A Comparitive Study of Time Frequency Transforms for Facial Expression Recognition

Kaushik Koneripalli, Ramanathan Raja, Satyajit Giri
Arizona State University

Abstract

This project broadly addresses the problem of feature selection for the purpose of facial expression recognition. The popular methods used for expression recognition in the literature are using the Gabor Energy filters [1], Haar like feature extractor [2] and local binary patterns. This project focuses on an advanced method to perform feature extraction based on the HVS i.e. it uses spatiotemporal Gabor motion energy filter (GME) which will be compared with the Gabor energy (GE) filter and Haar like feature extractor in terms of classification of expression using selected features. Finally, a Support Vector Machine (SVM) classifier is trained using the features from each of these filters to draw a comparison between the cross validation accuracy of the three filters. We observe that the GME outperforms the other two methods.

1 Introduction

Facial expressions are an important component of human communication, and are used to highlight nuances in speech. Understanding facial expressions are an important component in multiple applications including deception detection, monitoring of pain during treatment, empathetic tutoring, monitoring of stress and fatigue, etc. Understanding facial expressions are also of importance to perceptual understanding of AI technology and crucial to improving machine to human interface. Many of the current facial expression recognizers focus on the problem of analyzing the expression in static images. Appearance based discriminative approaches have been proven highly robust for face detection and expression recognition because they analyze things dynamically. However, the feature extraction method incorporated in the human visual system (HVS) is much more sophisticated compared to any current methods as it makes use of spatial and temporal features simultaneously. This project focuses on using the Spatiotemporal Gabor motion energy filter for feature selection. Since the filter is time varing, it captures subtle movements and actions very well.

The Cohn-Kahnade (CK) data set [3] of major facial expressions will be utilised for the project. This database consists of 123 subjects aged from 18 to 30 years who perform a series of 23 facial expressions displays, six of which are the prototypical emotions which are Happy, Sad, Surprise, Angry, Disgust and Fear. Each frame in this data set has a resolution of 640*490. These are video clips of various facial expressions which start from a neutral point (onset) and go to the apex of an expression. The expressions: Happy, Sad and Surprise are made use for the purpose of this project. The images from this dataset are cropped using the Viola Jones algorithm to obtain only the face.

Viola Jones is a face detection algorithm [4]. The Viola Jones algorithm consists of three parts. First, the image is converted into an integral image. An integral image at location x,y consists of the sum of pixels above and to the left of that pixel. The second step is to take a Haar basis function for feature selection. The Haar basis function determines rectangular features and hence are relatively primitive in nature. They give information about the image in only vertical, horizontal and diagonal orientations. The third and final step uses the feature vector to train a cascade of classifiers. The cascade consists of a combination of weak and strong classifiers, the weak classifier was used to select important features and the strong classifier was used to determine if a region belonged to a face or not. Once the face is extracted from the images, they can now be used for feature selection with methods like GE, GME and Haar filters.

The GE filter is a linear filter which is generally used for texture analysis. It analyses whether there is any frequency content in the image in specific directions in a localized region around the region of analysis. GE filters are one of the most successful approaches to represent facial expressions in computer vision applications, including face recognition and expression analysis. The mathematical details of the filter are provided in Section 2.1.

Similar to GE, GME is also a linear filter. In the literature, there are two types of GME filters. The first one is frequency tuned and the second one is velocity-tuned. Frequency-tuned filters have a stationary Gaussian envelope while velocity-tuned filters have moving Gaussian envelopes. Velocity-tuned filters are non-separable, while frequency-tuned filter is spatial-temporal separable. Most neurons in primary visual cortex are frequency tuned, rather than velocity tuned. Hence, this project will focus more on frequency tuned filters. Initially, the research was aimed at studying spatial properties of the receptive field. However, further studies revealed that cortical cells change in time and some of them are inseparable functions of space and time. Therefore, these cells are essentially spatiotemporal filters and they combine information over space and time, which makes a good model for any kind of dynamic movement analysis. The mathematical details of the filter are provided in Section 2.2.

Once feature selection is completed, these features should now be used to train an SVM classifier. SVMs are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of the two categories, an SVM training algorithm builds a model that assigns new examples to either category, making it a binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of separate categories are divided by a clear margin that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the margin they fall. The mathematical details of SVM are provided in Section 2.3.

The organisation of the report is as follows: Section 2 provides the mathematical details of all the filters and SVM which is used for the project. Section 3 provides the overall flow of the entire project. Furthermore, it discusses simulations and results that were obtained. Section 4 discusses the task managenent in the group i.e. division of work amongst team members. Section 5 draws and presents conclusions from the project. Section 6 provides the acknowledgement to the concerned people and resources. This is then followed by the references and appendices where the code made use in the project is provided. This project broadly tries to implement the results obtained in [5]

2 Theoretical Expressions

2.1 Gabor Energy (GE) filter

The GE filter is governed by the equation as given in [6]:

$$g_{v,\theta}(x,y,t) = \frac{\gamma}{2\pi\sigma^2} \exp\left(\frac{-(\bar{x}^2 + \gamma^2\bar{y}^2)}{2\sigma^2}\right) . \cos\left(\frac{2\pi}{\lambda}\bar{x}\right) \tag{1}$$

$$\bar{x} = x\cos(\theta) + y\sin(\theta)$$

$$\bar{y} = -x\sin(\theta) + y\cos(\theta)$$

where:

γ is the parameter that specifies the ellipticity of the Gaussian envelope and is set to 0.5.

σ is the standard deviation of the Gaussian envelope.

$\bar{x}$ and $\bar{y}$ perform the transformation of the coordinate system with the orientation θ.

