Esin ERSOĞAN
708191001

BLG 513E Image Processing
Homework 3 Report

27.12.2019

1. Implementation

In this homework, a color classification model has been implemented. The required steps to construct such a development is listed below:

Red, Black, Orange, Yellow, Green, Blue, White, and Violet folders are created and they are included different tones of colors learn HSV intervals of each color.

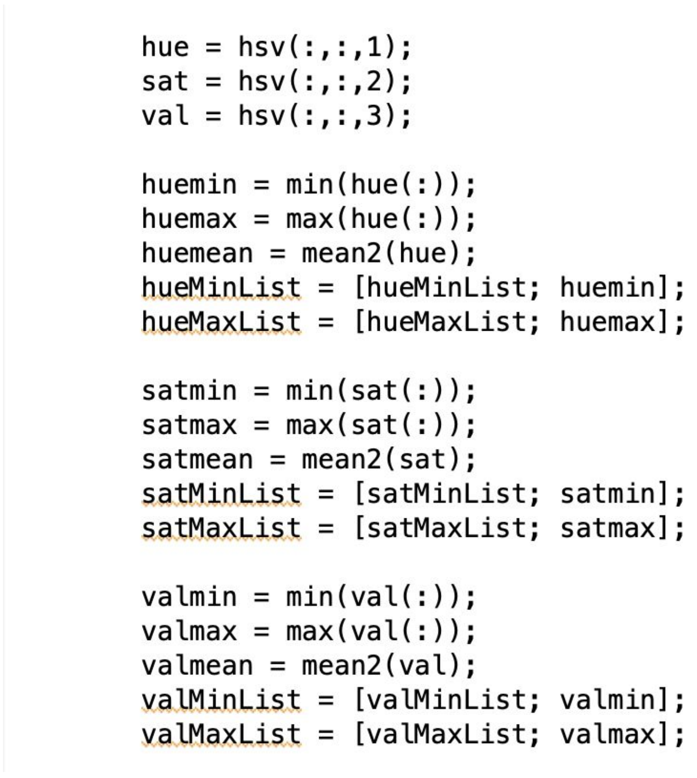
1.1. Get Color HSV Intervals:

The first step to recognize the colors which is included in images is the convention of the image from RGB space to HSV space. This convention is needed because, the red, green and blue channels are dependent informations in RGB space. However, in HSV space, the hue, saturation and value are independent informations.

HSV space is composed of hue, saturation and value. Every color has its own intervals in terms of hue, saturation and value independently.

Therefore, the intervals of hue, saturation and value informations for each color should be found so as to obtain the features of the colors.

```
function [hsvList,rowNames] = getColorHSVintervals(folderName, count)

    folder = folderName;

    fileExtension = fullfile(folder, '*.png');
    files = dir(fileExtension);

    hueMinList = [];
    hueMaxList = [];
    satMinList = [];
    satMaxList = [];
    valMinList = [];
    valMaxList = [];

    hsvList = [];
    rowNames = [];

    for k = 1 : length(files)
        root = files(k).name;
        fileName = fullfile(folder, root);
        im = imread(fileName);

        hsv = rgb2hsv(im);
```



```
hue = hsv(:,:,1);
sat = hsv(:,:,2);
val = hsv(:,:,3);

huemin = min(hue(:));
huemax = max(hue(:));
huemean = mean2(hue);
hueMinList = [hueMinList; huemin];
hueMaxList = [hueMaxList; huemax];

satmin = min(sat(:));
satmax = max(sat(:));
satmean = mean2(sat);
satMinList = [satMinList; satmin];
satMaxList = [satMaxList; satmax];

valmin = min(val(:));
valmax = max(val(:));
valmean = mean2(val);
valMinList = [valMinList; valmin];
valMaxList = [valMaxList; valmax];
```

In getColorHsvintervals.m file:

To obtain the features of the colors, the images in each folder are analyzed. They converted to HSV using rgb2hsv function. After that, hue, saturation and value channels extracted from these HSV images. For each image, the hue-saturation-value intervals obtained at the end of this process.

