BUTTERFLY CLASSIFICATION

Group members: Daria Kruzhinskaia, Yen-Ting Lee

1. Introduction of the project and dataset description

The aim of current work – to make an application that can predict category of an object. Project was developed to find a correct type of butterfly according to biological classification from picture of this butterfly.

As programming language Matlab was used. It is a multi-paradigm numerical computing environment. [1] It allows us to work with images as matrixes without loops.

In provided dataset there are photographs of ten (10) different kind of butterfly, **832** pictures in general. For each picture besides original dataset contain segmented image (image of the butterfly without natural background with black background instead) and output segmentation (binary image where white color represent shape of butterfly). For our method we have to use only segmented images and have to create a masking image (binary) inside the program (mostly due to the fact that we permuted all the images in dataset to provide a randomized amount of each butterfly kind to training set).

2. Methodology

Our algorithm is divided into three main steps: preprocessing (or preparation) of images, finding specific features for each picture and classification of these features using the similarity in the same type. Below at Figure 1 you can see the flowchart of the whole process that describe it step by step.

In the beginning method was tested on dataset of **100** images (10 pictures for each type of butterflies) to reduce compilation time on one hand, and on another hand this quantity of images allow us to calculate approximate accuracy and make final decision about significant for classification parameters. All final result are given

2.1. PRE-PROCESSING

2.1.1. Independence from rotation

Same orientation for all the pictures is needed because we plan to compare shapes. To do this we need to find a line parallel to butterfly (maximum distance between two nonzero points in output segmentation image). Then we found an angle between this line and horizontal line and rotate the image of the butterfly on this angle, the result as Figure 2 shows.

