# Chapter 3

# Methodology

## 3.1 Introduction

The whole implementation of this research is processed using MATLAB environment, which has the necessary tools and functions for the propose of the research.

## 3.2 System flowchart

The flowchart 3.1 shows the procedures that followed by the application in order to implement different functions and getting the results.

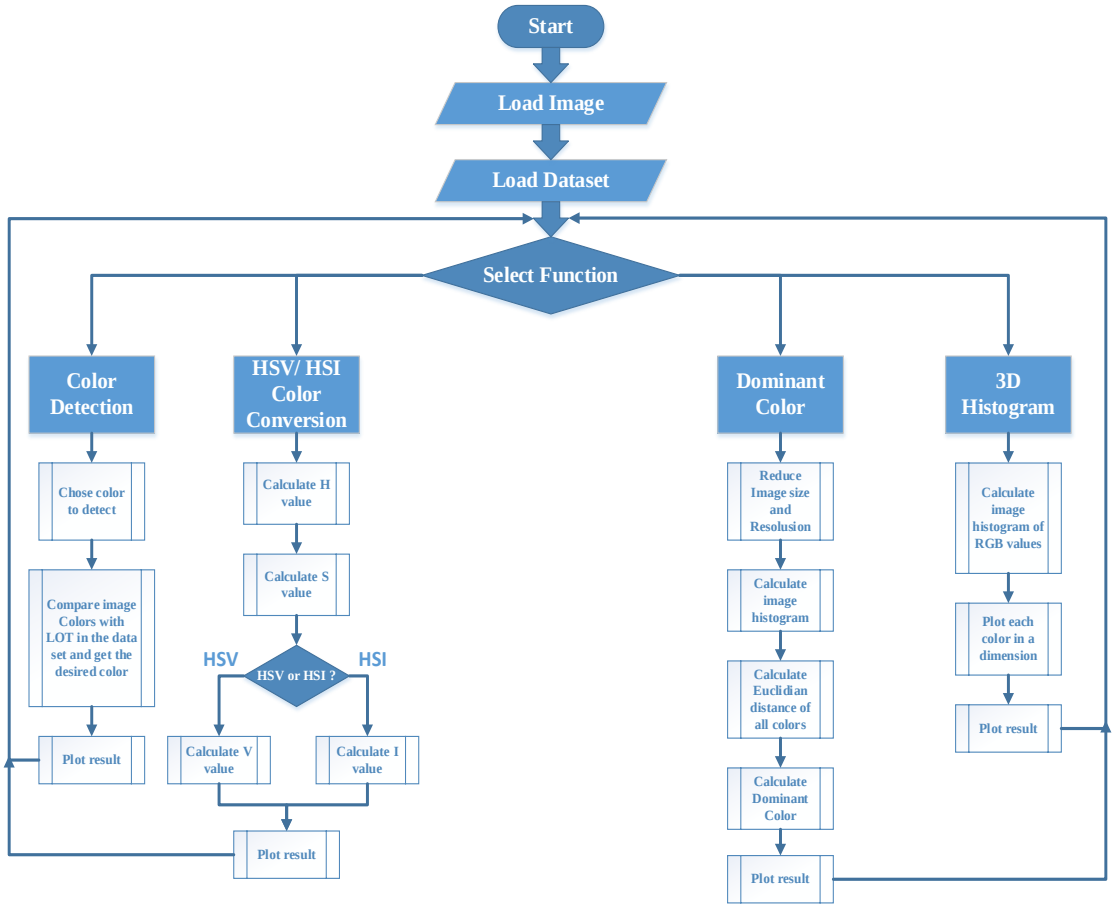


**Figure 3- 1 :** System flowchart

## 3.3 Color Detection Algorithm

Describe the fuzzy logic algorithm of the color detection and the dataset used for fuzzy logic processing.

**Step1:** Load the image as three-dimension array, each dimension represents the value of the color of Red, Green, and Blue.

**Step2:** Convert the value of the colors are changes from 0-255 range to 0-1 range by dividing over 255 and take the double result.

**Step3:**Reshape the converted 0-1 RGB image into one-dimension array for linear processing, and take the length of the serial array.

**Step4:** Create an empty image with the same dimension and size for the results.

**Step5:**take the index of the query color or the color to be compared with the data set, to loop over the index of the query color and make interpolation for 3-D gridded data in mesh grid format with linear method and make the decision of the color name and modify empty result image by the colors that found only.

In the step 5, the interpolated values of a function of three variables at specific query points using linear interpolation. The results always pass through the original sampling of the function. R, G, and B nodes contain the coordinates of the sample points. Color Lookup table contains the corresponding function values at each sample point. R, G, and B contain the coordinates of the query image points.

## 3.4 Dataset content

Table 3.1 describes the content of the dataset in MATLAB used in the fuzzy logic detection of color, used to get the color names by comparing colors with the look up table.

