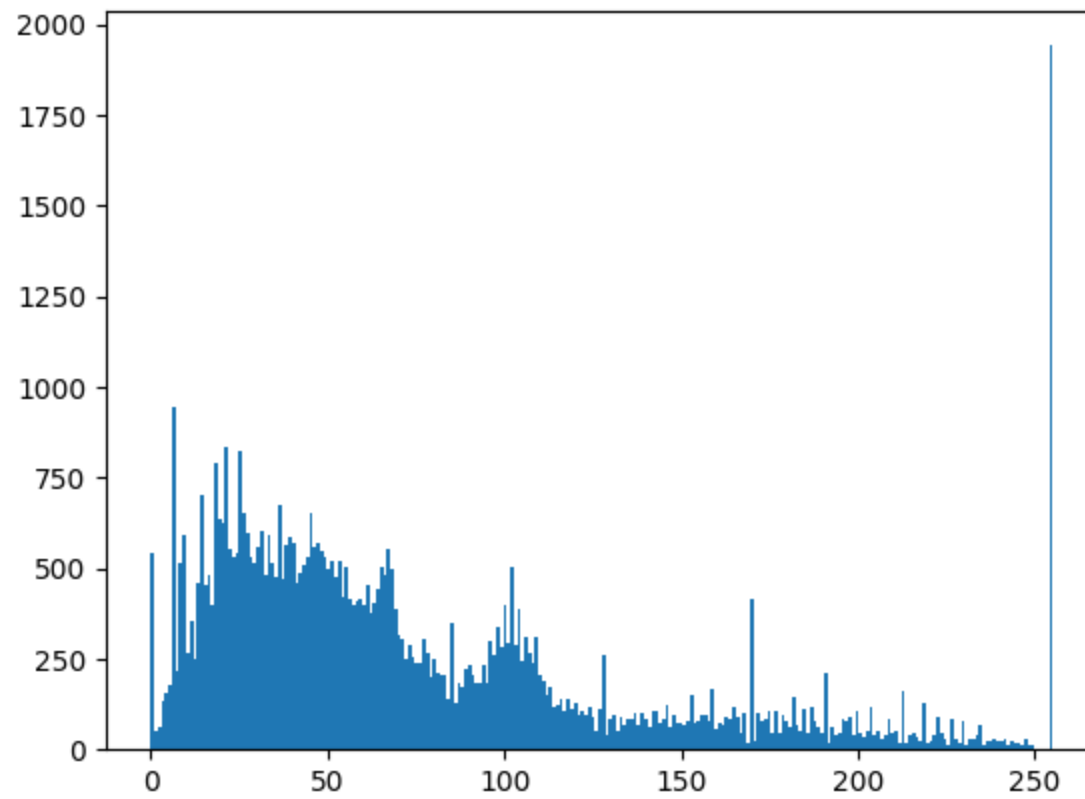
# Convert to grayscale
imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Isolate Hue channel and plot its histogram
plt.hist(imageHSV[:, :, 0].flatten(), bins=256)
plt.show()
# Isolate Saturation channel and plot its histogram
plt.hist(imageHSV[:, :, 1].flatten(), bins=256)
plt.show()

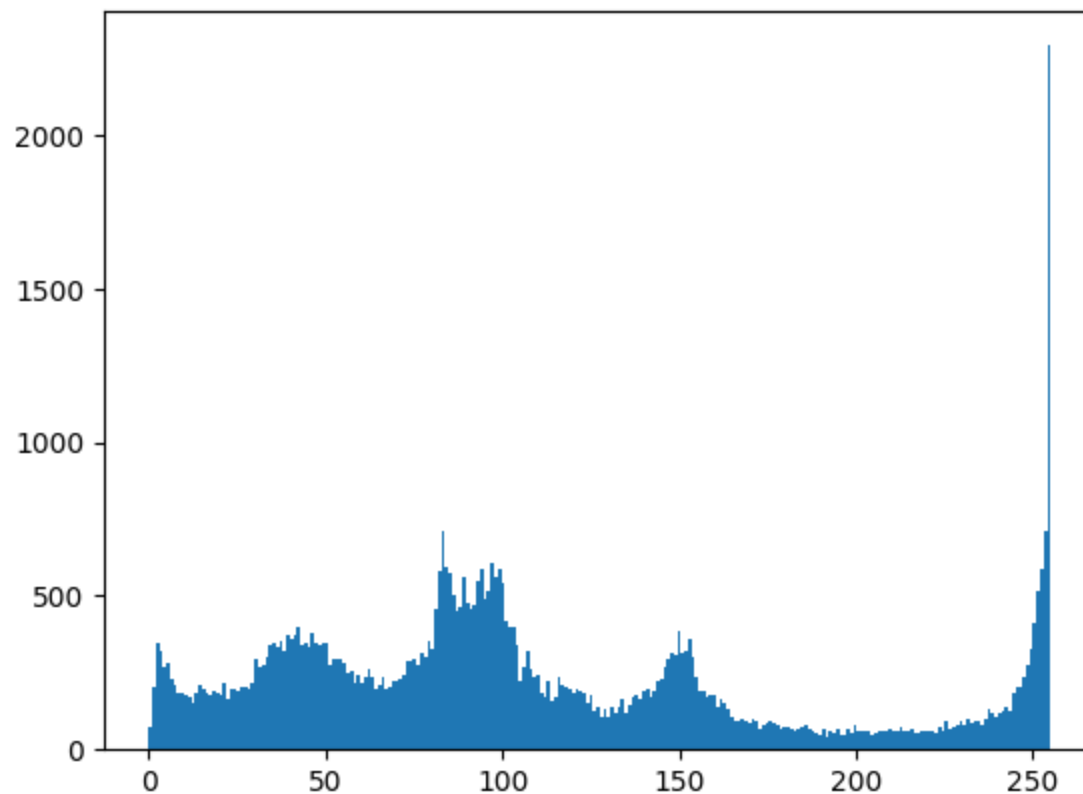
# Isolate Intensity channel and plot its histogram
plt.hist(imageHSV[:, :, 2].flatten(), bins=256)
plt.show()
```











## I.2.2 Segmenting based on color

In your line of work you know that the HSI space is very well suited to do the segmentation, specifically the hue channel. You can use `cv2.inRange()` to do the color extraction. This function acts like a band-pass filter in the way that it only keeps the pixels in the image with color values falling outside the chosen range to zeros. Try different ranges to best segment the plate.