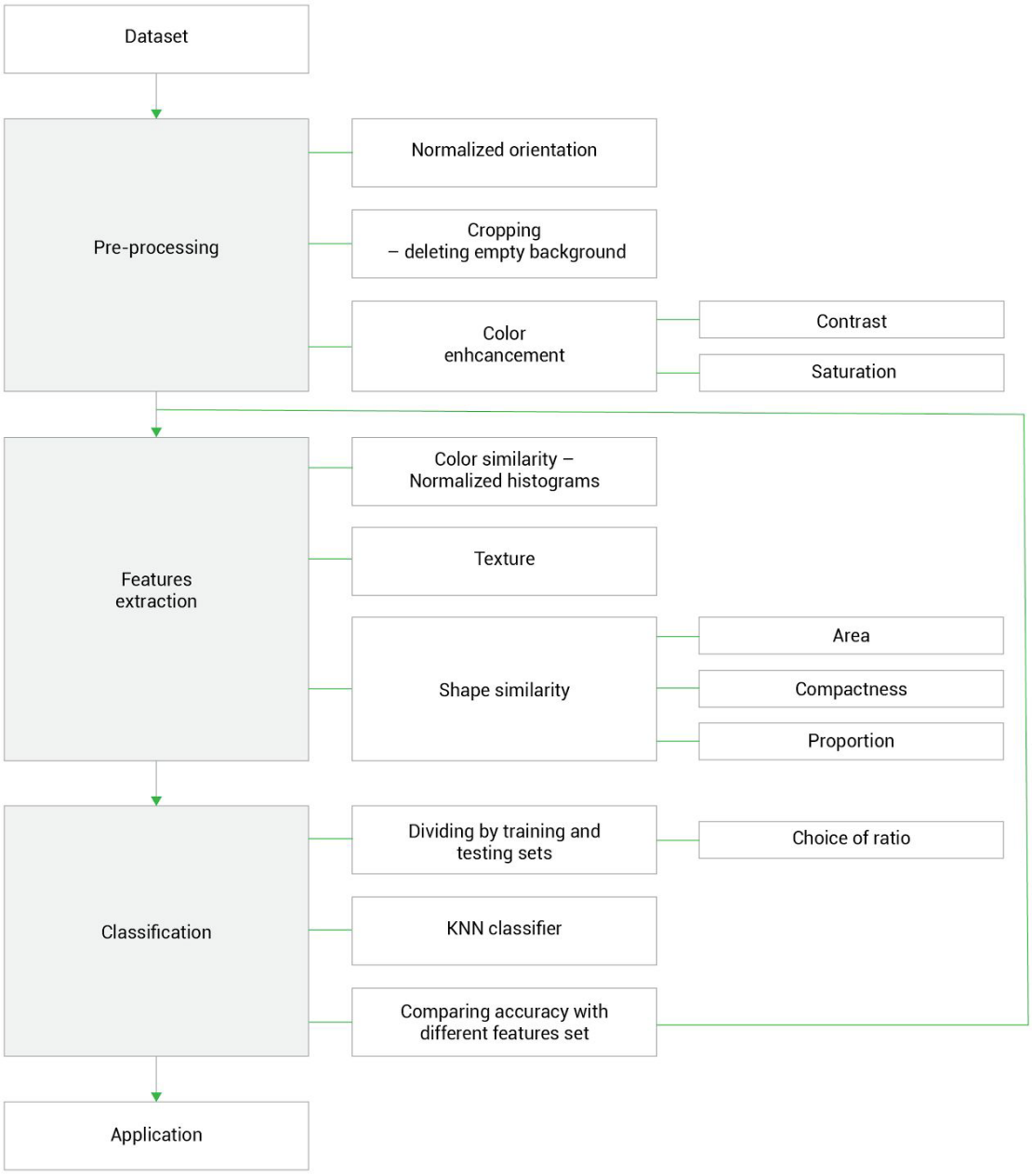

Figure 1. The flowchart of the process

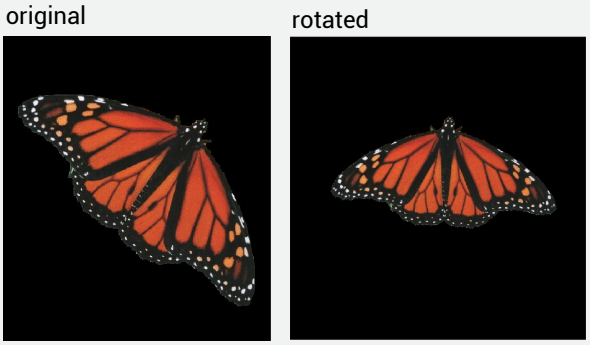

Figure 2. The result of image rotation

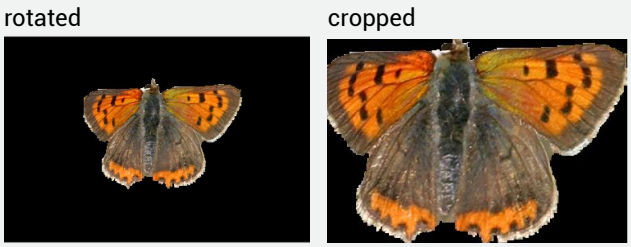

Figure 3. The result of cropping

2.1.2. Cropping

To avoid long calculation zero (black) pixels should be removed. So our images contain only butterfly in width and high (from first pixel to last one). With regionprops function we can find a bounding box containing only white pixels at mask image.

2.1.3. Enhancement of contrast and saturation

For the reason of good color dissimilarity between types images should be color calibrated. Because provided dataset with pictures are taken under different lighting condition we are only able to increase difference between types through contrast and saturation.
Comparison between images adjustment in RGB color space (using "imadjust" function) and HSV color space was made. Second one was chosen as a final option. After several tests we found a correct parameters (saturation index, for example, that in our case 1.2 that provide natural colorfulness for the butterflies) suitable for all the images in Dataset, the result as Figure 4 shows.