$\lambda = \lambda_0$ where λ is the wavelength of the cosine factor and the constant $\lambda_0 = 2$ The MATLAB code implementing GE filter is provided in Appendix A.

2.2 Gabor Motion Energy (GME) filter

The GE filter is governed by the equation as given in [6]:

$$g_{v,\theta}(x, y, t) = \frac{\gamma}{2\pi\sigma^2} \exp\left(\frac{-((\bar{x} + v_c t)^2 + \gamma^2\bar{y}^2)}{2\sigma^2}\right) . \cos\left(\frac{2\pi}{\lambda}(\bar{x} + vt)\right) . \frac{1}{\sqrt{2\pi\tau}} \exp\left(\frac{-(t - \mu_t)}{2\tau^2}\right) \quad (2)$$

$$\bar{x} = x\cos(\theta) + y\sin(\theta)$$

$$\bar{y} = -x\sin(\theta) + y\cos(\theta)$$

where:

τ and μ_t are the standard deviation and mean of the time gaussian which is used to model the change in intensities.

v is the phase speed of the cosine factor, which determines the speed of motion.

v_c is the speed which the center of Gaussian envelope moves along the x axis and the other parameters are same as above.

As mentioned in the introduction, there are two kinds of GME filters i.e. frequency and velocity tuned. When $v = v_c = 0$, it is a frequency tuned filter. On the other hand, when v and v_c are $\neq 0$, then it is a velocity tuned filter. The MATLAB code implementing GME filter is provided in Appendix B.

2.3 Support Vector Machines (SVM)

We make use of the SVM to train a classifier with 3 emotions i.e. happy, sad and surprise for all the 3 filters i.e. GE, GME and Haar. The SVMs are primarily used for binary classification problems i.e. differentiate between two classes. However, the similar idea can be extended to multi-class clasification. We make use of the one vs all method where we train 'k' svm binary classifiers corresponding to k classes. In every classifier, one class is taken as the postitve class and the rest are taken as negative classes. With this procedure we end up with 'k' classifiers. Thus for m training examples $(x_1, y_1), ..., (x_m, y_m)$ where $x_j \epsilon R^n$ and $y_i \epsilon \{1, ...k\}$, we have k SVM classes where the i^{th} SVM classifier has the minimisation problem posed as below:

$$\min_{w^i, b^i, \xi^i} \frac{1}{2} (w^i)^T w^i + C \sum_{q=1}^{m} {\xi_q}^i$$

subject to:

$$(w^i)^T \phi(x_j) + b^i \geq 1 - {\xi_i}^m, \text{if} \quad y_i = m$$

$$(w^i)^T \phi(x_j) + b^i \leq -1 + {\xi_i}^m, \text{if} \quad y_i \neq m$$

where x_j is transformed to a higher dimensional space by ϕ, which in this case is a linear kernel and C is the penalty parameter. The basic concept behind SVM is to search for a balance between the regularization term and the training errors. After solving the above posed minimisation problem, there are a set of 'k' decision functions. Finally, the class determined for x by finding that decision function which yields the largest value. Here, we have made use of MATLAB's default implementation of SVM called 'fitcecoc'. Also, we have used the default linear kernel for SVM.

3 Experimental Results

3.1 Flowchart

Fig. 1 provides a flowchart describing the flow of the entire project. The succeeding subsections provide the detailed experimental results of every block in the flowchart.

3.2 Cohn Kahnade dataset

As mentioned in the previous section, CK dataset is a collection of facial expression video clips of a wide range of subjects. An example of the dataset is provided in Fig. 2.

3.3 Viola Jones

This subsection discusses the application of the Viola Jones face detection algorithm onto the Cohn Kahnade dataset. The MATLAB implementation of this, is available in Appendix A.

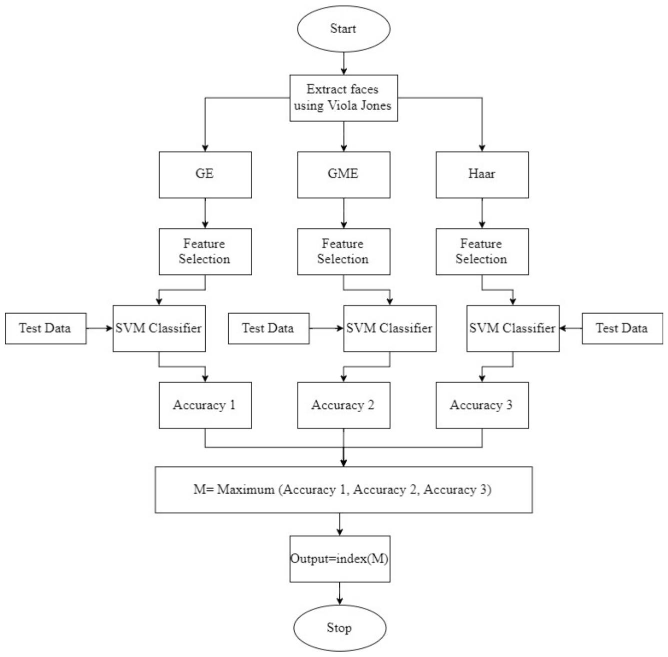


Figure 1: Flowchart of the entire process

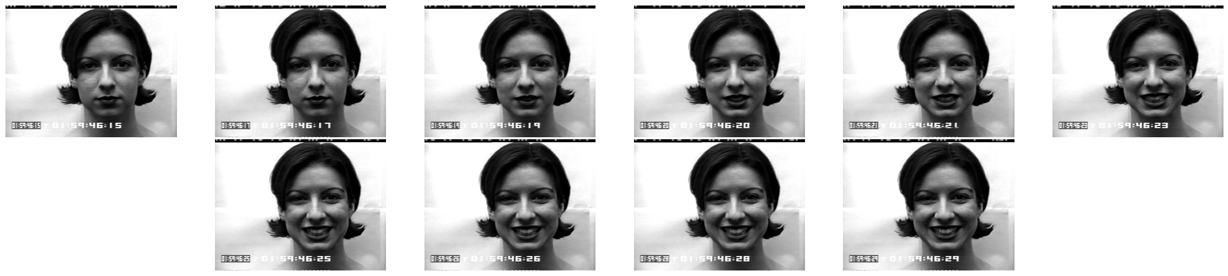


Figure 2: Cohn Kahnade dataset: Video frames of an expression of a subject each of size 640x490

