**Experiments: Specification Document**

Student: Luke Agius

Student Number: 080358084

Table of Contents .

**1.**[**Description**](#_1._Description_)**X**

**1.1** [Configuration](#_1.1_Configuration)**X**

**1.2** [Training](#_1.2_Training)**X**

**1.3** [Testing](#_1.3_Testing)**X**

**2.** [**Hard ware specifications used**](#_3._Hardware_specification/Software)**X**

**2.1** [Work station](#_3.1_Laptop)**X**

**2.2** [Cluster](#_3.2Cluster)**X**

**3.** [**Technical Architecture & Code**](#_4._Technical_Architecture.)**X**

**3.1** [Experiment[n].m](#_4.1_Experiment2.m)**X**

**3.1.1** [Flow Chart](#_4.1.1_Flow_Chart)**X**

**3.1.2** [Code](#_4.1.2_Code)**X**

**3.2** [Experiment[n]\_Parameters.m](#_4.2_Experiment[n]_Parameters.m)**X**

**3.2.1** [Flow Chart](#_4.1.1_Flow_Chart)**X**

**3.2.2** [Code](#_4.1.2_Code)**X**

**3.3** [ConfigureOperators.m](#_4.3_ConfigureOperators.m)**X**

**3.3.1** [Flow Chart](#_4.1.1_Flow_Chart)**X**

**3.3.2** [Code](#_4.1.2_Code)**X**

**3.4** [SaveOperators.m](#_4.4_SaveOperators.m)**X**

**3.4.1** [Flow Chart](#_4.1.1_Flow_Chart)**X**

**3.4.2** [Code](#_4.1.2_Code)**X**

**3.5** [TrainOperators.m](#_4.5_TrainOperators.m)**X**

**3.5.1** [Flow Chart](#_4.1.1_Flow_Chart)**X**

**3.5.2** [Code](#_4.1.2_Code)**X**

**3.6** [TestOperators.m](#_4.6_TestOperators.m)**X**

**3.6.1** [Flow Chart](#_4.1.1_Flow_Chart)**X**

**3.6.2** [Code](#_4.1.2_Code)**X**

**3.7** [ReadXML.m](#_4.7_ReadXml.m)X

**3.7.1** [Flow Chart](#_4.1.1_Flow_Chart)**X**

**3.7.2** [Code](#_4.1.2_Code)**X**

**5.** [**Tools**](#_6._Tools) **X**

**5.1** [Rho Viewer](#_6.1_Rho_Viewer)**X**

**5.2** [Result Processor](#_6.2_Result_Processor)**X**

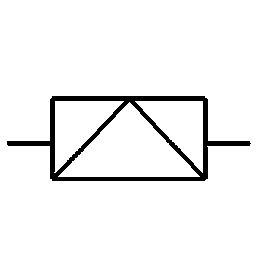
**5.3** [Operator](#_6.2_Result_Processor) Viewer**X**

# 1.Description .

In order to test the efficiency of COSFIRE filters for symbol recognition, a series of experiments will be executed against a number of different data sets with different levels of degradations or invariance under different values for the COSFIRE parameters. This part of the report will detail a very high level description of what this classification does upon using the COSFIRE filter. Each experiment is sectioned into three parts. These parts are the configuration stage, the training stage and finally the testing stage. These three sections are described in finer detail in the following sub sections

## 1.1 Configuration

In the configuration part of the experiment each symbol model image has the COSFIRE filter applied to it. Each application yields an a modified image of the symbol model image, but pin pointing salient features in the image. The following is an example of a simple model symbol and its COSFIRE equivalent.