```
In [ ]: # Load image
        path = "resources/lab_01/exercise_02/"

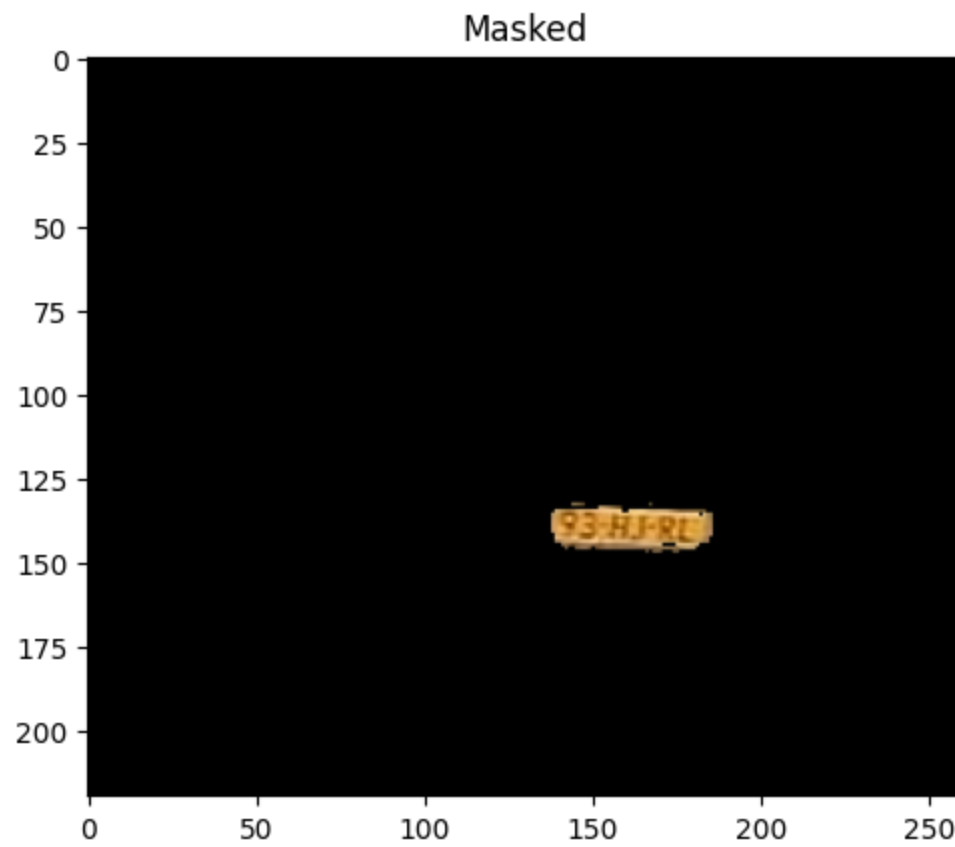
        # Convert to HSI/HSV
        imageHSV = cv2.cvtColor(imageRGB, cv2.COLOR_RGB2HSV)

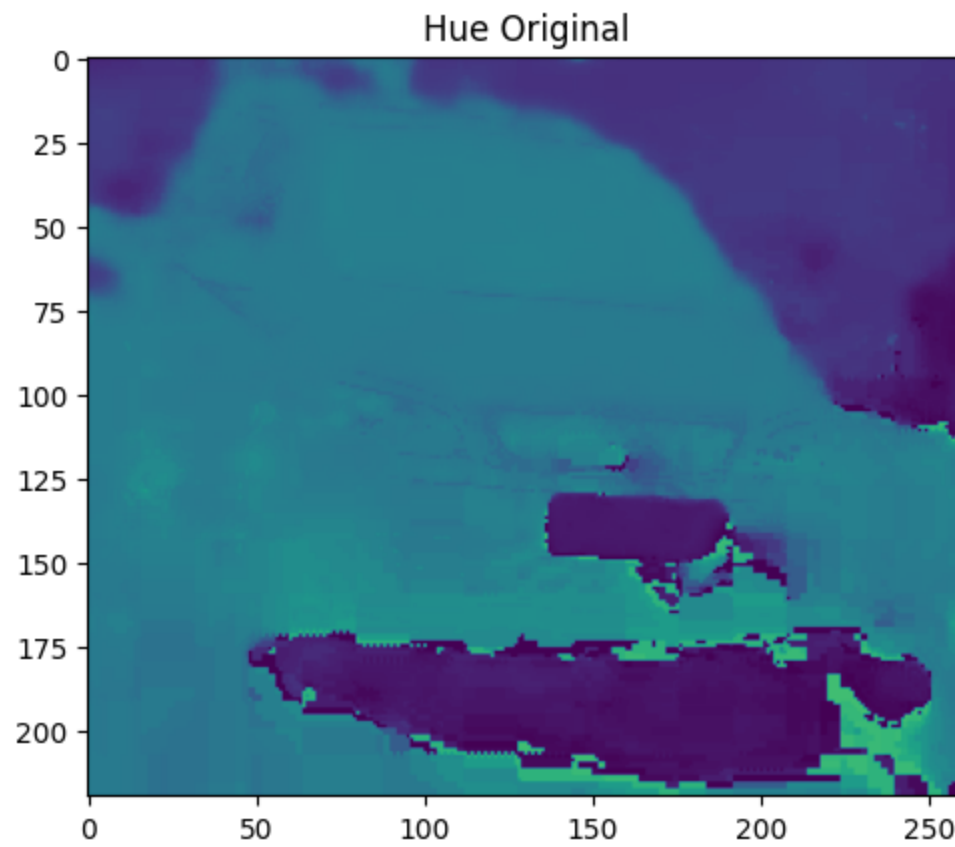
        # Define color range
```

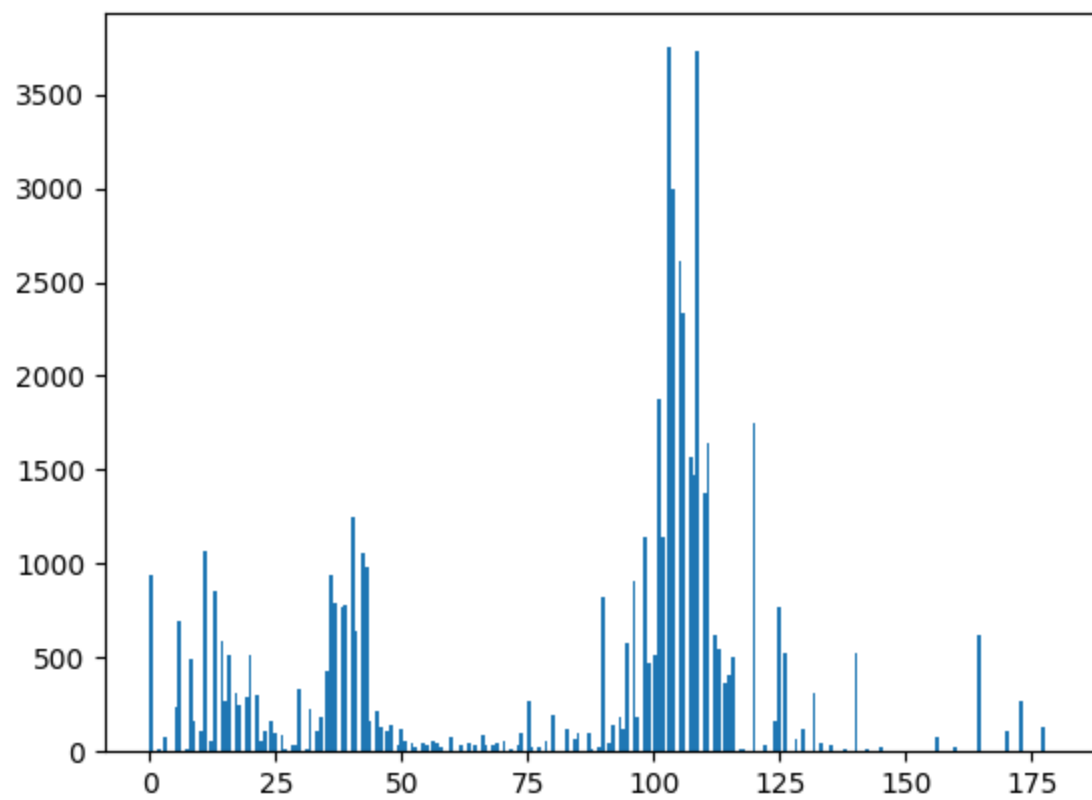
```
colorMin = np.array([10,100,100])
colorMax = np.array([30,255,255])