Figure 2 is an illustration of the CK dataset after extracting the faces from the images represented in Figure 1 using the VJ algorithm. Once the faces are extracted using VJ, these images are then resized to 96x96 w.r.t. to the original dimensions of 640x490 of the parent image. This process is repeated for all the examples in the dataset over all the three expressions. Once the concise faces are extracted from the dataset, this data is now ready for feature extraction using GE, GME and Haar filters whose results are discussed in the subsequent sections.

3.4 Gabor Energy filter

The mathematical details of the filter are provided in the previous section. Using these equations, filter of size 3x3 is generated. The filter is varied across 13 orientation(θ) values ranging from

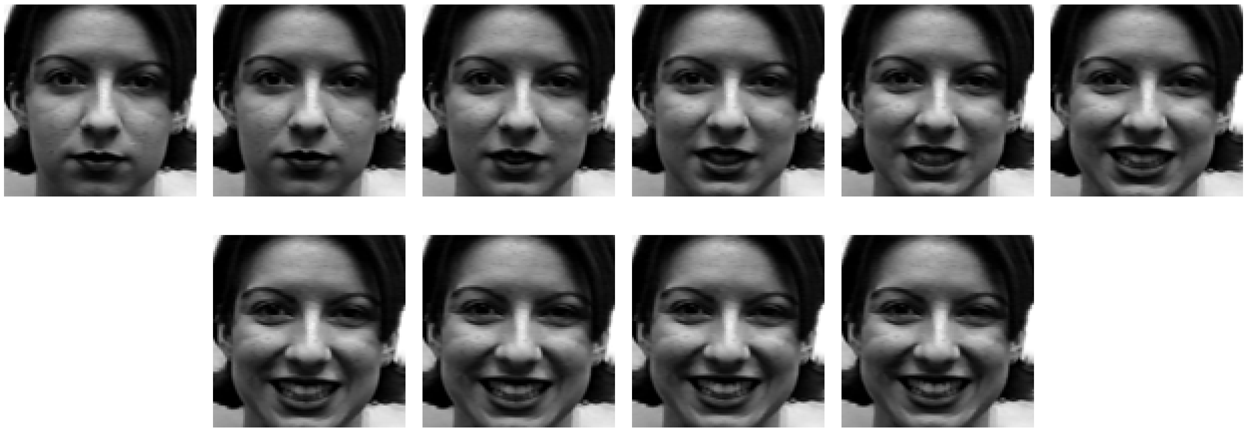
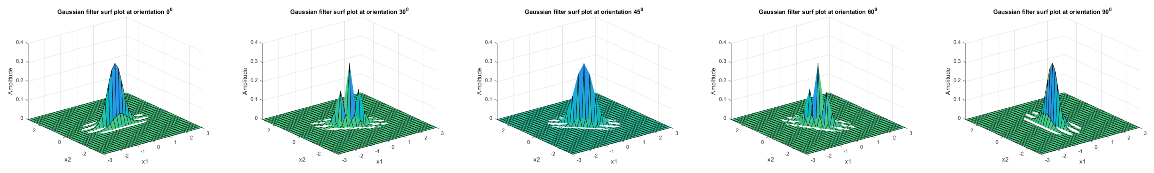


Figure 3: Cropped images after the application of Viola Jones Algorithm of size 96x96

0° to 180° in steps of 15°. Now, we have a set of 13 filter banks corresponding to the 13 orientation values. Each filter bank is now convolved with all the frames of an example. Mathematically,

$$R_\theta(x, y, t) = I(x, y, t) * g_\theta \tag{3}$$

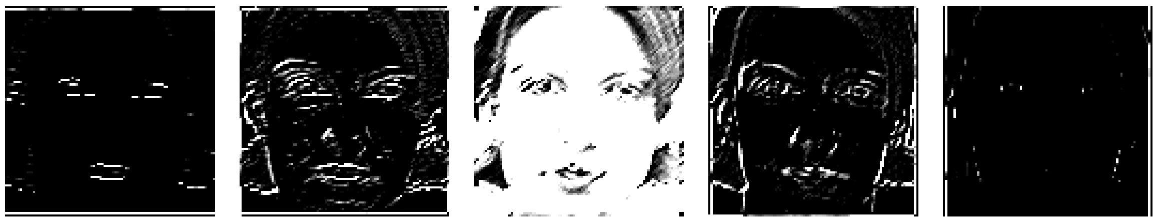
The convolution output of each frame is squared and then added across all frames. The surf plot of the GE filter for different orientations are given in Fig. 4. The convolved outputs of the images for 5 orientations of $0°, 30°, 45°, 60°, 90°$ are displayed in Fig. 5. The MATLAB implementation of this, is available in Appendix B.



(a) $\theta = 0°$ (b) $\theta = 30°$ (c) $\theta = 45°$ (d) $\theta = 60°$ (e) $\theta = 90°$

Figure 4: Surf plots of GE filters at different orientations

From Fig. 5 we can visually notice that the features detected from the face are approximately in the corresponding directions mentioned below them. For ex. Fig 5(c) has features looking like they are in the cross direction which corresponds to 45°. Similar analogy applies to the other figures as well. Finally, the feature vectors are computed for all the examples and are stored.