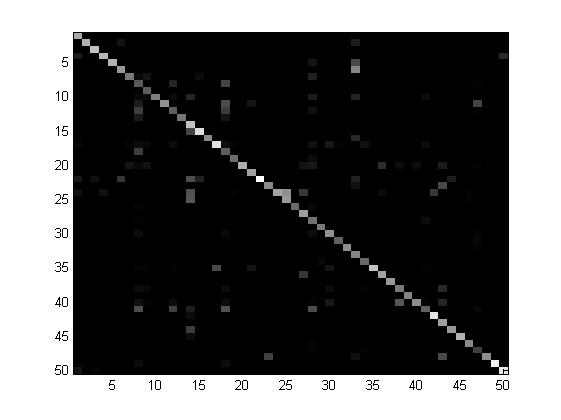
 

The COSFIRE output shown above is merely the representation of the filter. The image above is a visual representation of the filter’s output. In the experiments to be held, there will be different numbers of symbol model images. For each data set, and for each of it’s symbol model images we have to set up a COSFIRE filter. Meaning if a data set has 50 symbol model images, we should have 50 different operators. This step is part of the configuration. The configuration part is a crucial step in the experiment as it is the basis of the experiment, upon which the training will take place. More on training in the next part.

## 1.2 Training

In the training part of the experiments to follow, the same data set of symbol model images used in the configuration part is used, however this time in a different manner. The COSFIRE filters which the configuration part has yielded are used again by applying them again. This time all the filters will be applied to all the symbol model images. This creates a collection of different results from the different operators for each model image. From each of those COSFIRE outputs, the MAX value was extrapolated in order to be put into a vector of size [1 50]. This vector will be the representation of the symbol model image, where each row represents the symbol being used and the columns represent a different COSFIRE Operator output for that same symbol. These results will be put into an excel sheet for later review. The following is a scrap example from a sample excel sheet

The training phase also yields something of particular interest to us. Once the training values have been achieved, these can be plotted to achieve something like the following diagram.



The diagram above is a plot which was also generated from the training values. As one may notice there is a diagonal in the plot. This shows that the Note that the diagonal values are the brightest. This means that for each symbol, it’s corresponding COSFIRE operator had the highest value, hence the diagonal line.

## 1.3 Testing

During the testing stage we are getting the testing data set, which was composed from the model images but this time with different levels of degradation or deformations and we are looping through each test image. For each test image, we are applying the 50 COSFIRE filters again. This yields a representation vector of size [1 50] as well.

Once the 50 COSFIRE operators were applied against the current test image. Each test image is represented by a vector of size [1 50]. This vector is cross checked with the training data and the distance to the nearest vector from the collection of training data is calculated. This distance is calculated in Euclidean distance. The right classification should show up, it will ideally be the row from the collection which resembles the testing image vector the most. Thus ending the classification problem.

# 3. Hardware specification/Software used .

The following is the hardware upon which these experiments are being conducted. This will help in giving an overview onhow efficient the classification is, relative to the computing power available.

## 3.1 Laptop

|  |  |
| --- | --- |
| **Software :** | |
| Operating System | Windows 7 (64-Bit) |
| Software Used | MATLAB (R2012a) |
|  |  |
| **Hardware:** | |
| CPU | Intel(R) Core 2 Duo P8600 @ 2.40GHz |
| RAM | 4GB DDR3 |
| Graphics Card | ATI Mobility Radeon HD 3670 |
| Storage | 250 GB 5400rpm |

## 3.2Cluster

|  |  |
| --- | --- |
| **Software :** | |
| Operating System | Scientific (64-Bit) |
| Software Used | MATLAB (R2012a) |
|  |  |
| **Hardware:** | |
| CPU | 64 processing nodes, with 12 cores on each node |
| RAM | 2TB of Main memory, 32 per node |
| Graphics Card | - |
| Storage | 20TB |
| Benchmarked at: | |
| 6.2 Teraflops (Linpack Benchmark) | |

# 4. Technical Architecture

## 4.1 Experiment[n].m

### 4.1.1 Flow Chart

The following is the main MATLAB script which initiates the process. Tic Toc commands are MATLAB commands which times how much the process takes.