1.2. Training Classifier

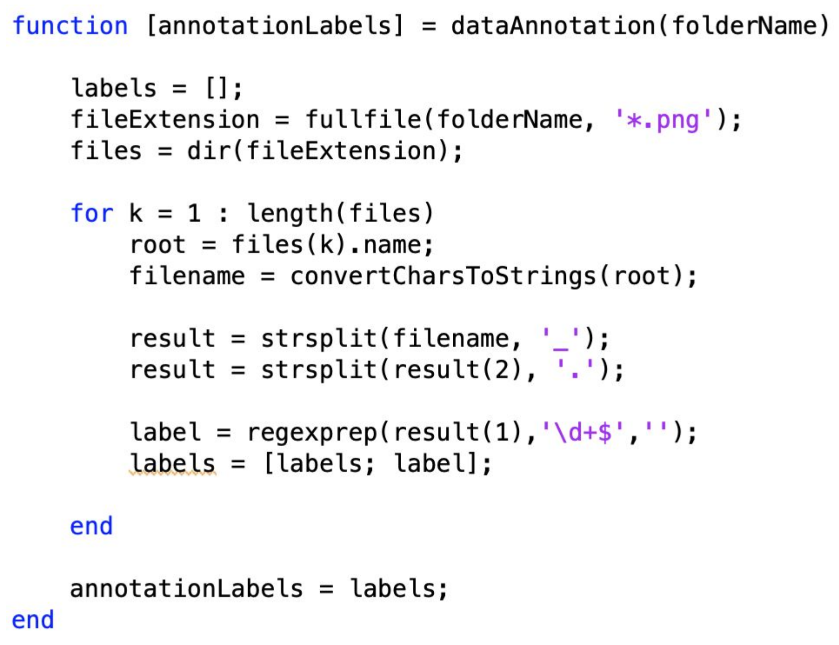
In this step, each hue-saturation-value intervals which is obtained from each image in the color folders is added to the classification model and labeled as their color.

At the end of this process, the model is trained with training samples.

```
function [] = trainClassifier()

    global hsvListAll
    hsvListAll = [];
    global rowNamesAll
    rowNamesAll = [];

    [hsvList, rowNames] = getColorHSVintervals("black", 1);
    hsvListAll = [hsvListAll; hsvList];
    rowNamesAll = [rowNamesAll; rowNames];

    [hsvList, rowNames] = getColorHSVintervals("blue", size(rowNamesAll,1)+1);
    hsvListAll = [hsvListAll; hsvList];
    rowNamesAll = [rowNamesAll; rowNames];

    [hsvList, rowNames] = getColorHSVintervals("green", size(rowNamesAll,1)+1);
    hsvListAll = [hsvListAll; hsvList];
    rowNamesAll = [rowNamesAll; rowNames];

    [hsvList, rowNames] = getColorHSVintervals("orange", size(rowNamesAll,1)+1);
    hsvListAll = [hsvListAll; hsvList];
    rowNamesAll = [rowNamesAll; rowNames];


    [hsvList, rowNames] = getColorHSVintervals("red", size(rowNamesAll,1)+1);
    hsvListAll = [hsvListAll; hsvList];
    rowNamesAll = [rowNamesAll; rowNames];

    [hsvList, rowNames] = getColorHSVintervals("violet", size(rowNamesAll,1)+1);
    hsvListAll = [hsvListAll; hsvList];
    rowNamesAll = [rowNamesAll; rowNames];

    [hsvList, rowNames] = getColorHSVintervals("white", size(rowNamesAll,1)+1);
    hsvListAll = [hsvListAll; hsvList];
    rowNamesAll = [rowNamesAll; rowNames];

    [hsvList, rowNames] = getColorHSVintervals("yellow", size(rowNamesAll,1)+1);
    hsvListAll = [hsvListAll; hsvList];
    rowNamesAll = [rowNamesAll; rowNames];
```

