M20 - Introduction to Computer Programming with MATLAB
Instructor: Prof. Enrique López Droguett, Ph.D.
Teacher Assistants: M. Fidansoy, G. San Martín, M. Pishahang, V. Vela.
Fall 2023 – UCLA
Student: *Alex Lie*
UCLA ID: *905901892*

HOMEWORK 2

Task 1: Histogram and data descriptors

Introduction

I am using MATLAB to graph histograms of the maximum lateral force thirty pieces of structural timber could withstand right before they failed. I am modeling 5 histograms with different bin widths to determine which histogram can represent the data the best. I am also calculating the mean and standard deviation of the data, as well as creating a sample of 10,000 pieces of timber following a normal distribution to estimate the 10th and 90th percentile of the breaking force.

Model and Theory

N/A

Methodology

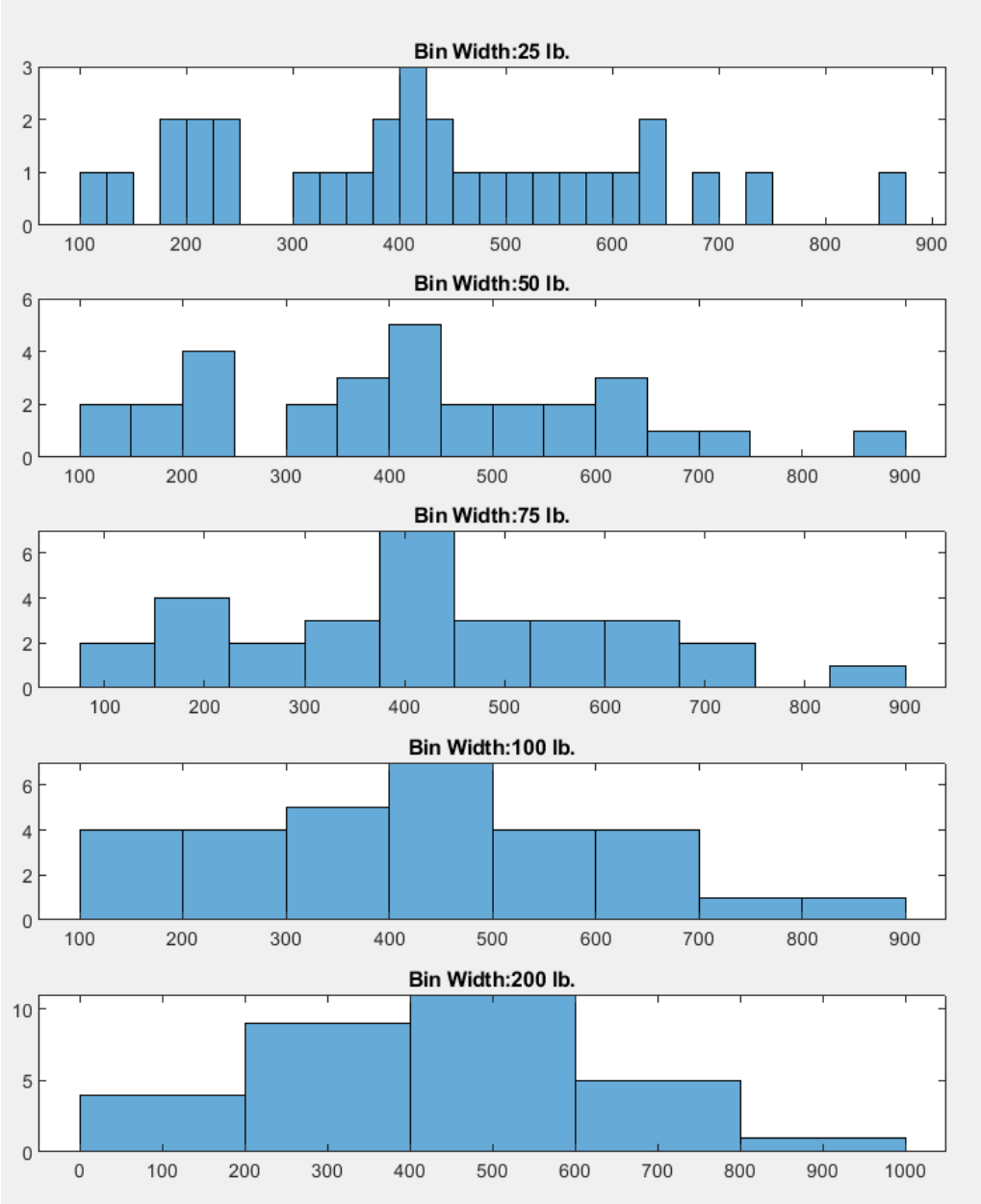
I first cleared the workspace and command window. Then, I inputted data of the timber breaking force given in the problem in a row vector and the varying bin widths in a separate row vector. Then I created five histograms using the subplot(), histogram(), and title() commands. Since I am making 5 histograms, I wrote my code using a for-loop to reduce the amount of code I had to write and to avoid repetitiveness. My for-loop iterates five times, and each time it accesses a different element in the row vector with the bin widths. I then calculated the mean and standard deviation of the timber breaking force data using the mean() and std() functions. I then created a sample of 10,000 timber breaking forces using the normrnd() function, using the mean and standard deviation I calculated as well as N=10,000 as arguments. I then sorted the sample using the sort() function, and found the 10th and 90th percentile of the sample by finding the 100th and 900th element in the sample by using an access function. You can then compare the 10th and 90th percentile of the sample I created to the 10th and 90th percentile of the actual data collected. To do this, you need to sort the actual data, and then find the 3rd and 27th elements. Lastly, I added comments to make my code easier to understand for other users.