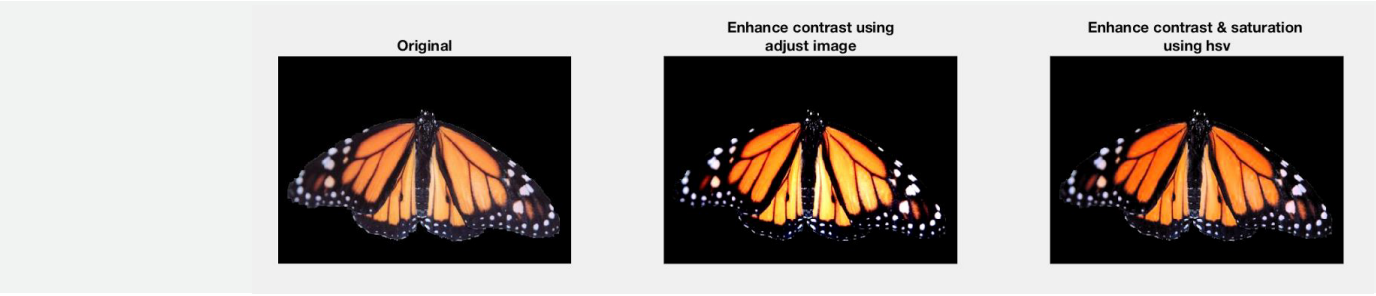

Figure 4. The results of image enhancement

2. Features extraction

2.2.1. Normalized histograms

In this step, each butterfly image with RGB channel will be used to calculate the histograms by 16 bins. The reason we choose 16 bins is because after testing different bins from 256 to 8, we observe that the accuracy will increase while using fewer bins, as Table 1 shows. In this table, we use the same training and test data to do research. When using 16 bins can get the best result, hence we set bins =16 to count histograms. In addition, each image has different scale, to compare the histograms, so we need to normalize it by using formula (1). From normalized histograms, we can obtain 48 features(16 bins * RGB channel).

Table 1. The accuracy by using different bins to do histogram

Training data: 743 images, Test data: 100 images						
bins	8	16	32	64	128	256
Accuracy(%)	73.0	80.0	75.0	73.0	72.0	66.0

$$\frac{each\ bin - min\ bin}{max\ bin - min\ bin} \quad (1)$$

2.2.2. Entropy index

Image entropy can be the richness of image or the information that image has.

2.2.3. Proportion

We know each butterfly type has different ratio of width and height. Accordingly, we calculate the ratio of width and height of each image after image preprocessing, rotation and crop.

2.2.4. Area of the butterfly shape

Since each type butterfly has different size, we can utilize this characteristic to separate 10 types butterfly. We use mask images, and build-in function "find" to calculate the area of butterfly(the number of white pixel value). Also, considering the scale, we have to normalized the area by dividing width and height of the calculated image.

2.2.5. Compactness

From the course, Digital Image Fundamentals [2], we learned different object has different compactness. Even same shape with different scale can still get the similar compactness. Formula (2) shows the equation of compactness.

$$\frac{Perimeter^2}{4\times\pi\times Area} \tag{2}$$

All features we detect from each image are 48 features from normalized histogram, 1 feature from entropy, 1 feature from proportion, 1 feature from area of butterfly and 1 feature from compactness. In total, we use **52** features to classify butterfly.

3. Classification

At final stage we used build-in Matlab KNN-classifier.
A nearest-neighbor classification object, where both distance metric ("nearest") and number of neighbors can be altered. The object classifies new observations using the predict method. The object contains the data used for training, so can compute resubstitution predictions.

3. Experimental results

3.1. Results

We use different ratio of all data set to be training data and the remain images would be test images, the results as Figure 5 shows. We can find the accuracy increases when using more training data. Based on this results, we devote **86%** of all data set (832 images) to be training data and remaining images will be test data depending on how many images that users want to test. Actually, when we run the program, we may get a little different accuracy, since we

randomize the order of images when program reads the images. The training data won't be the same comparing the last time.