(a) $\theta = 0°$ (b) $\theta = 30°$ (c) $\theta = 45°$ (d) $\theta = 60°$ (e) $\theta = 90°$

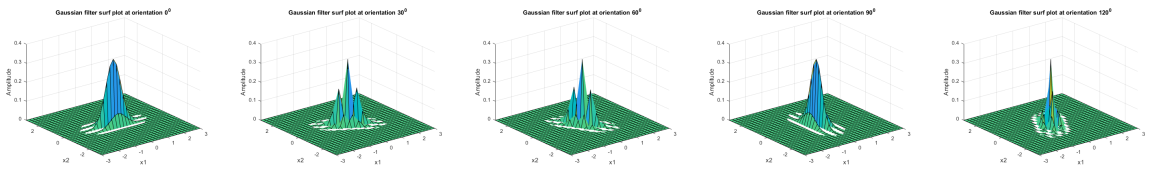
Figure 5: Images convolved with different orientation GE filters.

3.5 Gabor Motion Energy filter

Similar to GE, the mathematical details of the filter are provided in the previous section. Using these equations, filter of size 3x3 is generated. The filter is varied across 13 orientation(θ) values ranging from 0° to 180° in steps of 15°. Here, we assume frequency tuning i.e. $v = v_c = 0$. Additionally, the filter values also vary with respect to time. The time steps in which it varies is taken to be 0.04 seconds. Now, we have a set of 13 filter banks corresponding to the 13 orientation values. Each filter bank is now convolved with all the frames of an example. Mathematically,

$$R_{v,\theta}(x, y, t) = I(x, y, t) * g_{v,\theta} \tag{4}$$

The convolution output of each frame is squared and then added across all frames. The surf plot of the GME filter for different orientations are similar to Fig. 6. The convolved outputs of the images for 5 orientations of $0°, 30°, 60°, 90°, 120°$ are displayed in Fig. 7. The MATLAB implementation of this, is available in Appendix C.



(a) $\theta = 0°$ (b) $\theta = 30°$ (c) $\theta = 60°$ (d) $\theta = 90°$ (e) $\theta = 120°$

Figure 6: Surf plots of GME filters at different orientations

From Fig. 7 we can visually notice that the features detected from the face are approximately in the corresponding directions mentioned below them. For ex. Fig 7(d) has features looking like they are in the vertical direction which corresponds to 90°. Similar analogy applies to

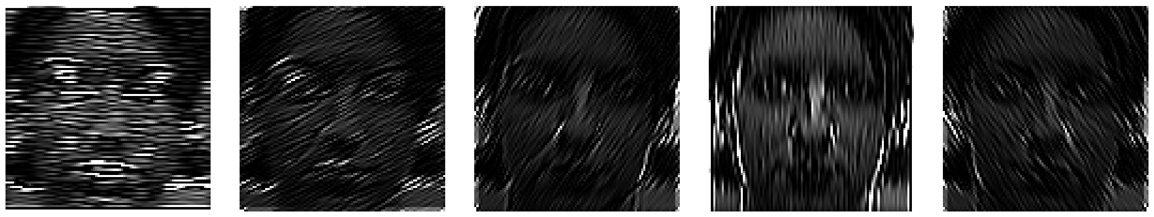

(a) $\theta = 0°$ (b) $\theta = 30°$ (c) $\theta = 60°$ (d) $\theta = 90°$ (e) $\theta = 120°$

Figure 7: Images convolved with different orientation GE filters.

the other figures as well. Finally, the feature vectors are computed for all the examples and are stored.

3.6 Gabor Motion Energy filter with velocity tuning

This follows the same equations as described by the previous subsection on GME. However, in this case $v = v_c \neq 0$. The Gaussian envelope moves with a certain speed described by the value contained in v. Fig.8 shows the surf plot of the filter for v = 1 at different time instants. The values of velocities 'v' to be considered depends on the application. For scenarios where there is a lot of movement e.g. traffic, requires velocity values to be in steps of 1,2,3 etc. However, in this application of expression recognition, there is very less movement in the global sense. Experimentally, we found out that the velocity values that can be used are in steps of 0.4 i.e. 0,0.4,0.8...etc. In a method similar to the one described above, feature selection is performed using a velocity tuned GME. Finally, the feature vectors are computed for all the examples and are stored. The MATLAB implementation of this, is available in Appendix D.

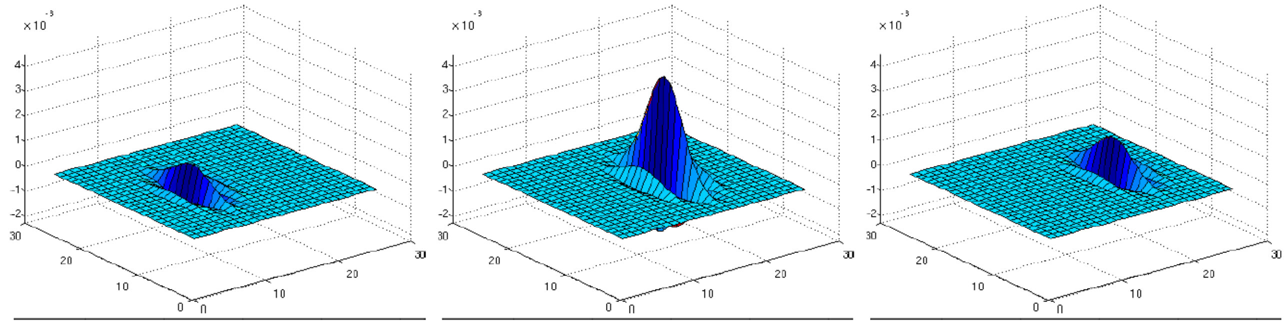


Figure 8: Velocity tuned GME at v = 1 borrowed from [7]