Calculations and results

See next page.

Student: *Alex Lie*
UCLA ID: *905901892*

1)



2)

```
mean =

   423.8333


std =

   187.0075


P10 =

   183.8547


P90 =

   664.0669


dP10 =

   196


dP90 =

   643
```

Five histograms of the timber breaking force data of different bin widths (25, 50, 75, 100, and 200 lbs.) are stacked on top of each other for easy comparison between them. The right side shows the command window outputs of the mean, standard deviation, sample 10th percentile, sample 90th percentile, data 10th percentile, and data 90th percentile.

Discussions and Conclusions

See next page.

Based on looking at all five histograms (1), I believe that the histogram with the bin width of 75 lbs. is most meaningful because it starts to resemble a normal distribution, and the bin widths are not so large that it hides small-scale details.

When I created my sample, I also created a histogram with it (not shown in this report, but you are able to view it in my code if you uncomment that section), it follows a normal distribution. It is unimodal, symmetrical, and has no outliers.

The 10th and 90th percentiles from my sample and the actual data (2) are close but differ. Since my sample is randomly generated, there will be a different 10th and 90th percentile value each time I run my code. But the percentiles of the sample is in the ballpark of the percentiles of the actual data. The 10th percentile of my sample tends to be smaller than the 10th percentile of the actual data, and the 90th percentile of my data tends to be larger than the 90th percentile of the actual data.

Task 1: Green-screen graphics

Introduction

I am using MATLAB to remove the green-screen background of a dinosaur picture and placing the dinosaur in a scenic background. I am also creating three histograms, each one showing a different color channel (red, green, and blue) in the dinosaur picture, that I can use to establish a criteria to determine which pixels are part of the green screen and which ones are not. Using that data, I can replace certain pixels on the background image with those on the dinosaur images to create the green-screen effect.

Model and Theory

N/A

Methodology

I first cleared the workspace and command window. I then used the imread() and imshow() functions to have MATLAB read (create a 3-dimensional matrix and store it) and show the image in a figure. I then created three different histograms to display the three different color channels in the object image using subplot(), histogram(), and title() functions. Since I needed to repeat this 3 times, I wrote my code using a for-loop to reduce the amount of code I needed to write and avoid repetition. My for-loop iterates 3 times. For the histogram function, I only called out the certain color value the histogram corresponds to. Red is equal to the first index, green is equal to the second index, and blue is equal to the third index in the third dimension. I also used the strcat() function and the “colors” row vector to create the histogram titles.