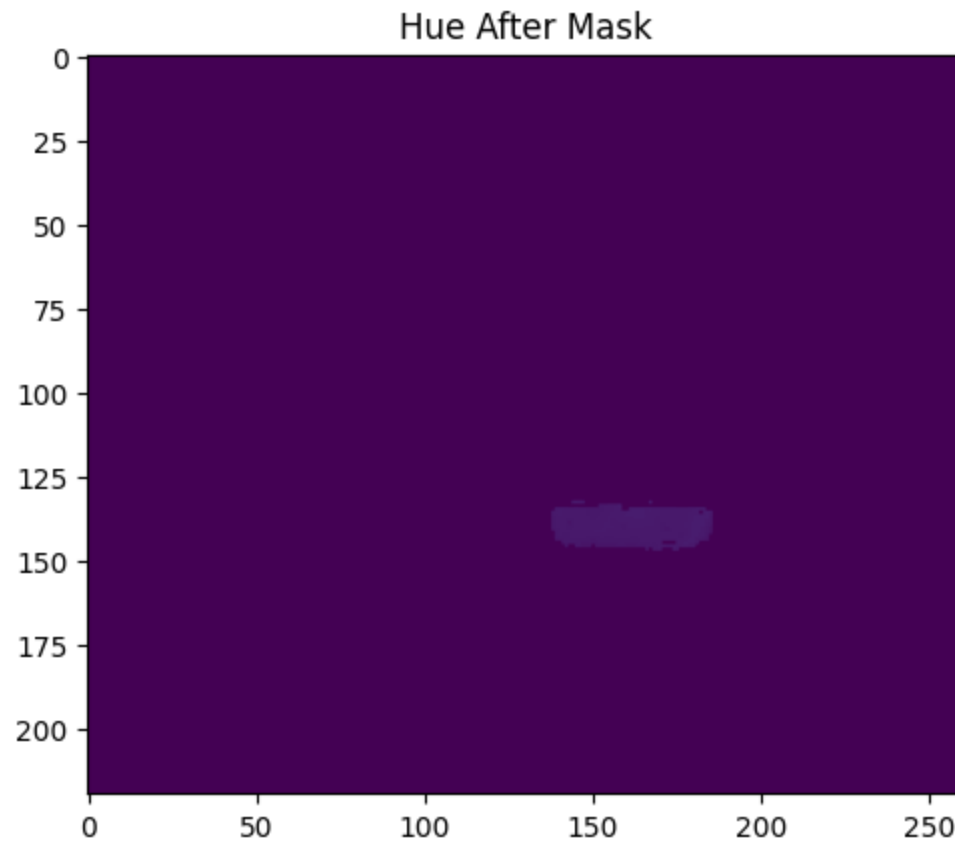
# Segment only the selected color from the image and leave out all the rest (apply a mask)
mask = cv2.inRange(imageHSV, colorMin, colorMax)
masked = cv2.bitwise_and(imageRGB, imageRGB, mask=mask)
plotImage(masked, "Masked")
# Plot the masked image (where only the selected color is visible)

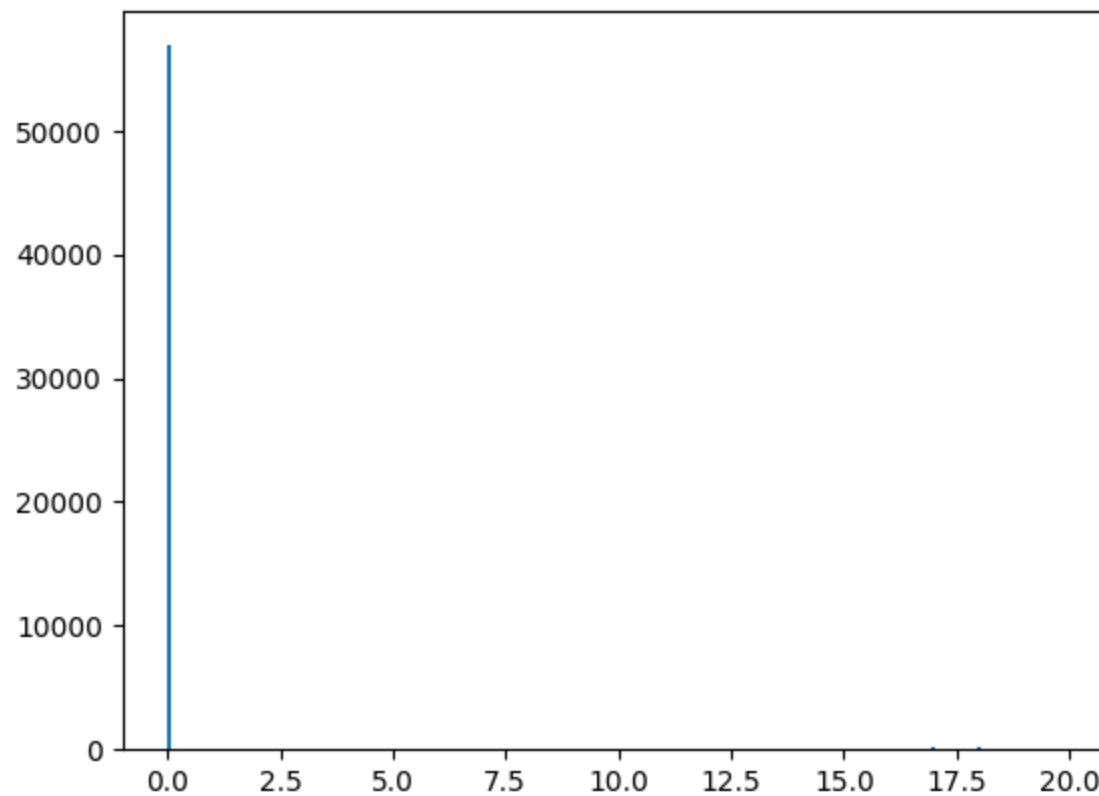
# Plot the original hue channel and its histogram
originalHSVHue = imageHSV[:, :, 0]
plotImageAndHistogram(originalHSVHue, "Hue Original")
# Plot the masked hue channel and its histogram (the hue channel after applying the mask)
targetHSV = cv2.cvtColor(masked, cv2.COLOR_RGB2HSV)
targetHSVHue = targetHSV[:, :, 0]
plotImageAndHistogram(targetHSVHue, "Hue After Mask")
```











### I.2.3 Cropping the licence plate

Now that you have extracted the plate location based on its color it is time to crop out the licence plate itself.

The masking process converted all pixels that were outside of the color range, but the size of the image is still the same. Thus, you now need to find the maximal(and the minimal) x and y coordinates that contain useful color pixels.