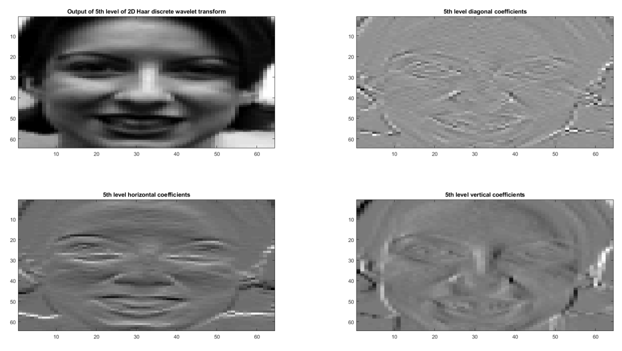

Output of 5th level of 2D Haar discrete wavelet transform
5th level diagonal coefficients
5th level horizontal coefficients
5th level vertical coefficients

Figure 9: Single level Haar wavelet decomposition

3.7 Haar wavelet based feature selection

We use the Haar Wavelet transform in order to generate a feature vector that encodes static information about an image. We utilise MATLAB's 'haart2' transform for 2-d images for the initial transform. The 'haart2' function is MATLAB's approximation to a 2D Discrete Wavelet transform. The Haar wavelet basically decomposes an image into LL, LH, HL and HH subbands. We take the LH, HL and HH coefficients of the first level of Haar transform to get 3 output vectors each of size 48x48. Each of these output vectors are generated and displayed in Fig. 9. Between the three vectors, we can observe that the horizontal coefficients highlight the shape of the eyes and lips. The MATLAB implementation of this, is available in Appendix E.

3.8 SVM classification

The theoretical intuition of SVM has already been provided in Section 2.3. The default implementation of SVM in MATLAB is utilised for this project. Three SVM classifiers for GE, GME and Haar have been trained. Hyperparameter optimisation has been performed on every trained class for about 30 iterations. A graph depicting the cross-validation accuracy vs the number of iterations for the GE filter is given in Fig. 10. With every iteration, parameter tuning takes place and hence the classification accuracy increases. Based on the classification accuracy obtained by SVM, the best feature selection method is opted. We have taken into account the cross-validation accuracy of the SVM and then compared accross the three methods. The MATLAB implementation of this, is available in Appendix F. Table 1 summarises the accuracies that were obtained.

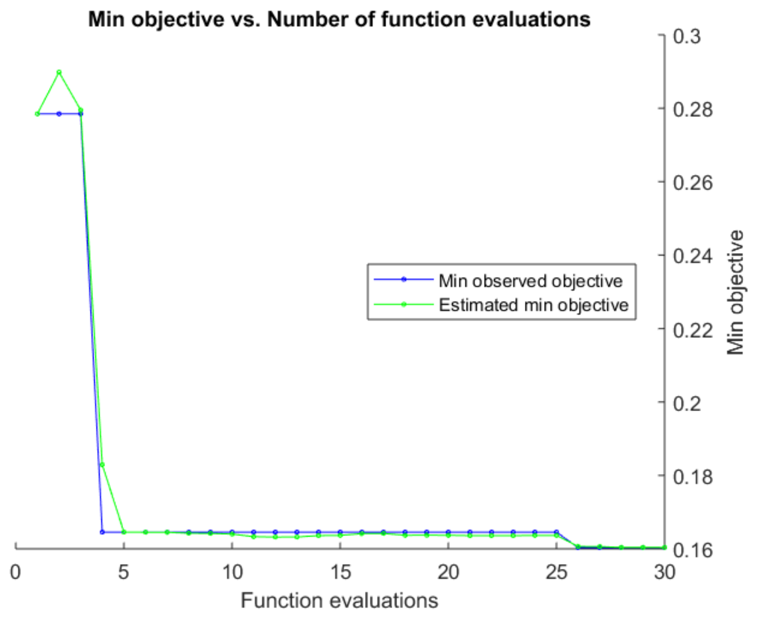


Figure 10: Accuracy vs number of iterations

Table 1: Cross-validation accuracies

Feature selection method	Accuracy(percentage)
GE	84.39
GME	**87.34**
GME with velocity tuning	86.32
Haar	82.67

4 Task Management

Kaushik has worked on the preprocessing of the CK data set. In this project, the selected image sequences are from 100 subjects who depict the emotions: Happy, Sad and Surprise. Firstly, the data set was labelled suitably according to the emotions they represent. To detect the faces in each frame, Viola Jones algorithm has been used. By using the algorithm, the faces were extracted from the images and then resized to 96x96. Resizing was done to ensure that calculating the features will not be computationally intensive. It can have observed that the values of the filter are time varying. The filtered data was obtained by varying the orientations of the filter from $0°$ to $180°$ in steps of $15°$. A feature vector was created by concatenating the energy responses obtained from the filtered data over all the orientations for each expression of the 100 subjects.

The entire feature vector was then stored in a Mat file. Next, the frequency tuned GME filter was implemented by using Eq.(2) mentioned in Section 2.2. Furthermore, the velocity tuned GME was also implemented in a similar fashion. The values of the velocity for the velocity tuned GME were experimentally determined by using the SVM. It was found out that lower velocities gave better results as the velocity of the facial expression is less for the subjects in the CK dataset. In the report, Kaushik has contributed to the theoretical parts of the GME and SVM and the experimental results of Viola Jones and GME filter.