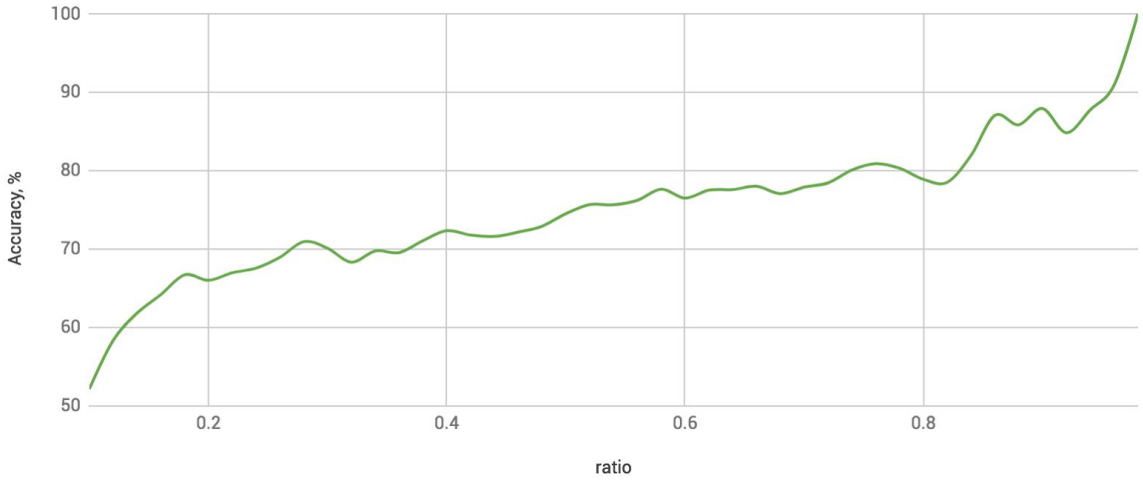

Figure 5. Dependence of accuracy from ratio between size of training and testing sets

Table 2 and Figure 6 lists the accuracy of each butterfly type. We can find that minimum accuracy for 10th type is upper 65% so current algorithm works well for all dataset.

Table 2. Accuracy for each butterfly type

Number of type	1	2	3	4	5	6	7	8	9	10
Accuracy	73.00%	78.57%	100.00%	75.00%	83.33%	90.00%	72.73%	90.91%	88.89%	66.67%

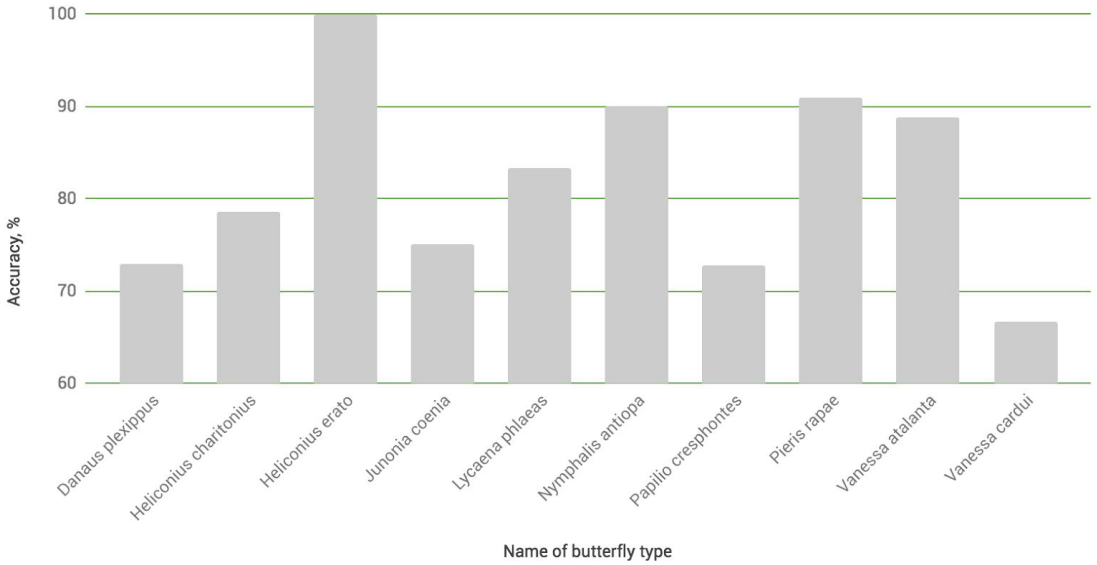

Figure 6. Bar diagram for accuracy of each butterfly type

Figure 7 illustrates how graphic user interface for this project looks like.