I then used a nested for-loop to iterate through each pixel in the image matrix. I then used an if-statement to determine if the red, green, and blue value of each pixel met my criteria. If the red value and blue value are not less than 100, and the green value is not larger than 100, then that pixel is not a green-screen pixel (I chose these numbers because it reduces the amount of green-screen pixels at the edge of the dinosaur the best). If it met my criteria, then the red, green, and blue values on the pixel on the background image will be set to the red, green, and blue values of the corresponding pixel from the object image, forming the dinosaur on top of the background image. Once iterating through every pixel, the final result of the green-screen effect is shown. Lastly, I added comments to make my code easier to understand for other users.

M20 - Introduction to Computer Programming with MATLAB
Instructor: Prof. Enrique López Droguett, Ph.D.
Teacher Assistants: M. Fidansoy, G. San Martín, M. Pishahang, V. Vela.
Fall 2023 – UCLA
Student: *Alex Lie*
UCLA ID: *905901892*

Calculations and results

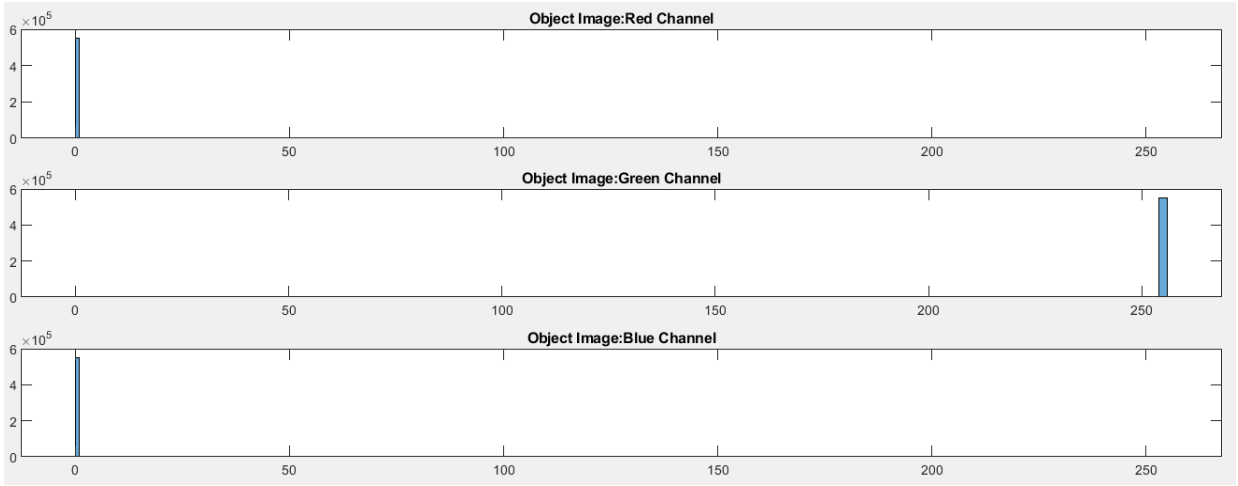
1)

2)

3)

The object and background image I showed in MATLAB were used to create the bottom image which is both of the images combined with green-screen effect.

4)



The histogram shows the red, green, and blue color channels of the object image.

Discussions and Conclusions

There are 589824 pixels in the object image, most of them being green-screen pixels. The histograms (4) indicate what most of the red, green, and blue values of the green-screen pixels are, indicated by the tallest bar in each histogram. Most of the red and blue values are 0 and most of the green values are 254 or 255 for a green-screen pixel. To check if a pixel was not a green-screen pixel, my condition in my if-statement checked at first if the red and blue values are both not less than 10 and the green value is not greater than 245 (I added some safety). When I ran my code, I saw that there were still some green-screen pixels at the edge of the dinosaur. To get rid of those, I modified my condition and tried different values until I found a set of values that got rid of most green-screen pixels, without getting rid of pixels from the object. My condition ended up checking if the red and blue values are both not less than 100 and the green value is not greater than 100. The result (3) ended up looking much better, with no green-screen pixels visible without zooming in.