1.3. Data Annotation

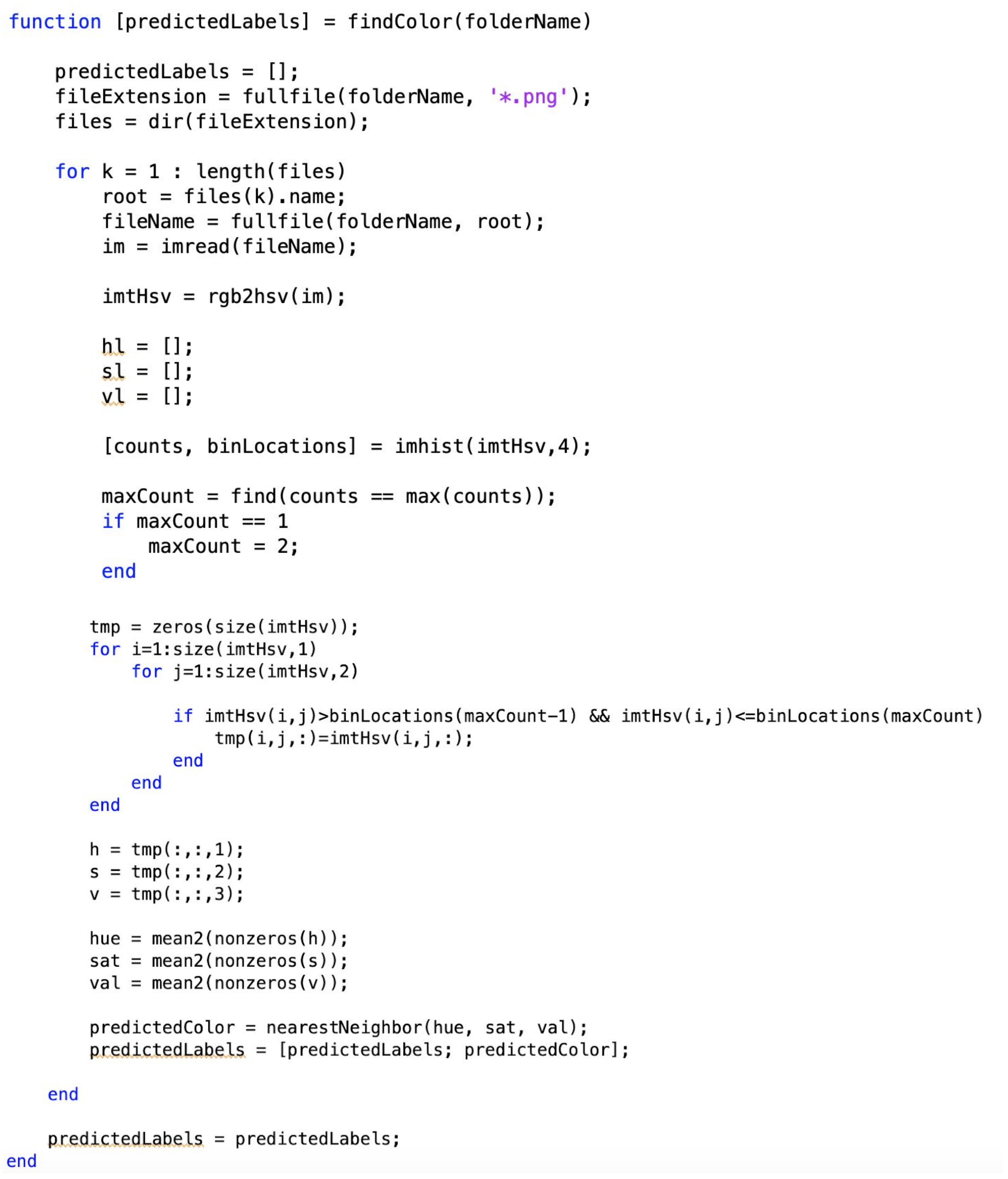
The test data which is included in the ‘testdata’ folder is annotated manually to analyze the correctness of predicted colors which is done by our classification model.



```
function [annotationLabels] = dataAnnotation(folderName)

    labels = [];
    fileExtension = fullfile(folderName, '*.png');
    files = dir(fileExtension);

    for k = 1 : length(files)
        root = files(k).name;
        filename = convertCharsToStrings(root);

        result = strsplit(filename, '_');
        result = strsplit(result(2), '.');

        label = regexprep(result(1),'\d+$','');
        labels = [labels; label];

    end

    annotationLabels = labels;
end
```