```
In [ ]: # Get coordinates of the plate

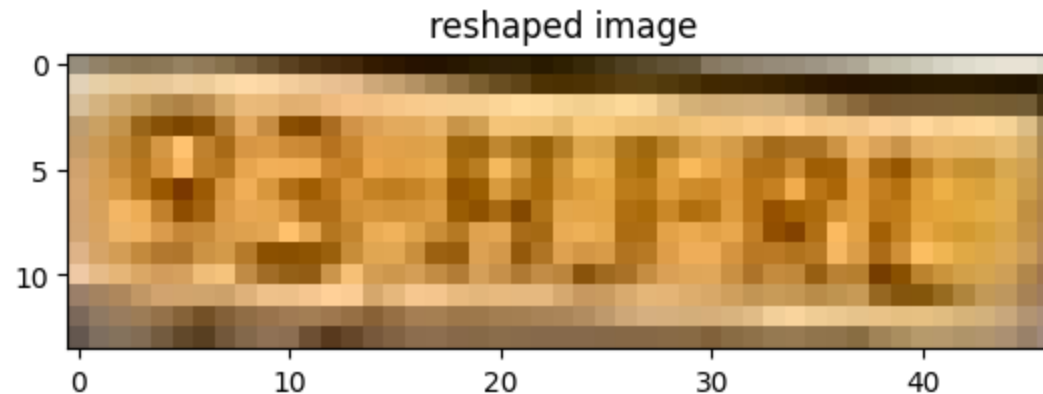
nonzero_indices = np.where(targetHSVHue > 0)

min_y = np.min(nonzero_indices[0])
```



```
max_y = np.max(nonzero_indices[0])
min_x = np.min(nonzero_indices[1])
max_x = np.max(nonzero_indices[1])

reshaped_image = imageRGB[min_y:max_y, min_x:max_x]
plotImage(reshaped_image, "Cropped image")
# Plot the cropped image
```



## Questions section

**?** (type your answer in the yellow field below) *Write the most appropriate color range you determined, with a 1 sentence explanation of whether inspecting the histogram was successful and why*

**Type your answer here.** range was from 10 - 30 for the hue and 100 - 255 for the saturation and 100 - 255 for the value I used color picker + trial and error, the histogram wasn't very clear for me, didn't see how to see values.

## PYTHON HANDS-ON Assignment I.3: Histogram comparison



One of your colleagues comes to you with a hypothesis and wants to ask you about your opinion. He asks you if it is possible to differentiate between different characters from their histogram. He says, since every character on a plate has a distinct shape and color distribution, maybe this could be shown from the histograms. To help you investigate, he provides you with the plate and cropped characters from it.

**Completion requirements for this assignment:**

- ☐ Plot and compare the histograms of the cropped characters
- ☐ Compute euclidean distances
- ☐ Answer the hypothesis

### I.3.1 Histogram comparison

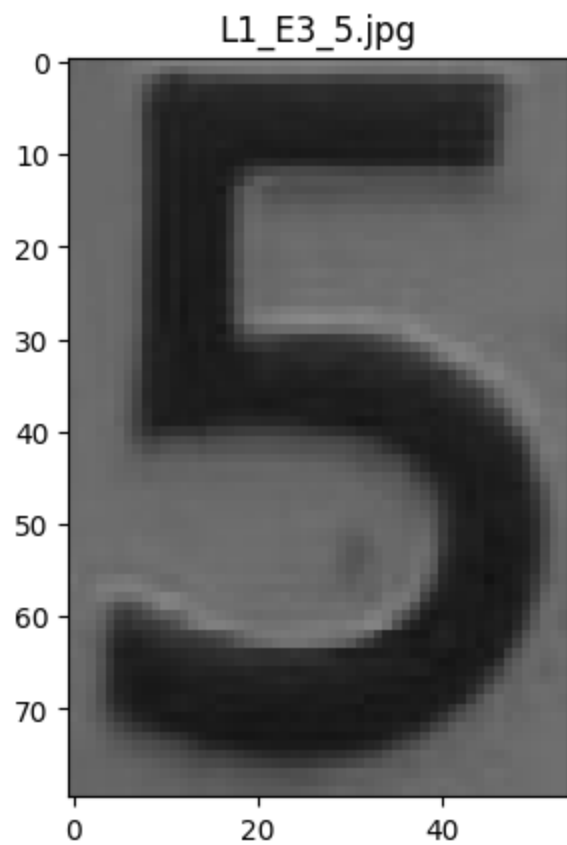
To verify his hypothesis, you execute the following experimental steps:

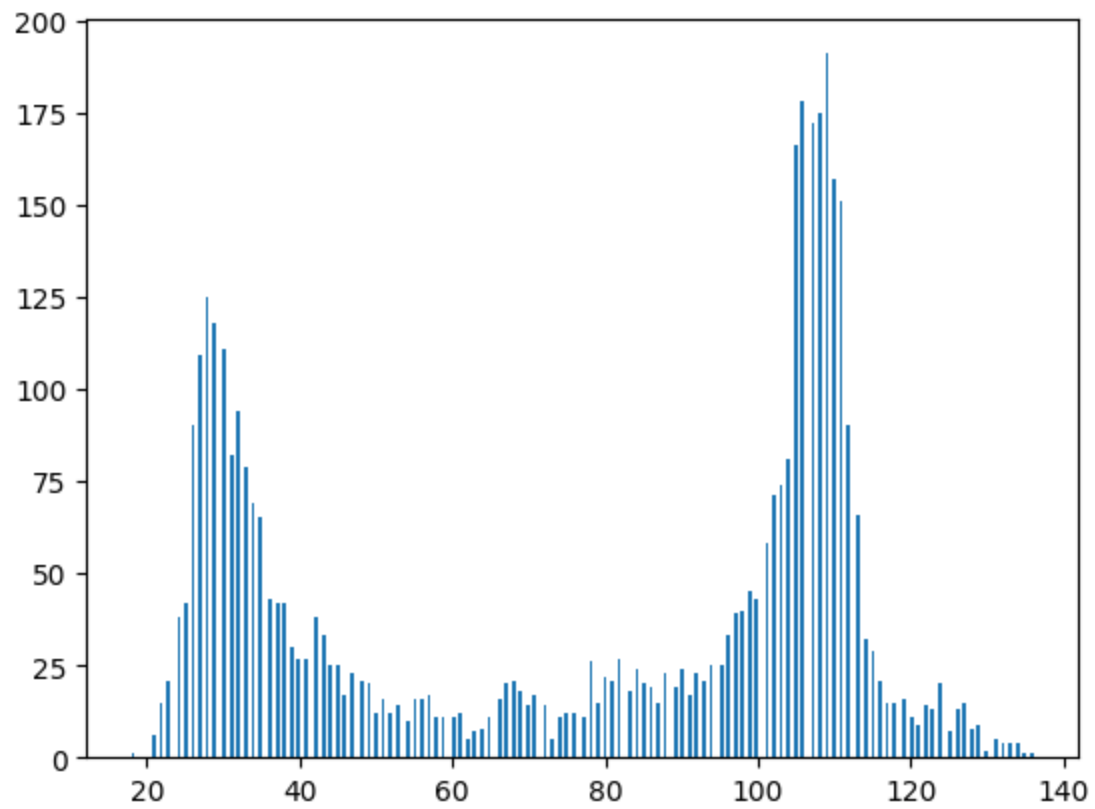
- Plot each character, convert to greyscale & compute then plot its histogram.
- Visually inspect if you see any differences between different character histograms.
- Compute the euclidean distance between the first character's histogram and all the rest.
- Write in max 3 sentences your conclusion on whether it is possible to do such a distinction based on the histograms or not and why.

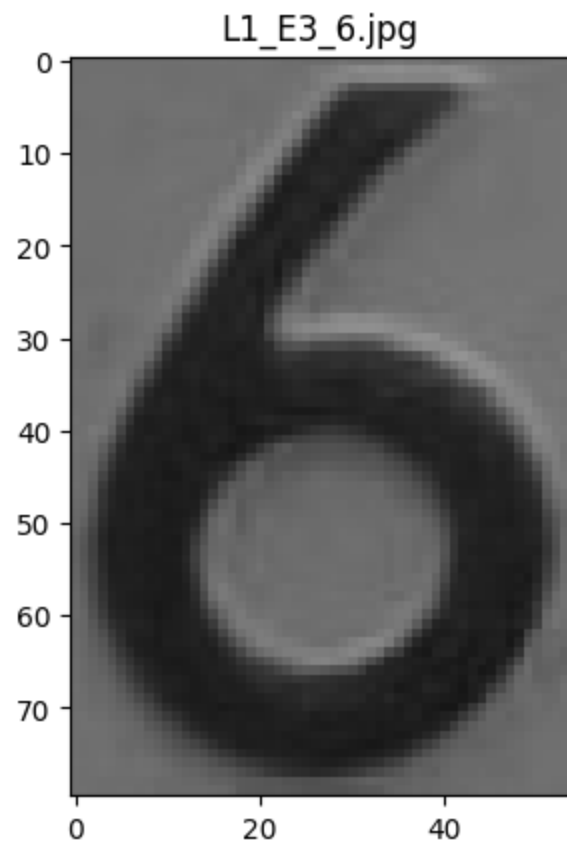
```
In [ ]: # Load images
path = "resources/lab_01/exercise_03/"

