**German University in Cairo**

**Faculty of Digital Media Engineering and Technology**

**Spring 2017**

**Mini Project I**

**Identifying the Music Notes**

|  |  |
| --- | --- |
| **Experiment** | **Mini Project I** |
| **Submitter** | **Amr Abu Greedah** |
| **Submitter ID** | **25-2284** |
| **Tutorial** | **T-17** |
| **Submission Date** | **2-24-2017** |

The submitted project consists of 17 (.m) files. Input 1, Input 2 and Input 3 music pieces are in the same directory as the (.m) files. The music notes are place in a folder named myNotes, this folder is in the same directory as the .m files.

To run the project type

[strInput1, strInput2, strInput3] = callIt();

strInput1, strInput2 and strInput3 are the output required. They are the translation of music piece to a string telling what keys to be played later.

The next steps are the order of the files created during this project. Each step describing a (.m) file and what its functionality is.

1. myMedianFilter.m (removing noise)

This function takes an inputImg and replaces each pixel by the average of its neighbors.

1. Step1.m (first step in the Otsu method)

Computes the normalized histogram of the input image.

1. Step2 .m(second step in the Otsu method)

Computes the cumulative sums.

1. Step3.m (third step in the Otsu method)

Computes the cumulative means.

1. Step4.m (fourth step in the Otsu method)

Computes the global intensity.

1. Step5.m (fifth step in the Otsu method)

Computes the between-class variance.

1. Step6.m (sixth step in the Otsu method)

Obtains the Otsu threshold.

1. getVarianceGlobal.m

Calculates the global variance which is needed in step7.

1. Step7.m (seventh step in the Otsu method)

Obtains the separability measure.

1. toBinary.m

Changes the grayscale image to binary image. Its inputs are the image and NDM (separability measure) which is calculated in step7. Calculates threshold by multiplying NDM by 255. Then the pixels with values over this threshold are white else black.

1. myMainFunc.m

This function call all the 10 last functions and connect them together in a right way to obtain the noiseless binary image.

The next (.m) files are responsible for obtaining the alphabet of each music piece in the right order. The next steps are going to describe what the functionality of each (.m) file is.

1. callIt.m

This (.m) file is responsible of running the project. First, it calls readImages function which is described later. Second, it calls getNotes function 3 times, one time for each music piece. Finally, the 3 outputs are being saved by calling writeIt 3 times, one time for each music piece.

1. readImages.m

It reads the images from the database, and stores them in a 2d array Im. The first column of this array contains the read image, while the second column has its name. the images are being given to myMainFunc to retrieve their in the binary form then saving them in the same raw as their name.

1. getNotes.m

Its input is a binary image. It is being called from callIt function. callIt function receives an image per time which is a binary divided image.

1. writeIt.m

This function takes two inputs:

1. The read images which are changed to binary.
2. The binary divided piece note.

It calls getKey function and getArrayOfNotes function. Then do some operation to concatenate strings and returns the final output.

1. getKey.m

This function takes the same inputs as writeIt function.

Its job is to tell whether this music piece is a left hand music piece (Treble clef) or a right hand (Bass clef).

In this function we do sum squared difference between the Treble clef image from DB and the first part of the binary divided piece note which is the key.

The same goes for the Bass clef.

1. getArrayOfNotes.m

The last function to talk about. We have 13 notes, and 2:14 parts of the divided binary input image.

So we loop 13 times for each part in the divided image. Calculates SSQ get the min one and returns its index in an array then do the same operation for the 3rd  part of the divided image.