1.4. Color Recognition

For each test image, the image is converted to HSV from RGB so as to find the hue-saturation-value intervals separately. These obtained values are sent to the Nearest Neighbor classifier to decide the class which image is belonging to.



```
function [predictedLabels] = findColor(folderName)

    predictedLabels = [];
    fileExtension = fullfile(folderName, '*.png');
    files = dir(fileExtension);

    for k = 1 : length(files)
        root = files(k).name;
        fileName = fullfile(folderName, root);
        im = imread(fileName);

        imtHsv = rgb2hsv(im);

        hl = [];
        sl = [];
        vl = [];

        [counts, binLocations] = imhist(imtHsv,4);

        maxCount = find(counts == max(counts));
        if maxCount == 1
            maxCount = 2;
        end

        tmp = zeros(size(imtHsv));
        for i=1:size(imtHsv,1)
            for j=1:size(imtHsv,2)

                if imtHsv(i,j)>binLocations(maxCount-1) && imtHsv(i,j)<=binLocations(maxCount)
                    tmp(i,j,:)=imtHsv(i,j,:);
                end
            end
        end

        h = tmp(:,:,1);
        s = tmp(:,:,2);
        v = tmp(:,:,3);

        hue = mean2(nonzeros(h));
        sat = mean2(nonzeros(s));
        val = mean2(nonzeros(v));

        predictedColor = nearestNeighbor(hue, sat, val);
        predictedLabels = [predictedLabels; predictedColor];

    end

    predictedLabels = predictedLabels;
end
```

1.5. Nearest Neighbor

Using these interval values, the image is assigned to the class which is the nearest neighbourhood. To predict the color of the test images, nearest neighbourhood algorithm is used as shown below:

```
function [predictedColor] = nearestNeighbor(hue, sat, val)

    hsvListAll = getGlobalhsvListAll;
    rowNamesAll = getGlobalrowNamesAll;

    absoluteDiff = [];

    for i=1:size(hsvListAll)
        absoluteDiff = [absoluteDiff; (abs(hue-hsvListAll(i,1)) + abs(sat-hsvListAll(i,2)) + abs(val-hsvListAll(i,3)))];
    end

    minDiffIndex = find(absoluteDiff == min(absoluteDiff));

    predictedColor = rowNamesAll(minDiffIndex);
end
```

1.6. Correct Classification Rate

The correct classification rate is calculated with comparing the annotated labels and predicted labels.

```
trainClassifier();

annotations = dataAnnotation("testdata");

predictedLabels = findColor("testdata");

trueEstimate = 0;
wrongEstimate = 0;

for i=1:size(annotations)
    if annotations(i) == predictedLabels(i)
        trueEstimate = trueEstimate + 1;
    else
        wrongEstimate = wrongEstimate + 1;
    end
end

correctClassificationRate = (trueEstimate/size(annotations,1))*100;
```

1.7. Train and Test Datasets:

Train Dataset: They are included in the folders: *red, black, orange, yellow, green, blue, white, violet.*
Test Dataset: They are included in the folder *testdata*.

https://drive.google.com/drive/folders/1PPrJxDIW1YJavPOin98ydHTHkdOALghO?usp=sharing

2. How to Use:

You can run *colorRecognition.m* file. It executes other files respectively.