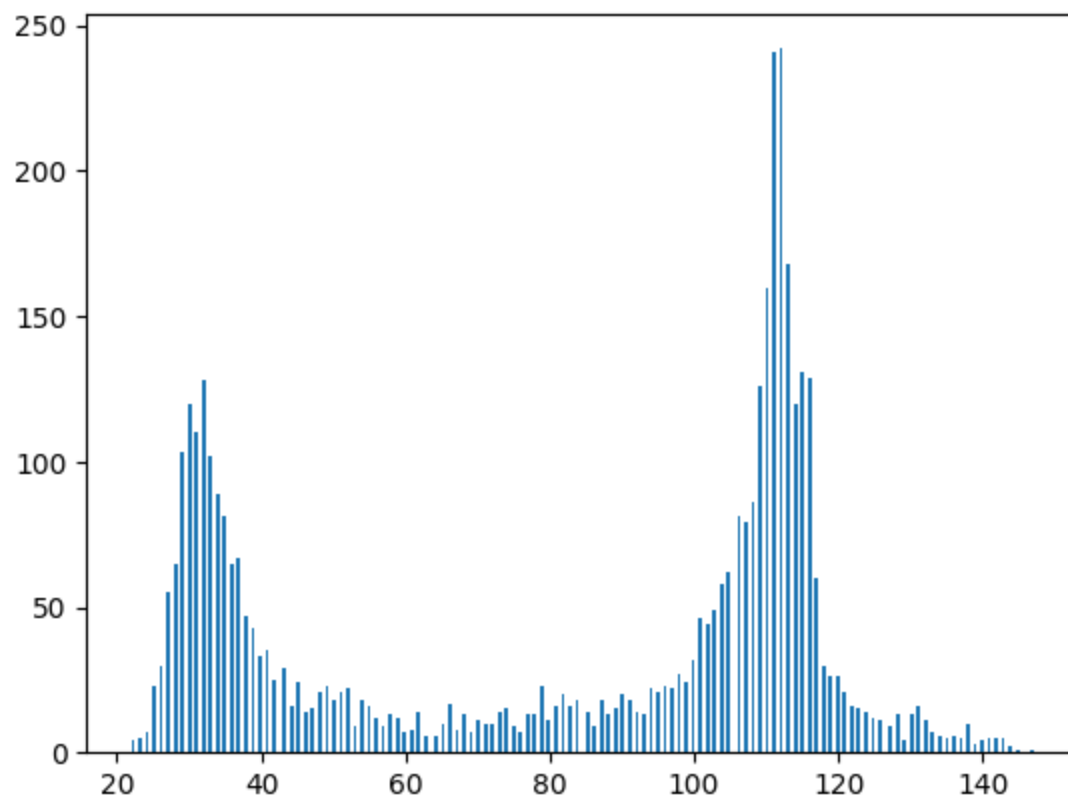
# Plot and compare all histograms
letters = ["L1_E3_5.jpg", "L1_E3_6.jpg", "L1_E3_J.jpg", "L1_E3_T.jpg", "L1_E3_T_2.jpg", "L1_E3
images = []
for letter in letters:
    image = cv2.imread(path + letter)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    