**Table 3.1 :** Dataset Content description

| Column | Description | Data type | Size |
|---|---|---|---|
| NCOLORS | Number of colors defined in the data set. | integer | 1x1 |
| COLORNAMES | The Names of the defined colors | Cell ”String” | 1xNoOfColors |
| COLORLUT | Colors look up table to compare the difference between the source (defined) color and the color of image pixel. (polygons) | Cell “Doubles” | 1xNoOfColors Of 52x52x52 double |
| RNODE | The red polygonsof the colors used in colors definition and interpolation. | double | 1x52 |
| GNODE | The green polygonsof the colors used in colors definition and interpolation. | double | 1x52 |
| BNODE | The blue polygonsof the colors used in colors definition and interpolation. | double | 1x52 |

## 3.5 Dominant Color

**Step 1:** Reduce the image dimensions (Resize to ¼ ) in order to reduce the time processing.

**Step 2:** Reduce the image resolution (multiply by constant “0.25”) to reduce the time processing and better results.

**Step 3:** Compute the image histogram to find the color pixel pins from the image and to retrieve the RGB values (cores) and the frequency of each color.

**Step 4:** Compute Euclidian distance of each color to all colors.

**Step 5:** Identify the colors nearest to the most predominant color.

**Step 6:** Find the colors with distance less than or equal 3.

**Step 7:** Compute weighted mean of the colors.

### 3.6 Conversion from RGB to HSV

**Input**: RGB
**Output**: HSV
**Method**:
Step1: *[Find the max and min values]*
M = max (R, G, B), m = min(R, G, B)
Step2: *[normalized the RGB values to be in the range [0, 1]]*
r = (M-R)/(M-m)
g = (M-G)/(M-m)
b = (M-B)/(M-m)
Step3: *[Calculate V value]*
V = max (R, G, B)
Step4: *[Calculate S value]*
if M = 0 then S = 0 and H = 180 degrees
if M <> 0 then S = (M - m) / M
Step5: *[Calculate H value]*
if R = M then H = 60(b-g)
if G = M then H = 60(2+r-b)
if B = M then H = 60(4+g-r)
if H >= 360 then H = H - 360
if H < 0 then H = H + 360
Where H in the range [0,360], S and H in the Range [0,100]
Step6: *[output HSV]*
The calculated H, S, and V are the output of the algorithm.
**End**

## 3.7 Conversion from RGB to HSI

**Input**: RGB
**Output**: HSI
**Method**:
Step1: *[Find the max and min values]*
M = max (R, G, B), m = min(R, G, B)
Step2: *[normalized the RGB values to be in the range [0, 1]]*
r = (M-R)/(M-m)
g = (M-G)/(M-m)
b = (M-B)/(M-m)
Step3: *[Calculate I value]*
I = (R' + G' + B')/3
Step4: *[Calculate S value]*
if M = 0 then S = 0 and H = 180 degrees
if M <> 0 then S = (M - m) / M
Step5: *[Calculate H value]*
if R = M then H = 60(b-g)
if G = M then H = 60(2+r-b)
if B = M then H = 60(4+g-r)
if H >= 360 then H = H - 360
if H < 0 then H = H + 360
Where H in the range [0,360], S and H in the
Range [0,100]
Step6: *[output HSI]*
The calculated H, S, and I are the output of the algorithm.
**End**

### 3.8 Histogram

In an image processing context, the histogram of an image normally refers to a histogram of the pixel intensity values. This histogram is a graph showing in figure(3-1)the number of pixels in an image at each different intensity value found in that image. For an 8-bit grayscale image there are 256 different possible intensities, and so the histogram will graphically display 256 numbers showing the distribution of pixels amongst those grayscale values. Histograms can also be taken of color images either individual histograms of red, green and blue channels can be taken, or a 3-D histogram can be produced, with the three axes representing the red, blue and green channels, and brightness at each point representing the pixel count. The exact output from the operation depends upon the implementation it may simply be a picture of the required histogram in a suitable image format, or it may be a data file of some sort representing the histogram statistics.

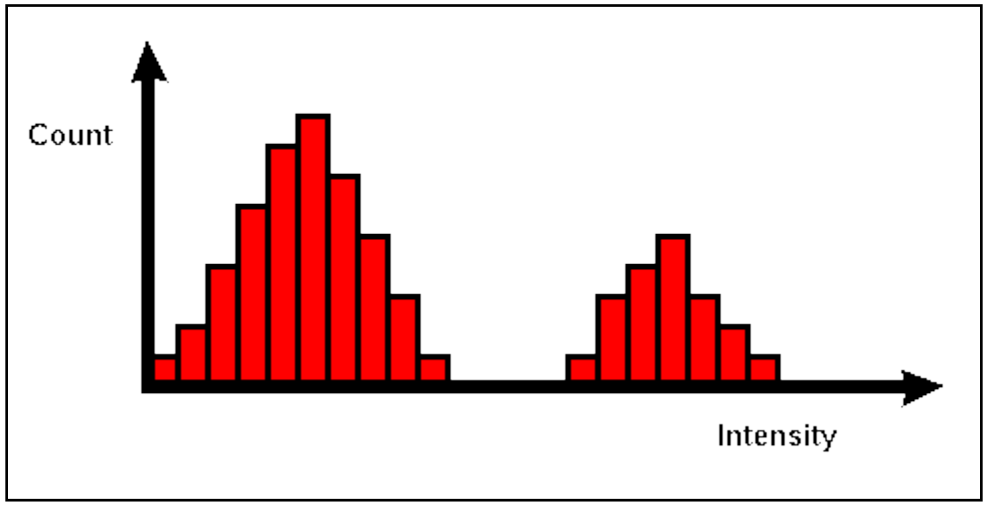


**Figure 3- 1**: Histogram values representation