Ramanathan has worked on the preprocessing of the CK data set. As the database contains expressions from 123 subjects, the major three i.e. Happy, Sad and Surprise expressions which are considered for this project are labelled according to the emotions. In order to detect the faces in each frame Viola Jones Algorithm has been used. The detected faces are then resized to a 96x96 in order to reduce the computational complexity. The GE filter is then implemented by using the Eq. (1) mentioned in Section 2.2. Since the GE filters are not time varying, the time dependent gaussian is set to zero. The filtered data was obtained by varying the orientations of the filter from 0to 180. A feature vector was created using the energy responses obtained from the filtered data over all the orientations for each expression of the 100 subjects. The entire feature vector is stored in a Mat file so that they can be easily used in MATLAB. The features vector obtained from the GE filter are used to train the SVM in MATLAB. The SVM is tested using a sample data from the data set and the cross-validation efficiency is found out. After observing the cross validation of the inbuilt SVM in MATLAB it has been understood that the accuracy from the features obtained from the GE filter are the least. It is justified because the GE filter has a stationary Gaussian value which is not time varying and is stationary. In the report Ramanathan has written the Theoretical parts of the GE, parts of Introduction, experimental results of GE filter and Conclusion.

Satyajit was responsible for getting Haar-like wavelet coefficients. The first step was to extract the facial regions using Viola Jones facial detection algorithm. Then he recursively loaded each image database, resized the image to a 96x96 image. Each image was passed through a Gaussian Low Pass filter for removing noise and then image contrast was enhanced by using histogram equalization. He tried converting the image in a binary image, but after using morphological closing, the feature regions started combing and it became difficult to notice changes in shape between adjacent images. Then he used the haart2d transform to get the coefficients for the wavelet. The output was three 48 by 48 matrices resolved by one scale factor. Reducing the scale even further led to a great reduction in output quality where key regions like the eye and mouth became indistinguishable. Then he combined the horizontal, vertical and diagonal coefficients for each image in a sample set by using mean square average. The reason he did that was because the number of images in each sample set was different, making it difficult to get a common length to input to the inbuilt SVM in MATLAB. Therefore, each sample set he finally got an output which was a 48x48x3 vector. In the report, Satyajit wrote the Abstract, part of Introduction, and all parts related to the Haar Wavelet.

5 Conclusions

We have observed that the SVM trained with features vectors of GE, GME with frequency tuned, GME with velocity tuned and Haar filters when tested by a sample data provides cross validation accuracy as illustrated by table 1. We conclude that GME gives the best cross validation accuracy among all the filters in consideration.

The referenced paper [5] considers only frequency tuning for comparison. But, in this project we have considered the velocity tuned GME as well and we have observed that GME with velocity tuning is better than GE and almost close to GME with frequency tuning. By varying the velocities of the GME filter we obtained different accuracies. Each time the accuracy varies depending upon the velocity of the of the object that is considered in the frame. Here in this project since the velocity of the facial expression is less we observe better accuracies at lower velocity values.

6 Acknowledgement

We would first like to thank Dr. Antonia Papandreou-Suppappola, for guiding us by giving ample office hours, helping us select relevant papers, and in general teaching us the basics of research process. We would also like to thank the Affect Analysis Research Group at the University of Pittsburgh for creating and curating the Cohn-Kanade AU-Coded Facial Expression Database which was very instrumental in testing out various algorithms.

References

[1] M.S. Bartlett, G.C. Littlewort, M.G. Frank, C. Lainsesek, I.R. Fasel, J.R. Movellan,“Automatic recognition of facial actions in spontaneous expressions,” *Journal of Multimedia*, vol. 1, no. 6, pp. 22-35, 2006.

[2] J. Whitehill, C. Omlin, “Haar features for FACS AU recognition,” *7th International Conference on Automatic Face and Gesture Recognition*, vol. 5, pp. 101, 2006.

[3] P. Lucey, J.Cohn, T. Kanade, J. Saragih, Z. Ambadar, I. Matthews, “The Extended Cohn-Kanade Dataset (CK): A complete dataset for action unit and emotion-specified expression,” *Computer Vision and Pattern Recognition Workshops*, IEEE Computer Society Conference, pp. 94-101, 2010.

[4] P. Viola and M. J. Jones, “Robust Real-Time Face Detection,” *International Journal of Computer Vision*, vol.57, no.2, pp.137-154, 2003.

[5] T. Wu, M.S Bartlett, J.R Movellan, “Facial Expression Recognition Using Gabor Motion Energy Filters,” *IEEE Computer Society Conference*, pp. 42-47, 2010.

[6] N. Petkov, E. Subramanian, “Motion detection, noise reduction, texture suppression, and contour enhancement by spatiotemporal Gabor filters with surround inhibition,” *Biological Cybernetics*, vol. 97, no.5, pp. 423-439, 2007.

[7] W.N Goncalves, B.B Machado, O.M Bruno, “Spatiotemporal Gabor filters: a new method for dynamic texture recognition,” *Workshop on Computer Vision*, 2011.