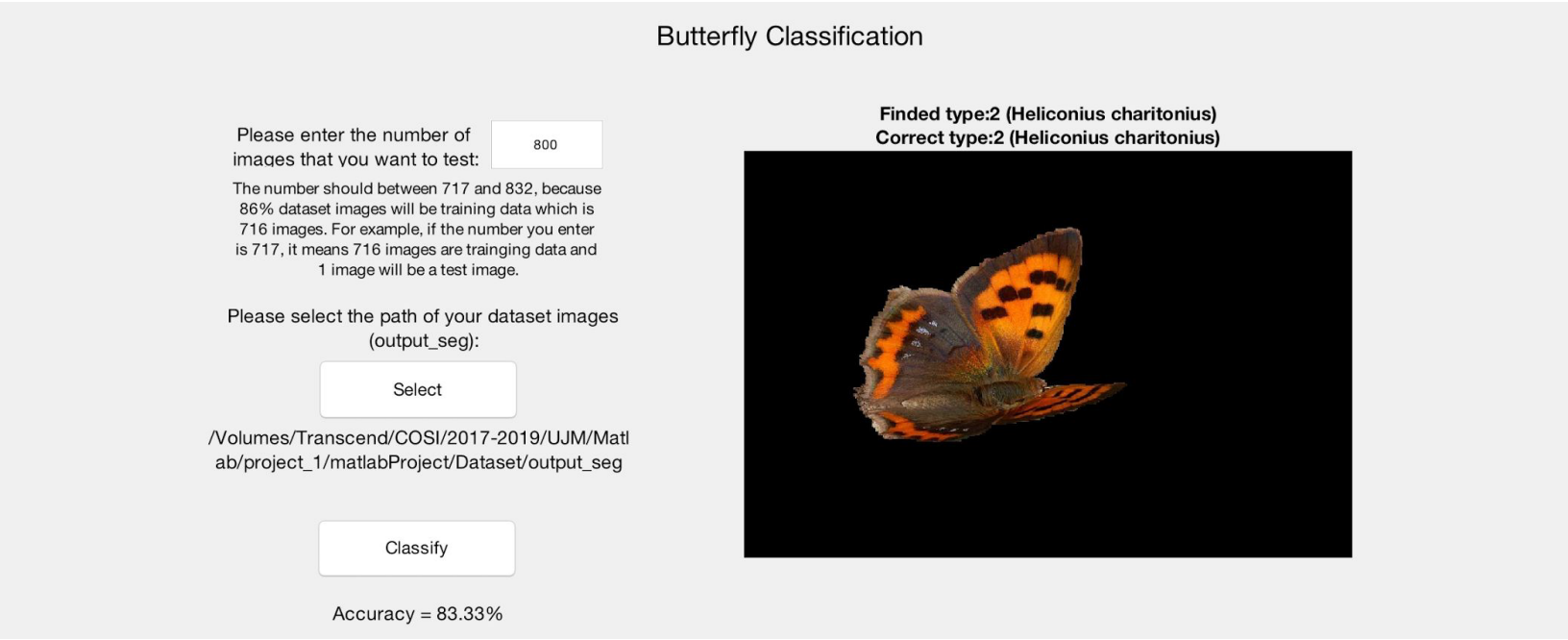

Figure 7. The GUI result

3.2. Remaining bugs

In preprocessing, the results of image rotation are not always correct, like Figure 8 shows. After analyzation, we found the algorithm should detect the endpoints of forewings originally, like yellow points in Figure 8. However, the minimum and maximum points in column direction that algorithm found are the blue points instead of yellow one due to the shape and shooting angle of butterfly. Therefore, that image doesn't rotate to the right direction.