images.append(image)
```

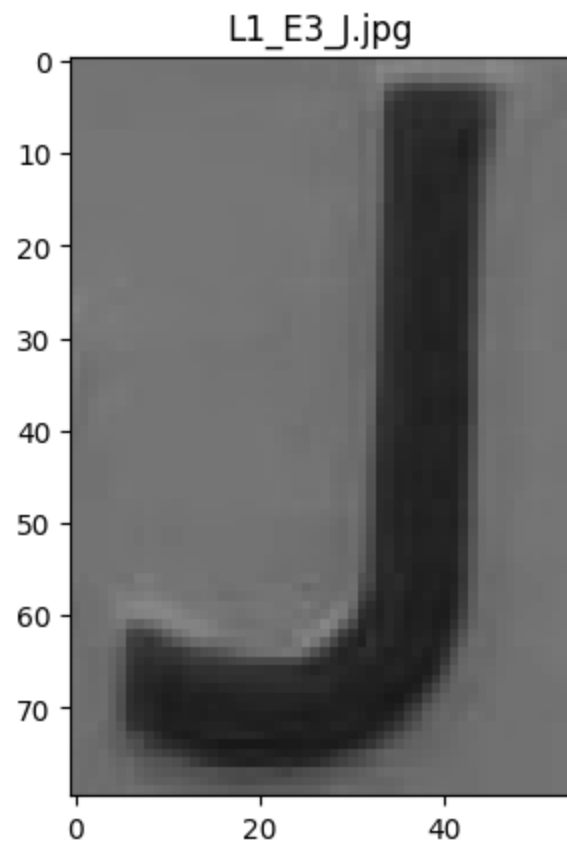
```
for i in range(len(images)):
    image = images[i]
    letter = letters[i]
    plotImageAndHistogram(image, letter, "gray")
```

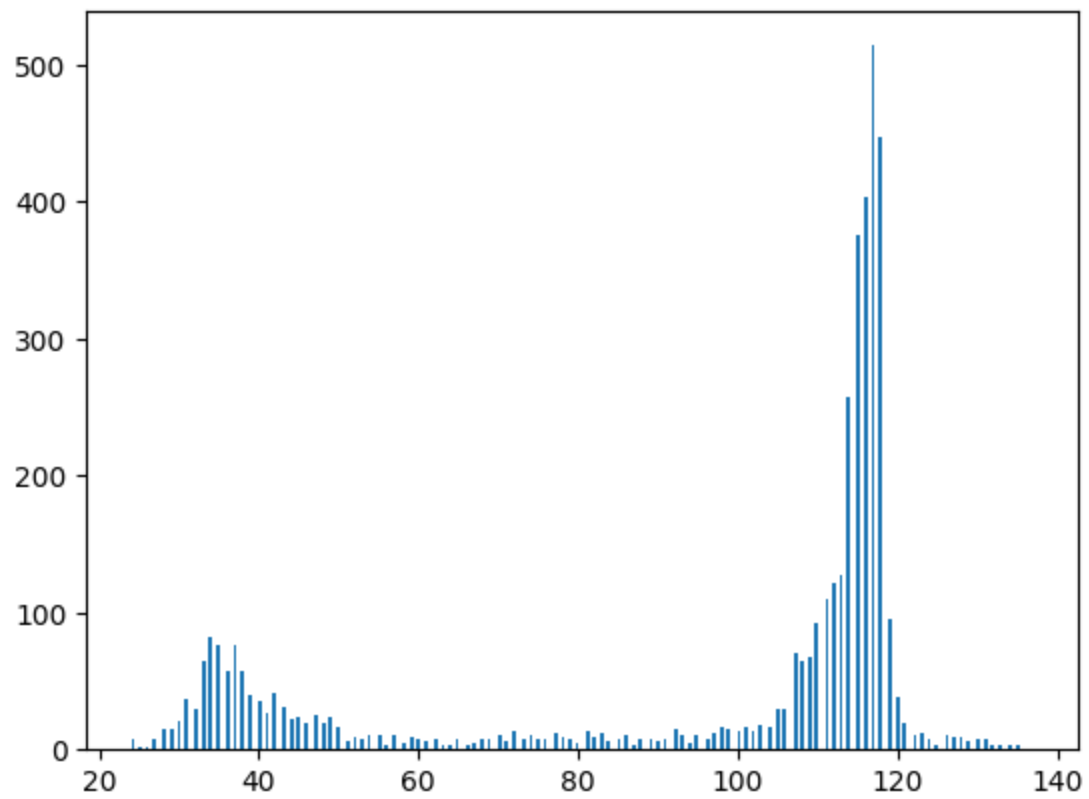