Appendix A: Matlab Code for Viola Jones

```
clc;clear;close
faceDetector = vision.CascadeObjectDetector;


% Detect faces.


%Path for where you want to save the cropped image.
file1 = 'E:\cohn-kanade-images\S100\happy_new\';


%Path to access the original image
srcFiles = dir('E:\cohn-kanade-images\S100\happy\*.png');
for i = 1 : length(srcFiles)
    filename = strcat('E:\cohn-kanade-images\S100\happy\',srcFiles(i).name);
    I = imread(filename);
    bboxes = step(faceDetector, I);
    I = I';
    im_new = I(bboxes(1):(bboxes(1)+bboxes(3)), bboxes(2):(bboxes(2)+bboxes(4)));
    im_new2 = imresize(im_new,[96 96]);

    %Note that you have to create a folder 00x_new inside 00x. Otherwise
    %you can simply save it as it is. x=1,2...6.

    new_name = strcat(file1, num2str(i), '.png');
    imwrite(im_new2', new_name);

    %figure, imshow(im_new2');
end
```

Appendix B: Matlab Code for GE Filter

```
clc;clear;close;


%GE filter parameters

theta = 0:pi/12:pi;

gamma = 0.5;

lambda = 2;

sigma = 0.56*lambda;

m = floor(sigma);

%GE filter equation

for t = 1:size(theta,2)

    for x = -m:m

            for y = -m:m

                x1 = x*cos(theta(t)) + y*sin(theta(t));

                y1 = -x*sin(theta(t)) + y*cos(theta(t));

                g(m+x+1, m+y+1) = (gamma/(2*pi*sigma^2))*exp(-(x1^2 +...

                 (gamma^2)*y1^2)/(2*sigma^2))*cos(2*pi*x1/lambda);

            end

    end

    g_new(:,:,t) = g;

end

src = 'E:\cohn-kanade-images\';

j=1;

E_final = [];

for k=3:length(dir(src))

    E = [];

    srcFiles = dir(strcat(src, 'S', num2str(k-2), '\surprise_new\*.png'));

    if(isempty(srcFiles))

%         xlRange = strcat(xlRange1, num2str(k-2));
%           xlswrite(file,E, sheet, xlRange);
        continue;
    end
    I_1=zeros(96, 96, length(srcFiles));
    I=zeros(96, 96, length(srcFiles));
    for i = 1 : length(srcFiles)
        filename = strcat(src, 'S', num2str(k-2), '\surprise_new\', srcFiles(i).name);
        p = imread(filename);
        I(:,:,i) = p;
        %figure, imshow(I);
    end
    for t = 1:size(theta,2)
        c = 0;
        for i = 1 : length(srcFiles)
            I_1(:,:,i) = sqrt(2)*conv2(I(:,:,i), g_new(:,:,t), 'same');
            c = c + I_1(:,:,i).^2;
            %figure, imshow(I_1(:,:,i));
        end

        E = [E;c(:)];
    end
    %I=0; I_1=0;
    E_final = [E_final;E'];
    disp(strcat('Done with', ' S', num2str(k-2)));
    j = j + 1;
end
a = [];
for i=1:size(E_final,1)
    a = [a;'surprise'];
end
```

```
save('surprise_ge_E_final_a.mat','E_final', 'a');
```

Appendix C: Matlab Code for GME Filter

```
clc;clear;close;


%GE filter parameters

theta = 0:pi/12:pi;

%g = size(5,5);

gamma = 0.5;

lambda = 2;

sigma = 0.56*lambda;

m = floor(sigma);

%m = 96;

tau = 2.75;

ut = 1.75;


%STGE filter equation


src = 'E:\cohn-kanade-images\';

j=1;

E_final = [];

for k=3:length(dir(src))

    %E = zeros(1, size(theta,2));

    E=[];

    srcFiles = dir(strcat(src, 'S', num2str(k-2), '\surprise_new\*.png'));

    if(isempty(srcFiles))

        disp(strcat('Done with', ' S', num2str(k-2)));

        continue;

    end

    I_1=zeros(96, 96, length(srcFiles));
```

```
I=zeros(96, 96, length(srcFiles));
for i = 1 : length(srcFiles)
    filename = strcat(src, 'S', num2str(k-2), '\surprise_new\', num2str(i), '.png');
    I(:,:,i) = imread(filename);
end

time = 0:0.04:0.04*(length(srcFiles)-1);
g_new1 = zeros(length(-m:m), length(-m:m), length(time));
g_new = zeros(length(-m:m), length(-m:m), length(time), size(theta,2));
for th = 1:size(theta,2)
    for t=1:length(time)
        for x = -m:m
            for y = -m:m
                x1 = x*cos(theta(th)) + y*sin(theta(th));
                y1 = -x*sin(theta(th)) + y*cos(theta(th));
                g(m+x+1, m+y+1) = (gamma/(2*pi*sigma^2))*exp(-(x1^2 +...
                 (gamma^2)*y1^2)/(2*sigma^2))*cos(2*pi*x1/lambda)...
                    *(1/sqrt(2*pi*tau))*exp(-(time(t)-ut)/(2*(tau^2)));
            end
        end
        g_new1(:,:,t) = g;
    end
    g_new(:,:,:,th) = g_new1;
end
for th = 1:size(theta,2)
    c=0;
    for i = 1 : length(srcFiles)
        I_1(:,:,i) = sqrt(2)*conv2(I(:,:,i), g_new(:,:,i,th), 'same');


    end
```

```
        E = [E;c(:)];

        %E(th) = sum(sum(I_1(:,:,i).^2));

    end

    %I=0; I_1=0;

    E_final = [E_final;E'];

    disp(strcat('Done with', ' S', num2str(k-2)));

    j = j + 1;

end

a = [];

for i=1:size(E_final,1)

    a = [a;'surprise'];

end

save('surprise_stge.mat','E_final', 'a');
```