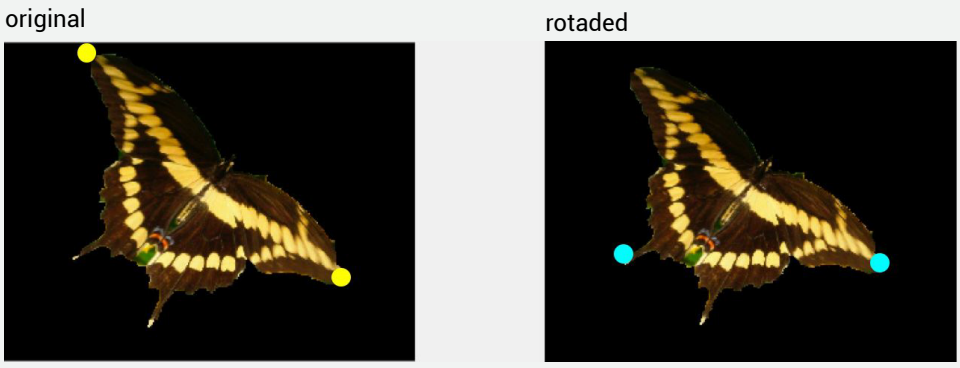

Figure 8. The error result of image rotation

4. MATLAB processing functions

Table 3 contain all the information about functions we create. But there is some additional descriptions below. (Same information you can find at comments to *.m scripts)

4.1. Get Pre-Processed Image

Pre-processing function does preparation of an image for following feature extraction. As input it takes the 3-dimensional array of image, as output it gives two images: first is preprocessed image (which contain normally oriented, cropped and colour corrected picture of butterfly) and second is binary mask the same size (which contain black values for background and white for butterfly shape).

4.2. Get My Feature

Getting Features is function creates a features vector for each picture of butterfly. As input it takes two arguments: image of butterfly (3-dimensional) and its binary mask that we have already calculated in pre-processing function. As output this function returns a vector that contain a features for this particular image. More precisely: normalized histograms for red, green and blue channel (three vectors of 16 elements), entropy index, ratio of width and height of the image, area of the butterfly shape and compactness.

4.3. Classify (KNN)

Butterfly_classify.m script creates a classification model. It contains variable “ratio” that in charge of ratio between size of training and testing sets. The script uses previously created file (my_features_labels.mat) with all features extracted from each picture and number of correct type for them. At the end of execution it saves classification model for future use.

4.4. Accuracy

This script calculates accuracy for whole dataset and independently for each type saving the results in variable “accuracy” and array “RES_type” respectively.

4.5. Best ratio

To find dependence of accuracy from ratio script check different values of ratio in the loop and seves results in the matrix “ACC”.

Table 3. MATLAB processing functions

Process	Description	Input	Output	Matlab built-in function
Pre-processing (getMyFeatures.m)	Create binary mask	Butterfly images (3-dimensional)	binary mask	imerode
	Normalized orientation(rotation)	Butterfly images	Most of images are in the same direction	find, imrotate, atan, rad2deg and length
	Cropping	Binary mask	Images contain only butterfly	Regionprops and imcrop
	Color enhancement	Butterfly images after rotation	enhancement images	Rgb2hsv, imadjust, and hsv2rgb
Features extraction (getMyFeatures.m)	Normalized histograms	Butterfly images after preprocessing (3-dimensional)	Normalized histograms	Find and hist
	Proportion: The ratio between width and height of images	Butterfly images after preprocessing (3-dimensional)	Proportion	We don't use built-in function
	Entropy	Butterfly images after preprocessing (3-dimensional)	Entropy of each image	entropy
	Area of the butterfly	Butterfly images after preprocessing (3-dimensional) and its binary mask	Area	find
	Compactness	binary mask	Compactness	bwperim and find
Classify training data (butterfly_classify.m)	Classify training images by using KNN	Training images' features (my_features_labels.mat)	Results of classification(my_mdl.mat)	fitcknn
Test (accurance.m)	Classify the test images and calculate its accuracy	Test images' features and Results of classification of training images (my_mdl.mat)	Comparing results and accuracy	predict

5. References

[1] Matlab description. https://en.wikipedia.org/wiki/MATLAB
[2] Knonik. H., 2017. Compactness definition. Digital Image Fundamental lecture_7 (DIF_Lecture_7.pdf), pp.14.
[3] Image Entropy. http://www.astro.cornell.edu/research/projects/compression/entropy.html