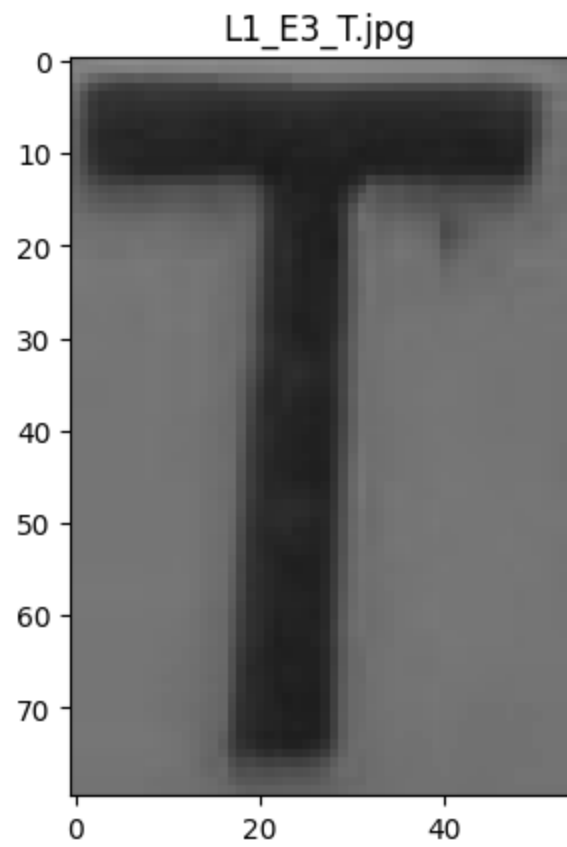


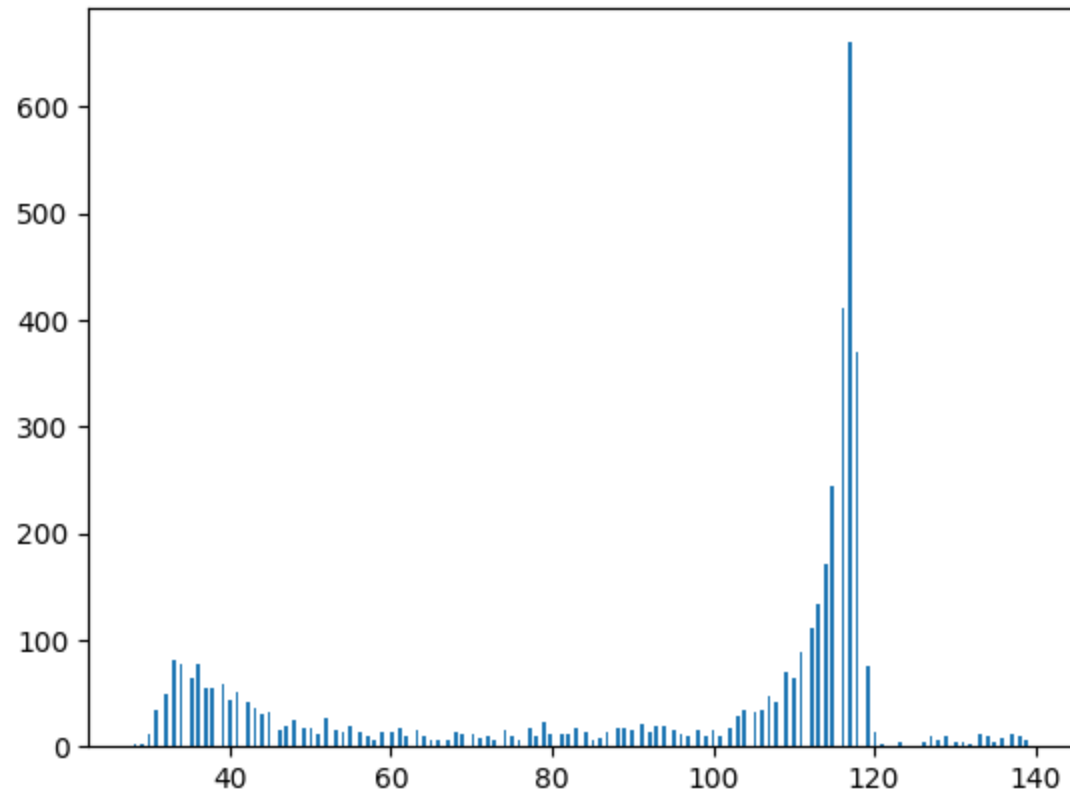


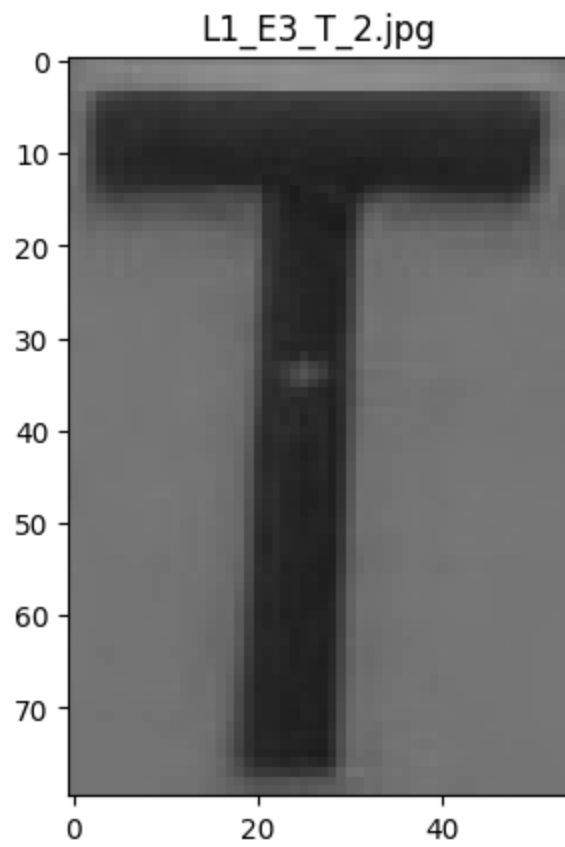


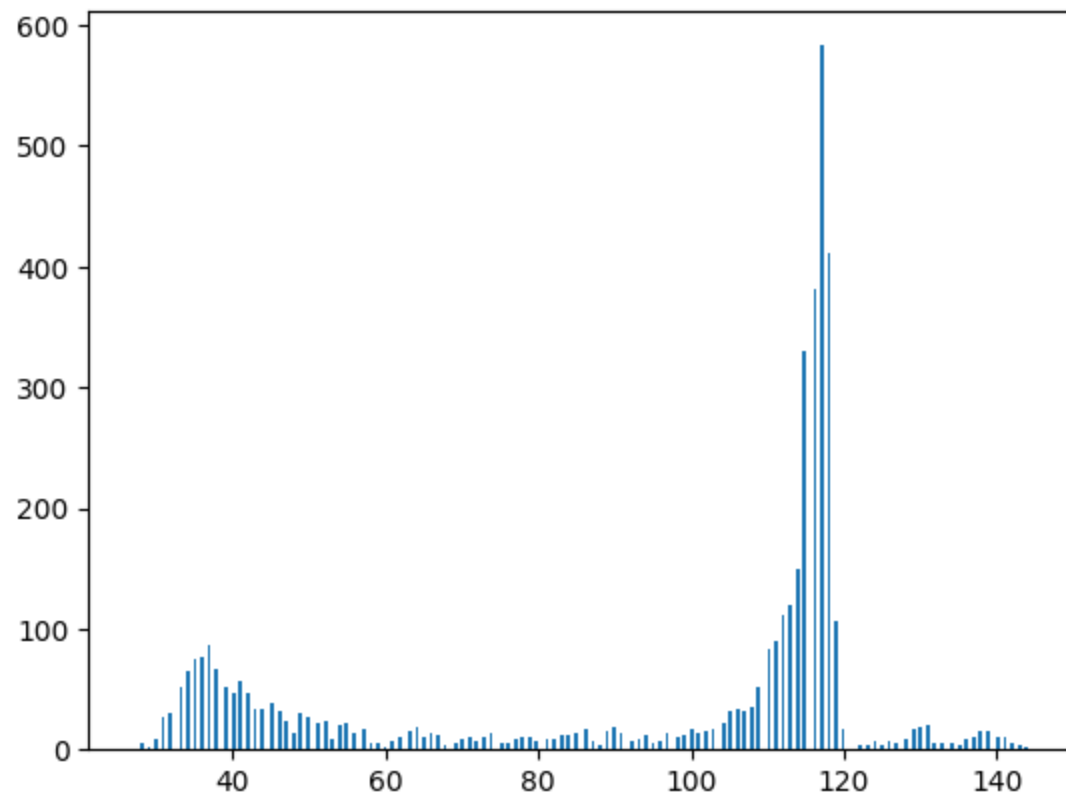


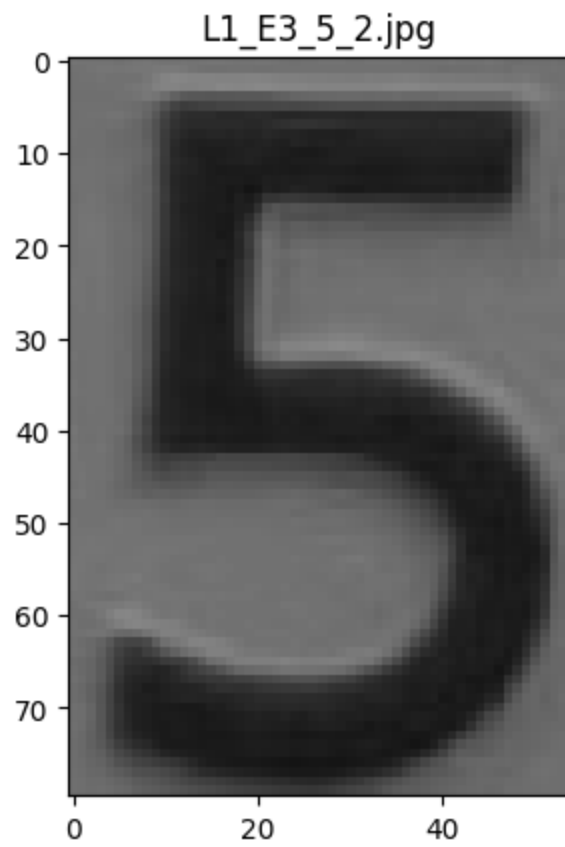


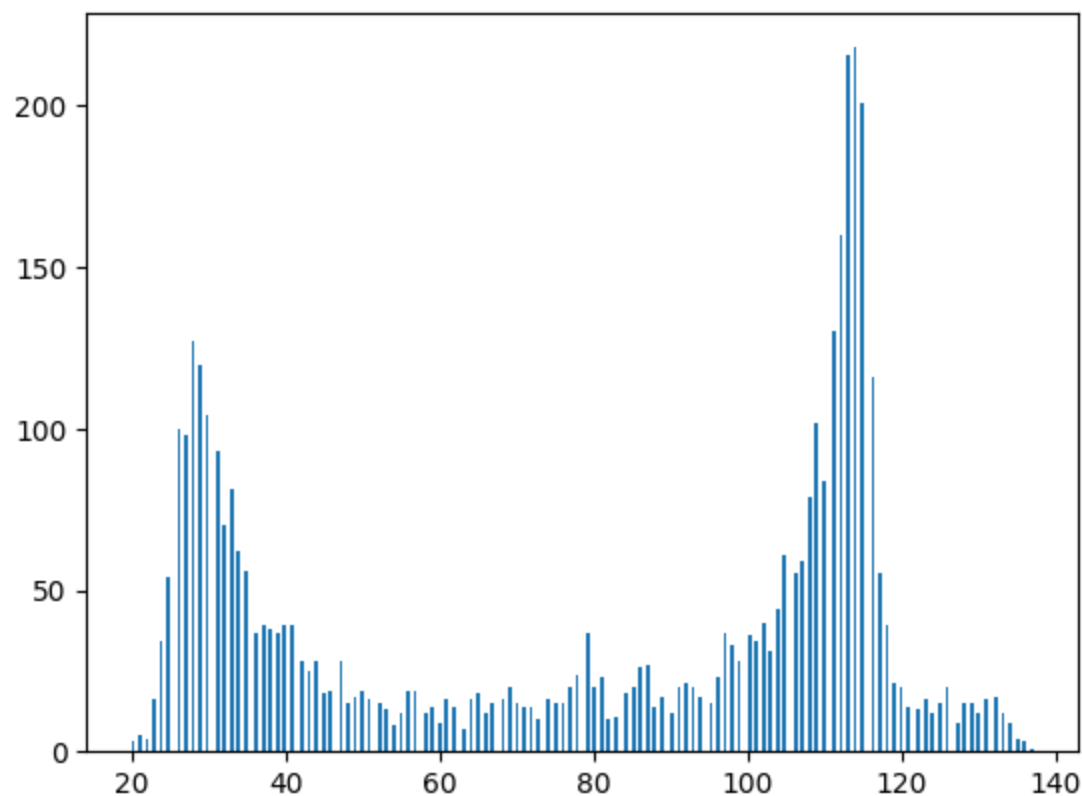