Appendix D: Matlab Code for GME Filter with velocity tuning

```
clc;clear;close;


%GE filter parameters

theta = 0:pi/12:pi;

%g = size(5,5);

gamma = 0.5;

%m = 96;

tau = 2.75;

ut = 1.75;

vel = [0 0.15 0.3];

%STGE filter equation


src = 'E:\cohn-kanade-images\';

j=1;

E_final = [];
```

```
for k=3:length(dir(src))
    %E = zeros(1, size(theta,2));
    E=[];
    E1=[];
    srcFiles = dir(strcat(src, 'S', num2str(k-2), '\happy_new\*.png'));
    if(isempty(srcFiles))
%         xlRange = strcat(xlRange1, num2str(k-2));
%         xlswrite(file,E, sheet, xlRange);
        disp(strcat('Done with', ' S', num2str(k-2)));
        continue;
    end
    I_1=zeros(96, 96, length(srcFiles));
    I=zeros(96, 96, length(srcFiles));
    for i = 1 : length(srcFiles)
        filename = strcat(src, 'S', num2str(k-2), '\happy_new\', num2str(i), '.png');
        I(:,:,i) = imread(filename);
    end

    time = 0:0.04:0.04*(length(srcFiles)-1);
    m = 1;
    g_new2 = zeros(length(-m:m), length(-m:m), length(time));
    g_new1 = zeros(length(-m:m), length(-m:m), length(time), size(theta,2));
    g_new = zeros(length(-m:m), length(-m:m), length(time), size(theta,2), size(vel,2));
    for v = 1:size(vel,2)
        for th = 1:size(theta,2)
            for t=1:length(time)
                lambda = 2*sqrt(1+(vel(v)^2));
                sigma = 0.56*lambda;
                %m = floor(sigma);
                for x = -m:m
                for y = -m:m
                    x1 = x*cos(theta(th)) + y*sin(theta(th));
                    y1 = -x*sin(theta(th)) + y*cos(theta(th));
                    g(m+x+1, m+y+1) = (gamma/(2*pi*sigma^2))*exp(-((x1+vel(v)*time(t))^2..
                     + (gamma^2)*y1^2)/(2*sigma^2))*cos(2*pi*((x1+vel(v)*time(t))/lambda).
                        *(1/sqrt(2*pi*tau))*exp(-(time(t)-ut)/(2*(tau^2)));
                end
            end
            g_new2(:,:,t) = g;
        end
            g_new1(:,:,:,th) = g_new2;
    end
    g_new(:,:,:,:,v) = g_new1;
end
for v = 1:size(vel,2)
    for th = 1:size(theta,2)
        c=0;
        for i = 1 : length(srcFiles)
            I_1(:,:,i) = sqrt(2)*conv2(I(:,:,i), g_new(:,:,i,th,v), 'same');
            %figure, imshow(I_1(:,:,i));
            %E(th) = E(th) + sum(sum(I_1(:,:,i).^2));
            c = c + I_1(:,:,i).^2;


        end
        E1 = [E1;c(:)];
        %E(th) = sum(sum(I_1(:,:,i).^2));
    end
    E = [E;E1];
    E1=[];
end
```

```
    E_final = [E_final;E'];

    disp(strcat('Done with', ' S', num2str(k-2)));

    j = j + 1;


end

a = [];

for i=1:size(E_final,1)

    a = [a;'happy'];

end

save('happy_stge_vel04_E_final_a.mat','E_final', 'a');
```

Appendix E: Matlab Code for Haar feature extraction

```
%load the source directory

src = 'C:\Users\Satyajit Giri\Desktop\cohn-kanade-images\';

%list of feature vectors

E_final = [];


for i = 1:100

    %find all images in a sample

    imagefiles = dir(strcat(src,'s',num2str(i),'\sad_new\*.png'));

    nfiles = length(imagefiles);

    final_feature_vector = [];

    final_hor = zeros(48,48);

    final_ver = zeros(48,48);

    final_diag = zeros(48,48);


    for j=1:nfiles

        I= imread(strcat(imagefiles(j).folder,'\',imagefiles(j).name));


        %wavelet transform
        [a,h,v,d] = haart2(I,2);
        horI = mat2gray(h{1});
        verI = mat2gray(v{1});
        diagI = mat2gray(d{1});

        %get MSA of results
        final_hor = sqrt(final_hor.^2 + horI.^2);
        final_ver = sqrt(final_ver.^2 + verI.^2);
        final_diag = sqrt(final_diag.^2 + diagI.^2);

    end

    %concatenate the horizontal, vertical and diaganol vectors
    feature_vector = cat(2, reshape(final_hor,[1,48*48]),...
    reshape(final_ver,[1,48*48]),reshape(final_diag,[1,48*48]));
    E_final = [E_final;feature_vector];
    disp(strcat('Done with', ' S', num2str(i)));
end
a=[]
for i=1:100
    a = [a;'Sad'];
end
save('sad_haar_E_final_a.mat','E_final','a');
```

Appendix F: Matlab Code for SVM

```
clc;clear;close;

load happy_stge_vel_E_final_a.mat
mat1 = E_final;
class1 = a;
```

```
clear E_final a;


load sad_stge_vel_E_final_a.mat

mat2 = E_final;

class2 = a;

clear E_final a;


load surprise_stge_vel_E_final_a.mat

mat3 = E_final;

class3 = a;

clear E_final a;


Mat = [mat1;mat2;mat3];

class = [string(class1);string(class2);string(class3)];


t = templateSVM(’Standardize’,1,’KernelFunction’,’linear’);


Mdl = fitcecoc(Mat, class, ’Coding’,’onevsall’,’Learners’,t,’OptimizeHyperparameters’,’auto’,.

    ’ClassNames’,{’happy’,’sad’, ’surprise’},’HyperparameterOptimizationOptions’,...

    struct(’AcquisitionFunctionName’,.’expected-improvement-plus’));


CVMdl = crossval(Mdl);


oosLoss = kfoldLoss(CVMdl)

resubLoss(Mdl)


save(’stge_vel015_model.mat’,’Mdl’,’CVMdl’,’oosLoss’);
```