- Set Path to COSFIRE & Gabor functions. - Get Dir objects for model and testing - Sets Set number of symbols

**Configuration**

**Training**

**Testing**

### 4.1.2 Code

function output = ExperimentN()

% Init timer

tic;

% Init Paths, for COSFIRE and GABOR related functions

path('../../COSFIREFilter/COSFIRE/',path);

path('../../COSFIREFilter/Gabor/',path);

% Directories for Models and Testing Images

modelsDir = '../../../Data sets/Symbols Datasets/Sketched Symbols/sketches25f-models/';

testingDir = '../../../Data sets/Symbols Datasets/Sketched Symbols/sketches25f-level1/';

% Parameters

noOfSymbols = 17;

%Configuration of operators

configuration = ConfigureOperators(noOfSymbols,1);

%Populating Training set

trainingSet = TrainOperators(configuration, noOfSymbols,1,0);

%Testing operators

TestOperators(configuration,trainingSet,noOfSymbols,modelsDir,testingDir);

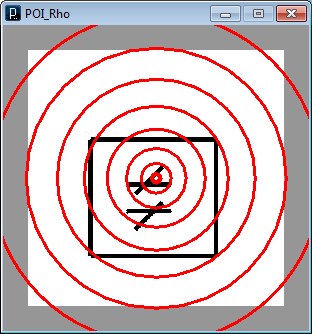
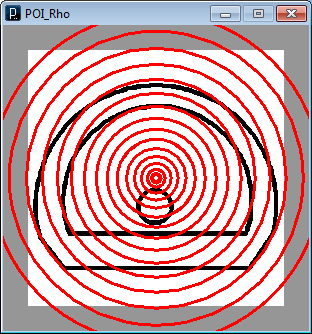
% Display time elapsed.

toc;

# 6. Tools

## 6.1 Rho Viewer

The Rho Viewer is a small application developed with Java Processing. The reason why this tool was developed was to visually present the concentric circles that are going to be used in the filter. This was done primarily to ensure that the whole symbol is covered with the concentric circles or Rho List. Note that some padding (grey borders) was included along with the image so that all 4 corners can be easily seen how they are covered. The code for this sketch is provided in the code base. This will be eventually an output from the MATLAB configuration script.

## 6.2 Result Processor

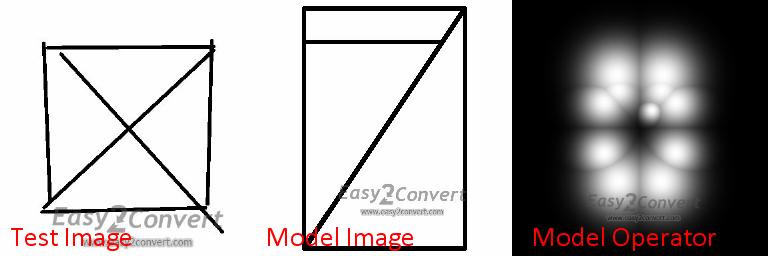
So far the classification output of the MATLAB scripts is in a text file. Therefore in order to process these results automatically instead of going through the results one by one, this small application was developed which parses this text file and extracts the image name for the symbol mode and the image name for the test image. This tool then combines the extracted info to generate an image which shows the results of the classification. The following is a sample of an output file

file\_0.tiff is closest to symbol088.tiff - Pass

file\_1.tiff is closest to symbol088.tiff - Pass

file\_2.tiff is closest to symbol034.tiff - Pass

The following is a test sample of a failed classification. The first image on the left is the test image, the image on the middle is the model symbol the filter thought it was the most close to, and the third and final image is the COSFIRE operator for the model image in the middle.



## 6.3 Operator Viewer

The operator viewer is a small application developed with Java processing. The reason why this tool was developed was to visually present and compare the two different COSFIRE operators generated for each symbol model image. For the experiment which used this tool, two batches of operators have been generated, therefore both batches are being compared next to each other. Note that the extra space at the right of the application is there, since some operators are larger than their originals. The code for this sketch is provided in the code base.

# 