```
In [ ]: # Create function to calculate euclidean distance between greyscale images' histogram (2d array)
# ! [img1] and [img2] must have the same shape
def euclideanDistance(img1, img2):
    hist1 = img1.flatten()
    hist2 = img2.flatten()
    distance = np.linalg.norm(hist1 - hist2)

    return distance

first, bins = np.histogram(images[0].flatten(), 256, [0, 255])

for i in range(1, len(images)):
    image = images[i]
    hist, bins = np.histogram(image.flatten(), 256, [0, 255])
    distance = euclideanDistance(first, hist)
```

```
print(f'diatance : {distance}')
# Compute the eucledian distance between the first character and each of the others
```

```
diatance : 371.4754366038218
diatance : 964.8232998844918
diatance : 977.9693246723028
diatance : 964.6761114488116
diatance : 418.2391660282427
```

## Questions section

**?** (type your answer in the yellow field below in max 3 sentences) *Is it possible to distinguish characters based on the histograms? Why or why not?*

**Type your answer here.** No it is not possible. The histogram only gives us a distribution of color, but if size changes there can be similar distributions without the same character. The histogram does not take into account the order of the pixels.

**?** (type your answer in the yellow field below in max 3 sentences) *Is it possible to distinguish characters based on the euclidean distances of the histograms? Why or why not?*

**Type your answer here.** Again No, because if the histogram doesn't give us an ordering neither does the distance. The histogram distances show the difference in distribution, but can't really tell us a shape change more so than a color change. So we face the same problem.

**⚠️ \*\*Checklist before submitting\*\***

(try to fix anything that you can't tick from the checkboxes below)

```
In [ ]: # !pip install ipywidgets

import ipywidgets as widgets
```

```

check1 = widgets.Checkbox(
    value=False,
    description='Filled in all of the yellow fields with answers to the questions marked with
    indent=False,
    layout={'width': '1000px'}
)
check2 = widgets.Checkbox(
    value=False,
    description='Haven\'t used imports outside of those provided in the original notebook for
    indent=False,
    layout={'width': '1000px'}
)
check3 = widgets.Checkbox(
    value=False,
    description='All the code blocks can be run in sequence and execute successfully',
    indent=False,
    layout={'width': '1000px'}
)
check4 = widgets.Checkbox(
    value=False,
    description='Haven\'t changed the layout or formatting in the notebook and have only writ
    indent=False,
    layout={'width': '1000px'}
)
display(check1, check2, check3, check4)

```

Checkbox(value=False, description='Filled in all of the yellow fields with answers to the qu  
estions marked wit...

Checkbox(value=False, description="Haven't used imports outside of those provided in the ori  
ginal notebook for...

Checkbox(value=False, description='All the code blocks can be run in sequence and execute su  
ccessfully', inden...

Checkbox(value=False, description="Haven't changed the layout or formatting in the notebook  
and have only writ...

In [ ]: