Internship Project

Lifelogging with Egocentric Images

Author:
Sweta Singh (S2209381)

Supervisor:
Prof. Petkov Nicolai

Coach:
Estefanía Talavera Martinez

August 15, 2016

abstract

Key-point detection is the basis for many object detection algorithms. In this project, we study and investigate the effectiveness of using Trainable COSFIRE Filters in identifying objects (eg. laptops) in a set of given images. The image dataset is obtained by using the Narrative Clip Wearable Camera. Our study shows conclusively that appropriate training of the COSFIRE filters does result in efficient detection of features in the object(eg. laptops). This was achieved by utilizing the multicore architecture of the compute cluster, which was essential to be able to do the processing in real time.

acknowledgements

I would like to express sincere thanks to My supervisor prof. Petkov Nicolai for providing me an opportunity to do an internship under his supervision. My heartfelt thanks to my coach Estefania Talavera, who was always supportive and helpful during the entire project. I am thankful to Dr. V.N.Pandey for his interest in the project and Mr. Martin Vogeloar for letting me use the computing facility at the Kapteyn Astronomical Institute (RUG).

Contents

List of Figures

Chapter 1

Introduction to Lifelogging

1.1 Lifestyle

Today we live in the world where electronic gadgets are a part of our daily lives. A report claims that in the United States of America, out of an adult working population of 139 million, 20% work from home or from remote places at least once a day every month [13]. The sale of gadgets like laptops, smartphones, or tablets has gone up many folds in last 5 years. As per the predictions [4] the sale of smart connected devices would rise by 58.1% from 2013 to 2017, where the sale of smartphones alone would rise by 11%. While the use of these devices increases the productivity at work and gives more scope for entertainment, it also has its severe side effects. Its over use causes postural problems, laptop burns [2] and Repetitive Strain Injury (*RSI*) [10] to name a few. This was one of the motivating factors to take up this study concerning object detection. The results of such studies can be used to statistically determine the time, frequency and the duration for which the laptop was used and can act as a feedback to the patient to correct his lifestyle.

1.2 LifeLogging

Life-logging is described as the automatic recording of a person's everyday activities like walking, eating, or sleeping, by using a digital wearable device. The idea of lifelogging was initially developed to help the patients of Alzheimer's disease which is a degenerative disorder that is characterized by symptoms like loss of memory. Wearable Cameras (small and light weight) could be used as a memory support for people with such problems. These cameras are programmed to take pictures frequently, and store them for reviewing them later at any time of the day. One of the pioneers in the field was *Steven Mann* who proposed the WearCam [12]. Later, the Microsoft Research Center also began using the SensCam [7]. Today we see a larger number of devices capable of image based lifelogging by using wearable cameras. These devices let us capture the incidences that we encounter in our everyday life. There are algorithms that enable the detection of the objects of interest, their location and also provide the time of the images capture, etc. These devices can be worn through out the day, automatically recording images - with no need of involvement or attention from the user. Capturing the moments not only helps in form of maintaining a diary, but also useful for health applications monitoring etc [5].

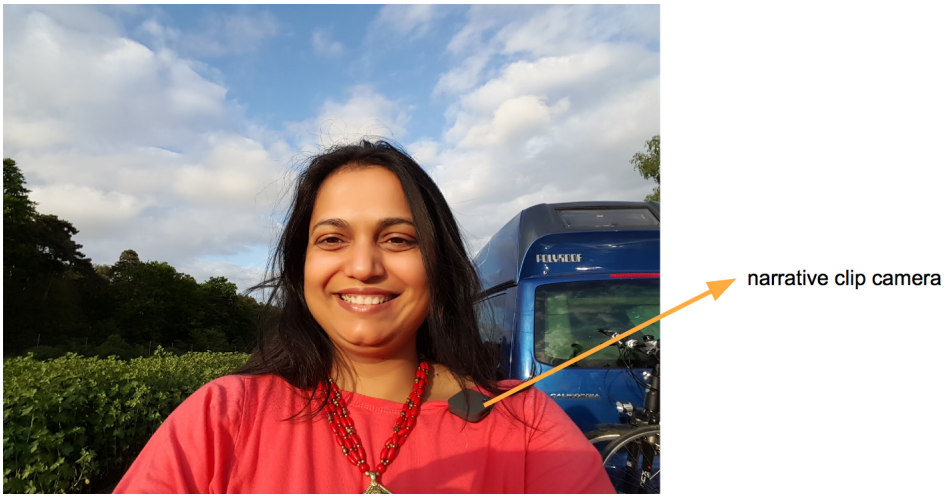


Figure 1.1: Me wearing a Narrative Clip Camera

Figure 1.2: Narrative Clip Camera connected to a laptop

1.3 Recognizing the Lifestyle

For many non-communicable diseases, the lifestyle of an individual is usually associated with the rate of morbidity and mortality [8]. The study of these associations is commonly based on self reporting that could be subjective to person reporting, errors due to inaccurate recalls, etc [9]. Therefore there has been a shift in approach towards objective method of measurements requiring less participant and researcher opinions. Wearable camera technologies are becoming popular research tools in health care. The ill effects of overuse of electronic gadgets like laptop causes Repeated Injury Syndrome inspired me to use Narrative Clip camera to capture the daily activities of the user (me in this case) and recognise the moments when I was using the laptop by using the COSFIRE filters [3]. The results of this can be used to statistically determine the time for which the laptop was used and can be treated as a feedback to the patient that he could use and correct his lifestyle by reducing the use of laptop durations and thereby cure his *RSI*.

1.4 Wearable Camera - Narrative Clip

Narrative Clip [14] is one of the most popular wearable cameras at the moment. It is a light weight device of about 20 gram and has a dimension of $36X36X9$ mm. It can be easily clipped on the user's clothing; like on a hat, coat, around the neck etc. It's camera has a resolution of 5 megapixel,

and captures an image every 30 second. It captures the image with a wide angle arc of 70^o and has aspect ratio of $2560X1920$. An inbuilt accelerometers turns off the camera by facing it down on a flat surface or in complete darkness. It has an on board memory of 8GB that can store about 6000 pictures. Thus one can record images for about 50 hour without the need to transfer it to an external storage device. The images are synchronized with Narrative Cloud service that sorts the uploaded pictures based on location (built in GPS) and time. The battery lasts for about 30 hour and can be charged with a micro *USB*. The inbuilt magnetometer keeps the photo upright regardless of how it has been clipped on to the user. There are four *LEDs* that display the status of memory and battery. In my experiment, I used this wearable camera to log my daily activities. The pictures captured by the camera form the database on which the trainable COSFIRE filters are applied with the aim to detect the objects of interest which is laptop in this study.

In chapter 2, we briefly describe the working method of the Trainable COSFIRE Filters. We explain the principle of the COSFIRE Filters, the mathematics behind, the factors that influence the configuration process, the process of validation and the process of testing the new images for recognition.

Chapter 3 describes the actual experiment that includes the practical work carried out to configure and train the COSFIRE filters,choose the best ones and analyse the result of object recognition when the new images were tested for. This chapter also covers the computational challenges we faced,the process by which we solved them by making the choices of resizing the images as a step of preprocessing, working on cluster, optimising the computation, steps taken for book-keeping and error handling in the code and finally the efficiency we achieved in terms of object recognition and computation.

In Chapter 4 we present the results, evaluation of our work and the conclusions drawn including possible improvements for future works. In this section, I also present the skills I developed in process of my experiments with COSFIRE filters. I humbly accept that my learning curve was very large. In the appendix, the list of codes used in the project are enlisted that is followed by the algorithm and the codes.

Chapter 2

Object Recognition Design

In this chapter we discuss our approach for object detection in lifelogging photographs captured form the narrative clip camera. The chapter explains the various steps involved in the process of recogising the images by using the Trainable COSFIRE Filters. The COSFIRE filters first configure the filters, train them on a dataset and choose the best of the configured filters by validation and finally the chosen filters are used to recognise the object of interest in the test dataset.

2.1 Introduction to COSFIRE Filters

COSFIRE is an acronym for *Combination Of Shifted FIlter REsponses*. This was proposed by G. Azzopardi and N. Petkov in [3], and can be used for keypoint detection and patterns recognition. This method was inspired by the way in which the cortex V4 in human eye recognizes the curved contours. The sub units (tuples in COSFIRE Filters) of the neurons in the human brain are sensitive to different curved patterns and the recognition is calculated by finding the geometric mean of their individual responses. With the same anology, the COSFIRE filters are configured. The responses of these configured filters are based on contours in an image. They are calculated as the weighted geometric mean of the shifted responses of the simpler orientation filters.

2.2 What is a Prototype?

The first step in the process is the selection of a prototype image. By doing this we decide what kind of pattern or objects are we going to identify in a given scenario by using a given algorithm. The selection of an appropriate prototype becomes important because the test image will be compared against it (for features in terms of shape, contrast and texture). When all these components in the test image match with that in the prototype image, the object recognition is considered as successful. This raises another point. The image in test could be in various situations like partially visible, visible under different illumination conditions etc. To address this aspect we need more than one filter where each filter could represent different situations under which the image appears in a given scenario. For the work presented in this report, the aim is to identify a laptop in a set of images. So for the prototype images, we used images containing laptop in various conditions. In each of these images we identified regions (or keypoits) which represent the key features, which are also known as key-points. From each of the selected key-points in the image, a filter is configured.

2.3 Selection of Keypoints

In this phase, the keypoints in an image are defined. These keypoints could be an edge, a corner, keyboard,display screen which may distinguish this object from others in identifying the key characteristics of the object. The trainable COSFIRE filters are configured based on these keypoints. Figure 2.1 shows a few keypoint examples and their zoom-in view. It shows six different examples namely the retina of an eye, the joints in a human knee, face of a woman, an electronic device, a chessboard and the number plate of a car. In the face, corner of an eye is chosen as a key point because it has very specific shape that can be easily distinguished from other features around it like the nose and the eyebrow. The corner of the number plate of the car is also a well defined keypoint with a white colour that can be easily distinguished from it's neighbouring region which is black in colour. Similarly the keypoint at the intersection of corners of the two black boxes in a Chess board describes a unique feature in the region that is distinguishable from its neighbouring regions of white boxes.

Figure 2.2 and 2.3 show examples of a few keypoints from the prototype image in the dataset for the preset work. Figure 2.2 shows the selection of corner of the flap of the laptop as a keypoint as it has a curved edge and a distinct feature of the laptop. The junction of the laptop flap and the keyboard is considered as another keypoint because it is an intersection point of edges with high contrasts. The keys in the keyboard of the laptop is taken as another keypoint. It is very specific in its appearance having small black boxes surrounded by silver colour background. Based on the above, from each of the prototype images, we chose 3 different positions (co-ordinates) as keypoints to configure the filters.

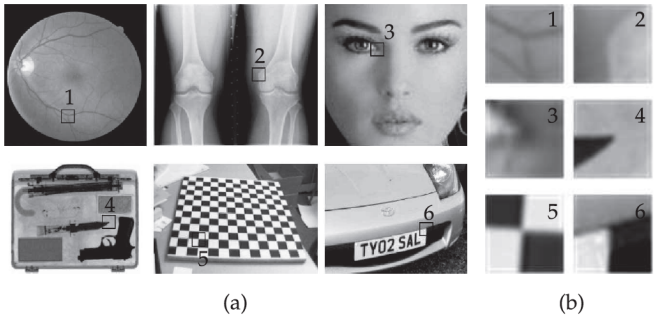


Figure 2.1: Keypoints of an image and their zoom-in views [3]

Figure 2.2: Keypoints of an image containing a laptop

The selection of keypoints is crucial for the configuration of filters to identify similar patterns in

Figure 2.3: A few more examples of keypoints in an image

similar images. The filters designed in this manner are both selective and robust in nature. This will be further explained in Chapter 3.

2.4 Configuring the COSFIRE filters

Once the keypoints are selected, the filters need to be configured. Configuration is the most important phase in this approach as it results in the filter's definition that will help in the object recognition. The quality of the filters decide the performance of the algorithm. Configuration helps to calculate the right combinations of the responses of gabor filters in specific locations that need to be considered for obtaining the final output of the COSFIRE filters. This phase is inspired by the working of Gabor filters.

2.4.1 2D Gabor filters

In the field of image processing, Gabor filter is widely used [1] as it is good in orientation selectivity. Gabor filter forms a good model of visual cortex cells in the human eye. It is basically a product of a sinusoid and a Gaussian with center frequency at zero in the standard form. It behaves like a bandpass filter with preferred wavelength and phaseshift that it responds to. A bank of such filters with different values of wavelength and phase shifts are used to respond to the different regions in the image. The 2D Gabor filters enhance the contrasts in the image whose results help in the process of both manual and automatic image segmentation process [11].

The response of COSFIRE filter $rs_f(x,y)$ is represented by equation 2.1 where $s_{\lambda_i}, \sigma_i, \rho_i, \phi_i(x,y)$ represent the tuples of Gabor filter response. The geometric mean of the Gabor response can be clearly noticed in the equation.

$$rs_f(x,y) = \left| \left(\prod_{i=1}^{|s_f|} (s_{\lambda_i}, \sigma_i, \rho_i, \phi_i(x,y)^{\omega_i})^{1/\sum_{i=1}^{|s_f|} \omega_i} \right) \right| \tag{2.1}$$

$$\omega_i = \exp -\frac{\rho_i^2}{2\sigma_j^2}$$

$$x' = x\cos\theta + y\sin\theta$$

$$y' = -x\sin\theta + y\cos\theta$$

The different variables used in the equation 2.1 are explained as below.

- The number of cycles over each pixel is called as *Wavelength* and is represented by λ.
- θ is the angle of the normal to the sinusoid is called as *Orientation*.
- Phase: ϕ is the offset of the sinusoid is called as *Phase*.
- The Ellipticity of the support of the Gabor Function is called as *Aspect ratio* and represented by γ.
- The standard deviation of the spatial envelope of the Gaussian is represented by σ, that can be varied by varying the bandwidth.
- The bandwidth is represented in terms of λ and σ whose value must be a real positive number and it's default value is 1.

Choosing the appropriate values of these parameters is important in order to achieve good configuration. More on this has been discussed in section 3.2.

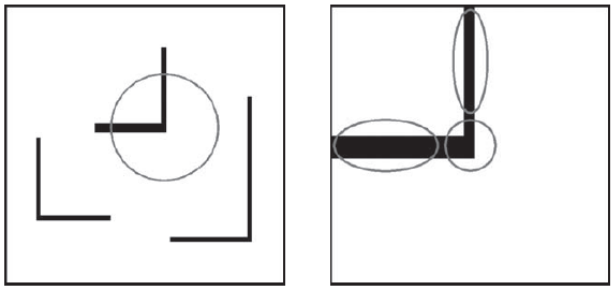


Figure 2.4: Illustration of Point of Interest. In the image with three different features all of which are formed by a combination of a horizontal and a vertical line. The circle represents the area of interest which is to be recognised is shown in Figure (a). Figure (b) shows the presence of three ellipses that will be matched and recognised by the algorithm. [3]

The Gabor responses are thresholded at t_1, a fraction of the maximum response of $r_{\lambda,\theta}(x, y)$ among all the combinations of the (λ, θ) used together with all the keypoint positions (x,y) in the image. The decision on the value of t_1 is based on the kind of images used for configuration. This has been further discussed in chapter 3.

A bank of Gabor filters give responses along different positions (ρ, ϕ) with respect to the center of the filter. To consider some tolerance in the contour parts, the response is first blurred. Then in order to bring them as close as possible to the center, they are shifted. The response of the COSFIRE filter is then calculated as the weighted geometric mean of all the individual blurred and shifted responses.In order to obtain the shifted responses, the corresponding responses at different locations are combined to obtain a bigger response in form of COSFIRE filter. For this computation, instead of the arithmetic mean, the geometric mean was considered for the reason that the curved contours are identified by the activity of neurons by multiplication of responses from the sub-units, that are sensitive to different parts in curve patterns [6]. This helps the COSFIRE filters to produce responses only when most elements of the keypoints are present. It is possible to achieve invariance with respect to scale, rotation and refection. The response of the COSFIRE filter is found by the equation 2.1

2.4.2 Response to the filters

The output response of the Gabor filters is used as an input to the COSFIRE filters. The Figure 2.4 shows an illustration of point of interest [3]. The Figure 2.4(a) shows an image in form of a junction of a horizontal and a vertical line considered as a prototype. The circle represents the feature of interest as chosen by the user. Figure 2.4(b) shows it in an enlarged form where the ellipses represent the *support of line detectors* that were found suitable for this feature in the image. The circle of radius ρ around the point of interest is defined by (ρ, ϕ). The response of the bank of Gabor filters in the circle is considered as an input to the COSFIRE filter. The maximum response of the filters at each of the positions for all the values of (λ, θ) are considered. These responses are compared to their neighbouring values along the radius θ and the greatest of the values are considered as the dominant orientations in that region. This gives us the location of maximum orientation which is translated into polar coordinates. For this location, all the maximum responses are compared for being greater than the thresholded gabor response. Therefore we have a set of tuples defined for each pair of (λ, θ). For each of these pairs there could be more than one tuple.

These responses occur at different locations (ρ, ϕ). Therefore they are first blurred and then moved to the center of the COSFIRE filter. In order to provide the tolerance, the responses are first blurred and then moved by a given distance away from the center. This is now considered for computation of weighted geomentric mean which forms the output response of the COSFIRE filters.These filters respond to the pattern that is similar to the prototype pattern.

Once the filters are configured, the response of each of these filters is checked on all the images used in the configuration phase. This is to check and ensure that the filters responded best to the images they were configured on. This has been discussed in chapter 3.

2.4.3 Self Validation of the Filters

In this process a new set of images are considered. This dataset has images that have the object of interest and also those images which do not have the object of interest in it. The configured COSFIRE filters are applied on these images. The response of these images are compared against the thresholded responses of the filters applied during the configuration phase. Afterwards we verify the result for the ground truth values. If the test image has a response higher than any of the configured filters and it's ground truth values are positive, the images are considered as rightly recognised and are referred to as "True Positive" or "TP" and the wrongly recognised images are referred to as "False Positive" or "FP". The statistical analysis of the TP and the FP is used to discard the filters that give a low value of precision, computed as $TP/(TP + FP)$. This exercise helps us have the best filters that would give the highest responses on the test images and hence completes the validation process.

2.4.4 Testing

The main difference between the validation and testing phase is that

- During the validation phase all the configured filters are used but during the test phase only the best selected ones are used.
- In the validation phase the set of images is not very large while in the test phase the set of images could be large.

The selected filters are used on the test dataset and the object recognition is achieved. This section is further described in chapter 3.

Chapter 3

Experiments and Results

Experiments In order to test the efficiency of the Trainable COSFIRE filters, we realized several experiments. For the experiment We used the images captured by the Narrative Clip Camera details of which are described in chapter1.

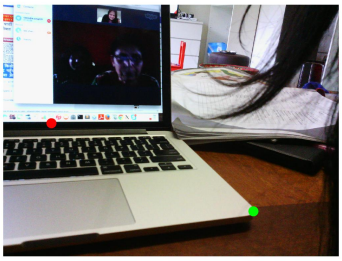
3.1 Preprocessing

As the first step in this experiment, We considered 10 distinct prototype images. On these images, keypoints were to be selected as points of interest. As the images had high in resolution and would require high computation time, we decided to preprocess them by resizing them by a factor (discussed in 3.6). I tried various values to the factor , thereby the generated image was used to generate the tuples. It was observed that if the resizing factor was 0.8, large number of tuples were generated but the processing was very slow. I then tried with 0.6, 0.3 and realised that at 0.3 the images were processed fairly fast and good number of tuples were generated.

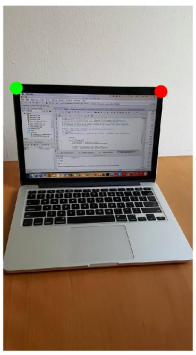
As the next step, the keypoint coordinates were to be selected which would represent distinct features of the prototype image. Manual keypoint selection of the images is quite time consuming and tedious. So I automated this action by writing a code *view_ new_ config_ images.m* that would generate the coordinate values on a click at the position of interest in the image by the user. Automating this process made the selection of points errorless and convenient especially when dealing with larger number of image coordinates. I went through numerous iterations, in order to get the best coordinates that would result in highest responses considering the effects of various parameters on it. The keypoints that I selected in the configuration images are shown in the Figure 3.1. There were two keypoints selected from each of the ten images and have been represented by the spots in red and green colours respectively. It is also to be noted that once the keypoint coordinates have been estimated for an image, they can be used (simply by multiplying with an appropriate number) even if the scale factor of the image under consideration is changed at any later stage.

3.2 Parameters of the Gabor Filters

The principles of configuration have been explained in chapter 2. Several parameters whose values had to be decided upon for a good configuration have been described here, the summary of which is presented in the Table 3.1. We considered the minimum distance between dominant contours lying on the same concentric circle as $\pi/6$.

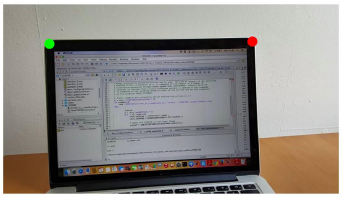


(a) Lap01

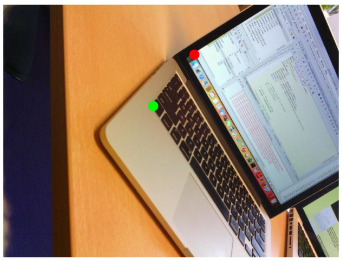


(b) Lap02

(c) Lap03



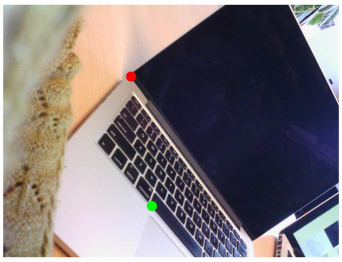
(d) Lap04



(e) Lap05

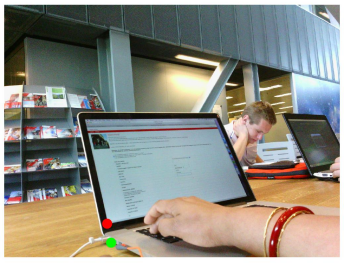
(f) Lap06

(g) Lap07



(h) Lap08

(i) Lap09



(j) Lap10

Figure 3.1: Keypoints of prototype images from (a) to (e). In each image two different keypoints represented by the green and the red spots were selected.

Parameters	values
ρ	[0,8,16,24,32,40,48,52,64,70]
η	
t_1	0.1
t_2	0.3
t_3	0
ψ	$0 : \pi/8 : (2 * \pi) - \pi/8$
θ	$0 : \pi/4 : (2 * \pi) - \pi/4$
γ	0.3
$\Delta\theta$	2
α	4

Table 3.1

It was important to cover the distinctive features of the pattern of the region in the image, so deciding on the number of ρ and their values was important. The size and value of ρ depends on the the nature of pattern of the image. The more complex patterns need larger number of ρ values. Considering more values would be able to capture more details of the region of point of interest that could result in better configuration but at the same time increase the computational cost. The other issue with number of ρ is if the radius is too large it could include the regions that are not important as regions of point of interest. After a few trials of configuration and observing the filters, I decided to use ten ρ values namely (0,8,16,24,32,40,48,52,64,70).

There are three threshold parameters namely t_1, t_2 and t_3. The value of t_1 describes the threshold required to suppress the effect of contrasts and noise in the image under consideration in the response of the Gabor filters. A value of 0.1 is used. Threshold t_2 is taken as 0.75 and this describes that the selected responses are comparable to the strongest response. Threshold t_3 is optional and can be used to suppress the COSFIRE responses to a certain value based on how variable the image can be. For the work in this repot, we have used the value of t_3 as 0.

The filter was used in rotation, scale and reflection invariant mode as the sample object considered was a laptop and the variation in rotational angle, scale or reflection did not cause any change in appearance of the laptop. In order to suppress the noise I used an inhibition factor of $\alpha = 0.04$.

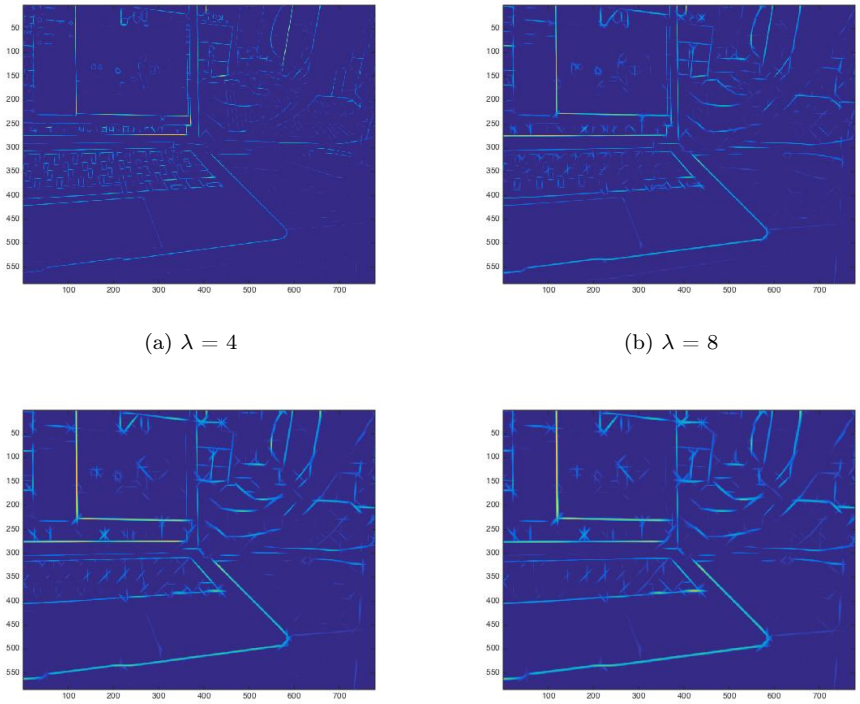
Keypoints and Configuration

In order to understand the principle of the algorithm I spent substantial time understanding the effects of good keypoints, Gabor filters and the tuples that the algorithm generates. For this I changed the values of different parameters, considered different number of images in the configuration phase. As a result I could confidently choose 10 different prototype images for the configuration phase. From each of images, I configured 2 COSFIRE filters constituting the two different distinct parts of the object used for detection (in our case the laptop). Therefore I had 20 COSFIRE filters and their corresponding 20 operators. While choosing the filters ensured to include the keypoints representing the distinct features of the laptop like the corner of the laptop, keyboard, junction between the laptop keyboard and it's flap. As already mentioned in section 2.4.1, the response of bank of Gabor filters is used as an input to the COSFIRE filter, therefore it was important to understand the impact of the various parameters of the Gabor filters.

The effect of some of the parameters that I investigated are explained below.

Effect of λ

The value of λ was varied and observed that with increase in it's value the edges of the prototype images became thicker.This can be seen in the images in Figure 3.2. This also gave rise to emergence



(a) $\lambda = 4$ (b) $\lambda = 8$

(c) $\lambda = 12$ (d) $\lambda = 14$

Figure 3.2: The effect of variation in λ values

of dormant edges which were not important for our object detection process. By comparision of the image at various values, in order to get the least noise and best shape, I decided to assign it a value of 8.

Effect of Aspect Ratio (γ)

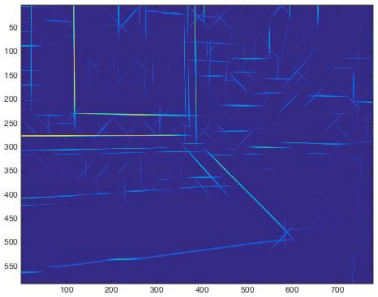
Aspect ratio determines the ellepticity of the image that enhances the visual effects of the image thereby an important factor in image recognition. I experimented with various values of it and decided to use 0.3 for the configuration (see Figure 3.3).

Effect of Bandwidth ($\Delta\theta$)

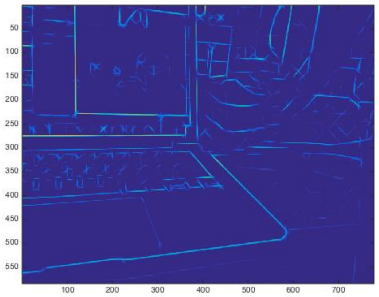
The bandwidth affects the response on dc components of the image which means that in the constant regions, the response of the filter would be determined by the grey scale values resulting in higher responses on higher grey scale values. But we want higher responses on regions like edges or the corners in those constant regions. Therefore to get the optimal value, I experimented with values ranging from 1 to 12 (see Figure 3.4) and settled down at 2 as it gave the highest responses at the keypoints and least at other regions.

Effect of Inhibition Angle

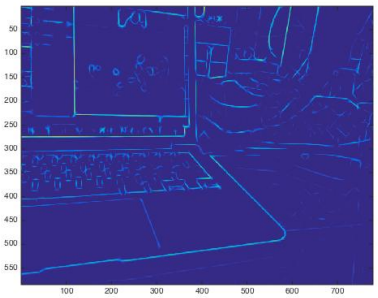
This effects the variation in texture of the image. In order to study this, I considered an image with lot of texture. It was seen that with increase in the value of inhibition factor, the image becomes



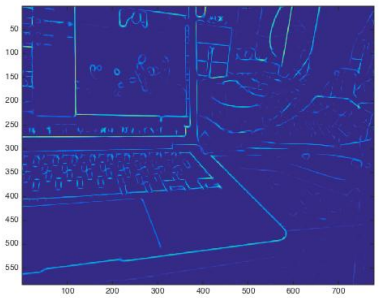
(a) Aspect Ratio = 0.1



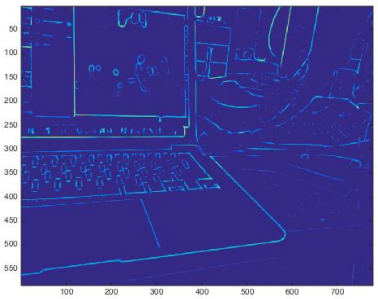
(b) Aspect Ratio = 0.3



(c) Aspect Ratio = 0.5

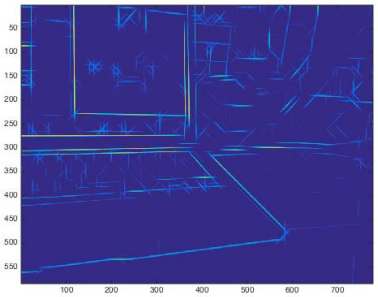


(d) Aspect Ratio = 0.7

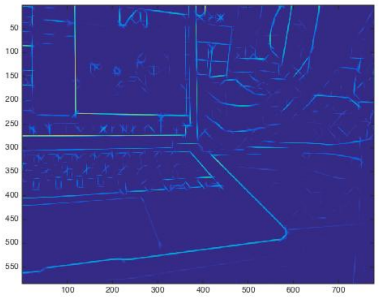


(e) Aspect Ratio = 1

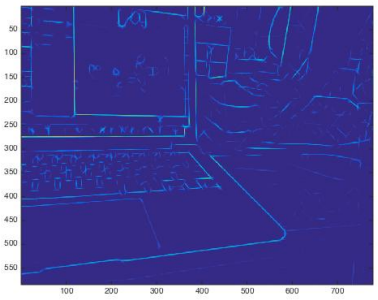
Figure 3.3: The effect of variation in Aspect Ratio values



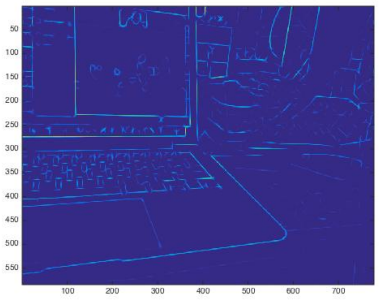
(a) Bandwidth = 1



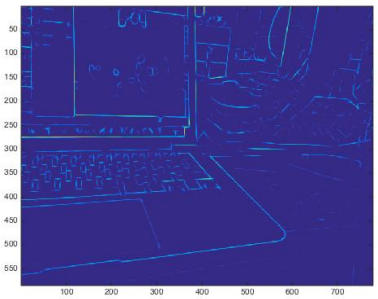
(b) Bandwidth = 2



(c) Bandwidth = 3



(d) Bandwidth = 5



(e) Bandwidth = 12

Figure 3.4: The effect of variation in Bandwidth values

(a) Inhibition with angle 0.02

(b) Inhibition with angle 0.03

Figure 3.5

more even by process of lightening the textures. This can be seen in the Figure 3.5. It can be seen that with a high texture image with inhibition factor being low, there are many maximum responses. As the inhibition factor increases the number of maximum responses decrease. I realised that with the kind of images I have considered for configuration, the most suitable value for this variable was 0.04.

Effect of Inhibition Method

As the images had lot of background noise, it was important to suppress it. I considered anisotropic denoising as it suppresses the region specific noises and preserves the edges of the image considered and therefore it also retains the features. For my experiment we have used anisotropic mode. The Table 3.2 shows the responses of each of the configured filters.

Image	cf1	cf2
lap1	0.1533	0.1131
lap2	0.3174	0.285
lap3	0.0825	0.0645
lap4	0.16684	0.1571
lap5	0.09789	0.07342
lap6	0.0728	0.0875
lap7	0.09643	0.0834
lap8	0.10318	0.07712
lap9	0.0933	0.10099
lap10	0.07254	0.07019

Table 3.2: Maximum responses produced by the ten images considered for configuration is shown in this table. The images are numbered from one to ten and each of them have two corresponding filters named as cf1 and cf2 respectively.

3.3 Training and self validation

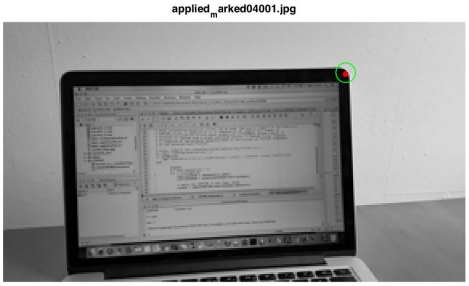
During this phase the same set of 10 configuration images were considered as the training images to test the effectiveness of the 20 COSFIRE filters. This was done to verify if the filters were

good enough in identifying the images they were configured from. If the filters performed poorly on their own images then they should be rejected. If they could not identify the laptop features in the same images they were configured on, then it is very less likely that they will identify the laptop features in the new images. In this process I realised that some of the filters failed this test as the keypoints selected in their configuration were not distinct enough to give a high maximum response at them. I therefore reconfigured the poorer filters and processed them through the self validation test again. This ensured that we are left with only high quality filters that identified all the 10 training images and hence passed the self validation test. Figure 3.6 shows a few examples of how the training images maximum response and the coordinates on which they were configured matched. It is to be noted that the values shown in Table 3.2 for the maximum responses are for the high quality finally selected 20 filters.

(a) Lap10

(b) Lap02

(c) Lap03



(d) Lap04

Figure 3.6: A few examples of the images considered for configuration. The co-ordinates of key-points considered is represented by the green circle. In the self validation phase, the filter was applied on the same image on which it was configured. As can be seen in the images, the key points are correctly recognised as shown by the red spots.

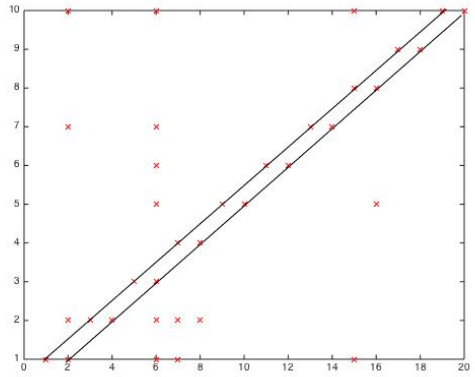
The images in the dataset might have background noises, large variation of textures and lot of other information that might interfere in the results of the COSFIRE filters and hence not produce the actual recognition. In order to take care of this, I experimented comparing the maximum response of the filters on the images with the thresholded responses produced when the filter

was configured. This improved the results of detection by producing a higher number of true positives. But as a result, a few false detections too appeared. By a trade off, I settled to a thresholding value of 70% . Thus if the outcome was greater than or equal to that of 70% the configuration responses (of the filter under consideration), it was considered as a true detection or "True Positive", otherwise it was called false detection or "False Positive". On this basis, the ten configuration images (now called the training images) were tested for their responses against the responses of all the filters designed. If the result of training phase was higher it was assigned a "1" otherwise a "0". This is called as "Self Validation". Figure 3.7 shows the result of self validation. In the Figure 3.7a, "X" represents a "1". It is very clear from the graph that all the filters responded to the same images on which they were configured. Quite a few of the filters also responded to the corresponding patterns in other training images as well. Figure 3.7b is another representation of the same result that helps us visualise the functioning of the filters. Figure 3.8 shows that the TP values are very high for the selected filters in comparison to the FP values generated by them. Figure 3.9 also show that the filters configured from the same parts in the different prototype images produce different number of True Positives that justifies the effect of contrasts, backgrounds etc as factors of influence in the object detection.

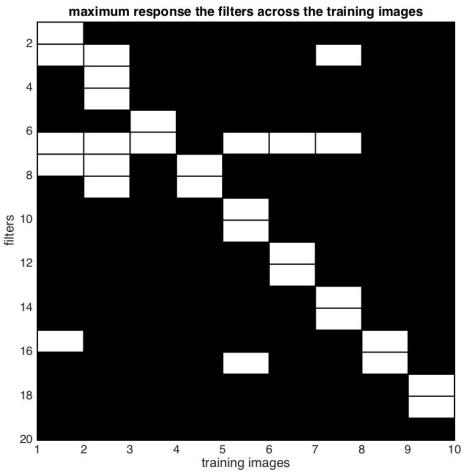
The graphs in the Figure 3.7 basically show the recognition of the keypoints of the images during the self validation phase. When the maximum response of an image to a filter is higher than the thresholded response it had during configuration, it is shown as "X" in Figure 3.7a and rectangle in Figure 3.7b. As each image has two filters, there are twenty filters and ten images. It can be easily seen that the recognition along the diagonals are predominant and complete in both figures. It shows that the keypoints were rightly detected in the images by the filters of their own prototype images.

COSFIRE filter	**TP**	**FP**
cf01	33	0
cf02	91	11
cf03	16	0
cf04	35	0
cf05	18	6
cf06	288	32
cf07	59	1
cf08	51	8
cf09	14	0
cf10	45	0
cf11	5	0
cf12	8	0
cf13	6	0
cf14	1	0
cf15	73	7
cf16	21	0
cf17	8	0
cf18	3	0
cf19	1	0
cf20	12	0

Table 3.3: This Table shows the result of the twenty configured filters where "cf" refers to COSFIRE filter applied to the 410 test images. TP is the true positive and FP is the False Positive. Each COSFIRE filter is numbered against it's TP and FP value explaining that in the datatset, each filter responded to that many numbers of true detections in case of TP and false detections in case of FP. We see that while cf06 results in best detections, cf14 and cf19 detect just one feature. I would still like to retain cf19 because it is the only filter that detects the charging wire connection, and the test dataset has few images with this feature.The same idea explains the presence of cf14.



(a) Plot of the response of the prototype images to the configured filters, where the filters where actually configured from these prototype images



(b) Checker Graph.

Figure 3.7: The graph of the thresholded responses of the COSFIRE filters of training images that were higher than that of the configuration phase responses. The plot shows that almost all filters are responding well on almost all the images

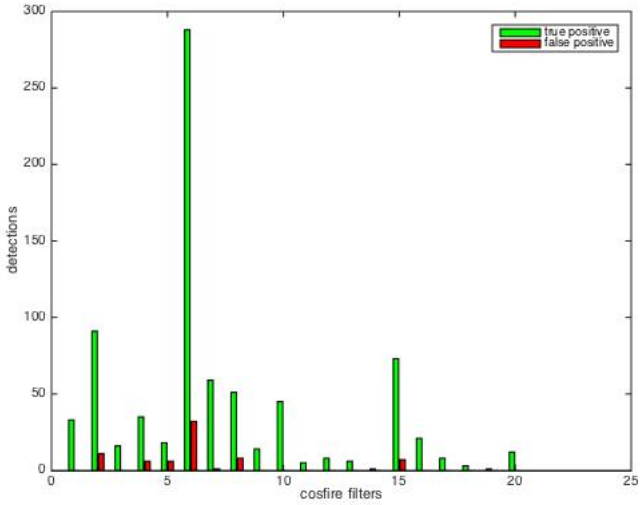


Figure 3.8: The Bar Graph with the TP (in green colour), FP (in red colour) detections as a function of filter number.

The chosen filters are as shown in the Figure 3.9. The filters have been named as "cf" followed by two digits where "cf" stands for "COSFIRE Filter", the first digit represents the serial number of the image from the list of the images considered for configuration and the second digit represents the corresponding filter number (of the two filters) of that image. For example, "cf32" means the second COSFIRE filter that was configured from the third image. As there were two filters configured from each image, the second digit represents the second of the two filters.

3.4 Testing

Narrative Clip camera captures a pictures every 30 seconds. This results in a big amount of information. If the user tends to be static, the images would represent the same environment. The image dataset also consists of several blurred and defective images. The image dataset was cleaned in order to remove them and select images representing different environments in order to have as many different scenes as possible to make the testing of the filters robust. For processing we select 410 random images out of the dataset for my test phase. These images were tested for presence or absence of the desired object by comparing them against the 20 filters (3.3) that we had configured (Table 3.3).

Considering that the test images were in varied situations and their responses might never be the same or very close to that of configuration phase, I decided to threshold the configuration response to 70%. The maximum response of each of the test images was compared against the responses of the configuration phase. The thresholded responses were compared with the ground truth, if they really responded the regions corresponding to their filters. Ground truth means the manual labeling of the images in order to say if it contained a laptop or not. If an image had a thresholded maximum response higher than its corresponding configuration phase maximum response, it was called True Positive. If the ground truth showed absence of laptop but the image showed maximum response in it, it was regarded as a False Positive). If the filters responded positively and their ground truth value was also positive, the result of detection was considered as TP, if the ground truth was negative but the response was higher than the thresholded value of the corresponding configuration maximum response, the result of detection was considered as FP. Because I had 20 filters, each one of which responded to a different number of images, I redefined the TP based on

the number of filters showing high responses. In first type of analysis, I stated that even if one of the 20 filters respond to the test image, and it had a positive result in ground truth, it will be considered as a TP While in the second type of analysis atleast two filters must respond to the test image should be considered. With these values of TP and FP, the performance metrics namely the precision and recall values were calculated. This is shown in the Table 3.4. In my second analysis, I considered atleast the response of two filters as high along with the high value from the groundtruth value as TP. This is seen in Table 3.5

Total Images	410
TP	248
FP	51
TN	85
FN	26
Precision	0.83
Recall	0.905
F1-Measure	0.866

Table 3.4: With at least 1 filter with high response to the test image considered as a High Response. The first column has the metric parameters and the second column has the values computed for these parameters on a set of 410 images.

In this process I reaslise that the filter number 6 had the highest performance and there were two filters namely filter number 14 and 19 that were able to identify only one image but correctly.

3.4.1 Analysis of some special images in the test data

There were few very troublesome images. The images had laptop in them, but there were also many other objects in the scene that had similar properties and hence the images were too complicated for the filters to detct them. Therefore they did not respond to any of the filters. These can be seen in Figure 3.11. As can be seen in Figure 3.11a, the filters always responded to the tyres of the bicycle and never to the laptop itself.This was probably because the tyres had features that responded more to the filters than the laptop did. Figure 3.11b shows an image where in the handle of the home trainer bike responded more to the filters than the laptop didfor the similar reasons. In Figure 3.11c the creeper responded more to the filters than to the laptop.

Total Images	410
TP	148
FP	1
TN	135
FN	126
Precision	0.993
Recall	0.54
F1-Measure	0.7

Table 3.5: With at least 2 filters responding to the test image

3.5 Computing Software, Platform Selection and Technical Analysis

In this section we briefly describe the processing software developed, computing platform selection and technical analysis.

The COSFIRE package already consists of inbuilt functions to configure the filter and apply the filters on an image. It does run in MATLAB environment which is a multi-paradigm numerical computing environment. The matlab version that was used on the for the experiment at RuG facility was:

Matlab version: R2016a (9.0.0.341360) 64-bit (glnxa64) February 11, 2016

We developed a program which can facilitate processing of the images in the pipeline mode. This includes taking the versatile inputs, pass on the appropriate parameters to the functions, have interactive display of the results, and store them in an appropriate well documented manner. The developed software can also handle obvious errors, warn about incorrect inputs and has a progress indicator.

The computing part can be mainly broken into four major parts (not necessarily in order) (a) filter configuration (b) filter application on the image (c) setup of image data base and accessing it (d) setup of the results (including plots) and storing them in a documented way. Step (a) and (b) are compute intensive while (c) and (d) are mainly Input/Output intensive.

We noticed during our analysis that steps (a), (c) and (d) consume less than 10% of the total processing time. Thus it was important to reduce the time taken for carrying out the step (b) i.e. applying the COSFIRE filter on images. For this we first estimated the amount of time it takes on the laptop with an advanced i7-processor. The compute time taken for applying a typical COSFIRE filter on a typical image is as shown in Table 3.6

Scale Factor	0.2	0.3	0.5	0.6	0.8	1.0
Time Taken(sec)	10.4	20.7	50.7	72.8	132.6	210.5

Table 3.6: Time taken for applying a COSFIRE filter on an image as a function of its resizing by a scale factor (thus scale factor of 1 corresponds to no resizing). It can be noticed that the time taken for applying the COSFIRE filter increases significantly as the scale factor is increased. The processor used for carrying out the computation was Intel(R) Core(TM) i7-4558U CPU @ 2.80GHz. Only one physical core was utilized.

Thus the time to carryout this exercise on a test data set of 410 unscaled images, with 20 COSFIRE filters would be ≈24 days. To fine-tune the filters and parameters, it is most likely expected that this process would be attempted a few times (repeated) during learning about the working of the filters. Thus it was clear that we cannot process such and bigger data set sizes in realistic times without (a) reducing the compute time in application of the COSFIRE filter and (b) increasing the number of cores to accomplish the task in multi-core parallel programming mode in Matlab.

We experimented with various scale factors as discussed in 3.6. Finally we found a scale factor of 0.3 to be the most optimum without loss in performance and quality of the filters.

As the configuration phase is not very compute intensive and also involves a lot of image inspection in detail, I used laptop to configure the filters. The specifications of the laptop are as below.

Model Name: MacBook Pro
Operating System: Yosemite
Processor: Intel Core, (TM) i7 4558U @2.80GHz

Number of Processors: 1
Total Number of Physical Cores: 2
Memory: 16 GB
Graphics : Intel Iris 1536 MB

For testing phase we needed more compute capacity. In order to get access to multi-core computing facility, I contacted Dr. V.N.Pandey of Kapteyn Astronomical Institue (RuG). He was interested to see the working and performance of these filters. At a later stage he may like to investigate if these filters may be applicable for Astronomical data (e.g. Galaxy classification in images). Once we had access to some of the compute facilities at the Kapteyn Institute, we thought it would be interesting to see how different processors in different compute clusters performed compared to each other.

A single COSFIRE filter was applied on a single image (resized by a scale factor of 0.3). The results of the exercise are as given in Table 3.7. It can be easily noticed that the version 3 processors easily outsmart the version 2 processors. The only processors which perform slightly faster the version 3 processor i-7 due to the fact that they are quite advanced ones and have a high clock rate. But since the number of cores on the host computers having the i-7 processors is limited, in a multicore environment, the bigger hosts would complete the job on the big data sets quicker. Of the hosts available, I chose to use norma4 to run my processes due to the fact that (a) it has a fast processor (b) the number of physical cores are 32 (c) the total available memory is 768GB (d) there is HDD capacity of more than 7TB (e) This was not being used for big projects so was easily available for my work.

Processor type	**Time taken (single physical core)**	**No. of physical cores (one node)**
Intel(R) Xeon(R) CPU E7- 8850 @ 2.00GHz	42.00s	80 (hathor)
Intel(R) Xeon(R) CPU E7-8850 v2 @ 2.30GHz	37.10s	96 (argo)
Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	20.76s	16 (gaia)
Intel(R) core(TM) i7-4558U CPU @ 2.80 GHz	20.70s	2 (laptop*)
Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz	20.02s	24 (dawn cluster)
Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.30GHz	19.79s	32 (norma4)
Intel(R) Core(TM) i7-3770S CPU @ 3.10GHz	18.26s	4 (horrocks)

Table 3.7: Comparison of average time taken by the different processors for a single application of the COSFIRE filter on an image (using only one core). The images have been resized by a scale factor of 0.3. It is to be kept in mind that there is no significant effect of any other factor of the host computers as the process is compute intensive and the impact due to the differences in the Input/Output performances is insignificant. It should be noted that while the processors allow hyper-threading but Matlab does not use it by default. Forcing it to use hyperthreading does not improve the performance in our kind of application.

The specification for Norma 4 are listed below:-
Compute node name: norma4 (in kapteyn Institue, RuG)
Operating System: Scientific Linux release 7.2 (Nitrogen)
CPUs available: 2 x Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.30GHz;
Number of cores: 32 (2x16)
Main Memory: 768GB memory
Hard Disk Drive: 7.1TB

So we used the *parfor* construction in our matlab code to utilize the multicore architecture of norma4 compute cluster. The program calculates automatically the ideal number of workers which should be utilized depending on the scale of the problem so as the time taken is minimum. Generally we used between 29-31 cores (depending on the problem size) for the workers. One core was always left for the main matlab executable to run. It also ensured that the load on the system is not too

high.

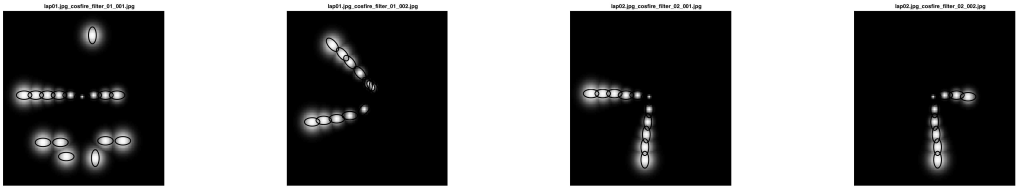
3.5.1 Multi-Core Efficiency

It is important to know the net speed up by using the multi-core CPUs. Let us look at an example below produced by an actual run:

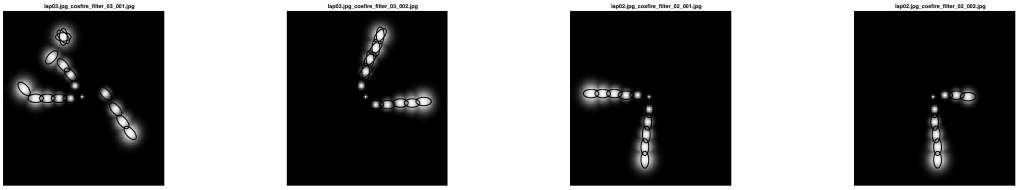
No. of Images: 410
Scale factor: 0.3
No. of Filters: 20
No. of applications: 8200
No. of cores used: 29
Starting time: Mon 11 Jul 11:11:54 CEST 2016
Ending time : Mon 11 Jul 13:25:27 CEST 2016
Total time taken: 8013 seconds.

Avg time for single COSFIRE application: 8033/8200 = 0.98s (Multicore mode)

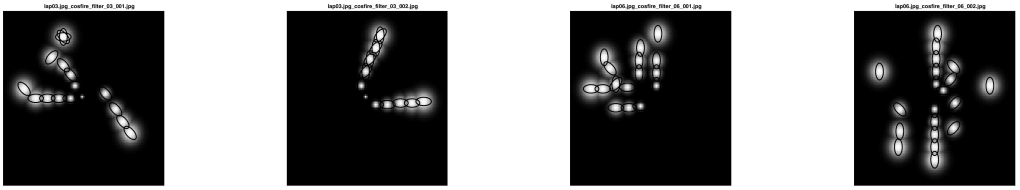
We know that without any multicore, a single COSFIRE application takes about 19.79s (See Table 3.7). Thus the net speed up is ≈20.25 which gives the multicore efficiency as 69.82%. Hence the effective speed up is about 70% of the number of physical cores used.



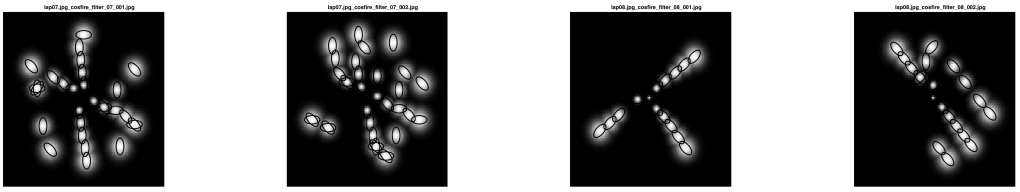
(a) cf11 (b) cf12 (c) cf21 (d) cf22



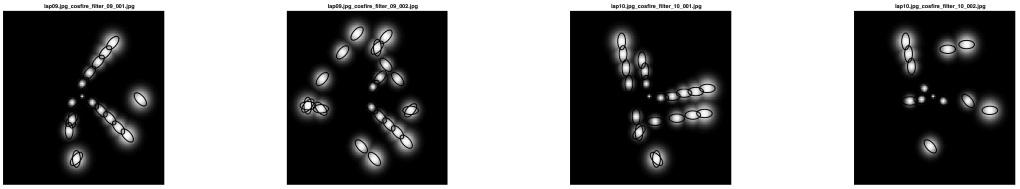
(e) cf31 (f) cf32 (g) cf41 (h) cf42



(i) cf51 (j) cf52 (k) cf61 (l) cf62



(m) cf71 (n) cf72 (o) cf81 (p) cf82



(q) cf91 (r) cf92 (s) cf101 (t) cf102

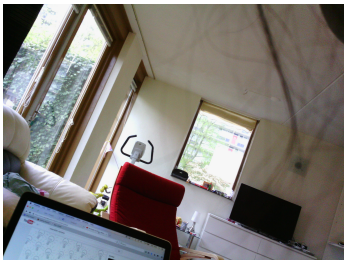
Figure 3.9: The 20 COSFIRE Filters configured from the 10 configuration images of the Figure 3.1. The figures show the tuples generated around the keypoints. The shape formed by the group of tuples show us the kind of regions that they would be matched with during the recognition phase of the test images. For example Figure 3.9b would recognise the corner of a laptop and it actually resembles a "V" shape and similarly Figure 3.9c forma a "L" shape and resembles the corner of the edge of a laptop.

(a) TruePositive

(b) False Positive

Figure 3.10: Examples of True Positive and False Positive

(a) Bike behind the laptop



(b) Home Trainer behind the laptop

(c) creeper behind the door

Figure 3.11: Examples of complex images not responding to the filters

Chapter 4

Conclusion

This chapter summarises the goals, results and evaluation of my experiment to check the effectiveness of Trainable COSFIRE Filters on the images captured by Narrative Clip Camera. It also discusses the scope of future works in relation with the Trainable COSFIRE Filters.

4.1 Project Goals

The goal of this project was to study the effectiveness of the Trainable COSFIRE filters in Object recognition. The test images were successfully recognised by the COSFIRE filters with a high precision. For this assignment, the coding language used was Matlab with multicore functionality.

4.2 Results

Configuring the best filters which was supported by validation process as discussed in Chapter 3, was followed by testing. In the process of testing, it was seen that the results of performance could be improved by thresholding the response of training process. But one has to be careful in not thresholding it too much otherwise there will be many false detections. The recognition process is defined based on the choice of filters and the number of filters being considered. The idea is that there are some features present in very few images and they would be recognised only by a specific filter. Let us consider the filter "cf19" recognising the charging wire connecting the laptop. Now in the test data, this kind of image present is just one and it was recognised correctly by "cf19". Statistically only one filter recognised it. If we decide that the image should be at least recognised by two filters, then this image goes unrecognised and therefore the TP, recall and F1 measure values would be lower whereas the precision would be higher. But if we decide to consider that each test image should be recognised at least by one filter, then this image would be rightly recognised and hence increasing the TP, recall and F1 measure values but decreasing the precision. This can be seen in Tables 3.4 and 3.5.

4.3 Evaluation

Number of filters

After identifying the specific area of interest, we considered 15 filters configured from 5 different images. On running it through the configuration and validation phase we realised that these five

images were not enough representations of the kind of images we were looking to detect. We therefore raised the number of images to 20 and now had 60 filters configured. This was high on computation, and therefore we decided to adapt our code for multicore programming which is explained in the section on Computational Aspects. A closer study of the configured filters by iterative process of training and validation helped us bring down their number to 20. These 20 filters were a good representation of all the key features in most of the images.

Dataset

Our dataset included many images images that were blur, dark and not unique (representing the same environment). Therefore, cleaning them up was necessary (see Section 3.4). This was quite time consuming. Therefore we decided to experiment on 410 good test images.

Computation cost

The computation cost of configuring these filters is directly dependent on the number of ρ values, number of orientations and image size. The resizing factor was closely studied (see Table 3.7) as too much reduction in size would result in loss of information in the image and hence would effect the results of object recognition. As a result it was decided that 0.3 would be a good scaling factor to retain the properties of the image and at the same time not being too compute intensive. Studying the dataset that we had, shows that capturing the keypoints in the given environments needed lot of information and therefore various and large number of ρ around the keypoints was necessary. As the laptop has a definite shape, a few orientations of it were enough in representation. By experimenting with their values, we were satisfied with keeping 6 values in 8 orientation. On 2.8 GHz Intel Core i7 processor on laptop with Matlab implementation, on image resolution of 8 MP being resized to one third of it's value, ten values of ρ and sixteen rotations, it took about 3.9 second to configure each filter. Whereas on the same machine, with the same image sizes as before if six ρ values, eight orientations were used, it took 3.35 second to configure one filter. Therefore we decided to use the latter combination of values.

Skills Learnt

As already mentioned, high computational speed and a good level of programming were very important aspects for realisation of this project. This motivated me to learn multicore programming using Matlab. I became proficient in advanced Matlab usage including running the scripts remotely using grid screen. We also learnt and utilised making plots and graphs in the non visual mode straight into the files without actual display on the screen. This was essential for remote processing. We successfully used the 32 cores of Norma4 machine. My knowledge of image recognition methodology broadened from theory to real time application. I improved my report writing skills to being more clear and concise in putting across my thoughts.

Conclusions

The algorithm is able to successfully recognise the images similar to the prototype. This algorithm uses the well known Gabor filters for detecting the edges and the lines in the image. The Gabor filter parameters when tuned wisely result in input values to the COSFIRE that would result in better detections. The COSFIRE filters are useful because of their selective and versatile nature. With approriate filter configuration and good quality test images, a high precision and recall rate can be achieved. In our case, a precision of 90.5% and a recall rate of 86.6% was achieved in one case while in the other case it was 99.3% and 50.4% respectively.

4.4 Future Works

As mentioned in the Section 4.3, the test data size was modest, so in future we would like to test this algorithm on a larger dataset. This work can also be extended in detection of scenes where there could be numerous different filters configured to recognise different parts of the scene in an environment.

Bibliography

[1] Til Aach, André Kaup, and Rudolf Mester. On texture analysis: Local energy transforms versus quadrature filters. *Signal processing*, 45(2):173–181, 1995.

[2] Conrado Avendano, Ariela Mata, César A Sanchez Sarmiento, and Gustavo F Doncel. Use of laptop computers connected to internet through wi-fi decreases human sperm motility and increases sperm dna fragmentation. *Fertility and sterility*, 97(1):39–45, 2012.

[3] George Azzopardi and Nicolai Petkov. Trainable cosfire filters for keypoint detection and pattern recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(2):490–503, 2013.

[4] L Columbus. Idc: 87% of connected devices sales by 2017 will be tablets and smartphones. forbes, 2013.

[5] Aiden R Doherty, Steve E Hodges, Abby C King, Alan F Smeaton, Emma Berry, Chris JA Moulin, Siân Lindley, Paul Kelly, and Charlie Foster. Wearable cameras in health: the state of the art and future possibilities. *American journal of preventive medicine*, 44(3):320–323, 2013.

[6] Elena Gheorghiu et al. Multiplication in curvature processing. *Journal of Vision*, 9(2):23, 2009.

[7] Steve Hodges, Lyndsay Williams, Emma Berry, Shahram Izadi, James Srinivasan, Alex Butler, Gavin Smyth, Narinder Kapur, and Ken Wood. Sensecam: A retrospective memory aid. In *UbiComp 2006: Ubiquitous Computing*, pages 177–193. Springer, 2006.

[8] Roya Kelishadi, Siamak Alikhani, Alireza Delavari, Farshid Alaedini, Afshin Safaie, and Eliyeh Hojatzadeh. Obesity and associated lifestyle behaviours in iran: findings from the first national non-communicable disease risk factor surveillance survey. *Public health nutrition*, 11(03):246–251, 2008.

[9] Paul Kelly, Aiden Doherty, Emma Berry, Steve Hodges, Alan M Batterham, and Charlie Foster. Can we use digital life-log images to investigate active and sedentary travel behaviour? results from a pilot study. *Int J Behav Nutr Phys Act*, 8(44):44, 2011.

[10] Xi Lu, Junko Watanabe, Qingbo Liu, Masayo Uji, Masahiro Shono, and Toshinori Kitamura. Internet and mobile phone text-messaging dependency: Factor structure and correlation with dysphoric mood among japanese adults. *Computers in Human Behavior*, 27(5):1702–1709, 2011.

[11] T Manglik, L Axel, W Pai, D Kim, P Dugal, A Montillo, and Z Qian. Use of bandpass gabor filters for enhancing blood-myocardium contrast and filling-in tags in tagged mr images. *Proc of Int'l Society for Mag. Res. In Med*, page 1793, 2004.

[12] Steve Mann. 'wearcam'(the wearable camera): personal imaging systems for long-term use in wearable tetherless computer-mediated reality and personal photo/videographic memory prosthesis. In *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pages 124–131. IEEE, 1998.

[13] Andrea Ozias. Telework 2011: A worldatwork special report, 2011.

[14] Katrin Wolf, Albrecht Schmidt, Agon Bexheti, and Marc Langheinrich. Lifelogging: You're wearing a camera? *IEEE Pervasive Computing*, 13(3):8–12, 2014.

Appendices

.1 Algorithm

.2 List of Code Names

- MainConfigure.m : configures the filters
- apply_cosfire_28june2016_invis_markselective.m applies the filter on other images.
- cosfire_testing_pll_03july2016_invis_steps_markselective.m : is used to validate and test the images on the configured filters.
- Plot_training.m : plots the response of images as compared against the configured filter responses.
- Image_coordinates.m : plots the coordinates of keypoints on the configuration images.
- View_new_config_images.m : is used to generate the coordinates of the keypoints on the images when clicked on a specific location.

.3 Code

‘

```
clear all;
close all;
disp('########␣START␣of␣the␣Program␣########')
%set(0,'DefaultFigureVisible','off')

path('COSFIRE/',path);
a = dir('config_8july/*jpg');
path('config_8july/',path);

path('./Gabor',path);
%for i = 3:length(a);
NCOSFIRES=3;
%config_max_response = zeros(10,3);
config_max_response = zeros(10,2);
operator_num = 0;
load x_y10july.mat;
%
output_dir='outputfilters_11july';
progress_file='output_Main_ConfigureCosfire_11july.txt';
system(['rm␣-f␣' progress_file])
system(['mkdir␣-p␣' output_dir])

% assign values to params
params = Parameters_Sw;% assigning parameter values
fac_resize=0.3; orig_sel_x=sel_x;
orig_sel_y=sel_y;

sel_x=sel_x*fac_resize;
sel_y=sel_y*fac_resize;

n_files= length(sel_x)

%%%%%%%%%%%%%%%%config below%%%%%%%%
for  i=1:length(sel_x)
    tic;
    a(i).name
    prototype = imread(a(i).name); % prototype is the image in this
        case example lap1.jpg
    prototype = imresize(prototype,fac_resize);
    prototype = preprocessImage(prototype); % If image is coloured then
         convert it to grayscale
    toc;
    num=0

    for j = 1:2

        figure(1);
        % Configure a COSFIRE operator
```

```
        tic
        operator = configureCOSFIRE(prototype, round([sel_y(i,j),sel_x(
            i,j)]),params);
%       hold on;
%     %pause;
%         plot(sel_x(i,j),sel_y(i,j),'r.','MarkerSize',50);
%         img_config=[output_dir '/' a(i).name '_cosfire_config_',
    sprintf('%02d',i),'_',sprintf('%03d',j),'.jpg']
%         name_tit=[output_dir '/' a(i).name '\_cosfire\_config\_',
    sprintf('%02d',i),'\_',sprintf('%03d',j),'.jpg']
%       title(name_tit)
%         saveas(gcf,img_config,'jpg') ;
%       hold off

        toc;
        pause
        size(operator.tuples,2)% generates edges of the image

        out_str=sprintf('Image␣%s␣filter_no=%02d␣fac_resize=%5.3f␣
            original␣x_coord=%06d␣y_coord=%06d␣␣No.␣of␣tuples␣size(
            operator.tuples,2)=%04d',a(i).name, j, fac_resize,
            orig_sel_x(i,j), orig_sel_y(i,j), max(size(operator.tuples
            ,2)))

system(['echo␣␣"' out_str '"␣>>␣' progress_file])


        if size(operator.tuples,2)>4 %%% We need to check this
            % if size(operator.tuples,2)<20
            num = num+1
            %             close all;
            % Show the structure of the COSFIRE operator
            viewCOSFIREstructure(operator);% shows the filters round
                structures
            %pause;
            hold on;
           % plot(sel_x(i,j),sel_y(i,j),'r.','MarkerSize',30); T
           %% The marker stuff does not work..

            operator_num = operator_num +1;
            img_cosfire=[output_dir '/' a(i).name '_cosfire_filter_',
                sprintf('%02d',i),'_',sprintf('%03d',j),'.jpg']
            name_tit=[a(i).name '\_cosfire\_filter\_',sprintf('%02d',i)
                ,'\_',sprintf('%03d',j),'.jpg']
            title(name_tit)
            saveas(gcf,img_cosfire,'jpg') ;
            close(gcf)
                        % %
            % %
            figure(3);

            imshow(prototype); hold on;
            %             imwrite(plot(x(j),y(j),'r.','MarkerSize',20),'
                prototype2.jpg');
```

```
            plot(sel_x(i,j),sel_y(i,j),'r.','MarkerSize',30);
            %disp('Pausing');pause
            hold off;
            %close(h);
            prototype2_img = [output_dir '/' a(i).name '
                _prototype_marked_',sprintf('%02d',i),'_', sprintf('%03
                d',j),'.jpg']
            name_tit=[output_dir '/' a(i).name '\_prototype\_marked\_',
                sprintf('%02d',i),'\_', sprintf('%03d',j),'.jpg']
            title(name_tit);

            saveas(gcf,prototype2_img,'jpg') % marker image
            %              % save operator
            save([output_dir '/operator_',sprintf('%02d',i),'_',
                sprintf('%03d',j), '.mat'], 'operator') ;
            config_max_response(i,j) = max(max(applyCOSFIRE(prototype,
                operator) ));
        else
            disp('The operator does not have enough elements for the
                cosfire filter')
            a(i).name
            size(operator.tuples)
            disp('Pauseing')


        end

% disp('Waiting'); pause

        if num >= NCOSFIRES
            break
        end
    end


%disp('Pause I am waiting ')
%pause


end

save('config_max_response_11july.mat','config_max_response');

‘
```

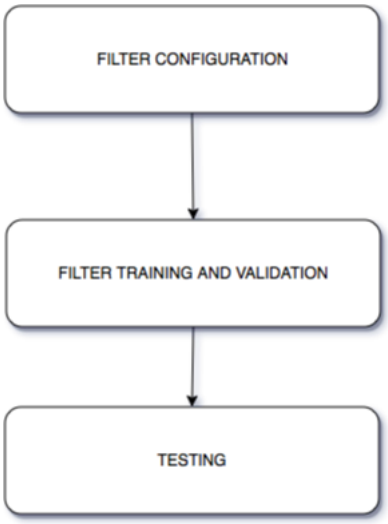


Figure 1: Flow chart representing the most important steps in the Experiment

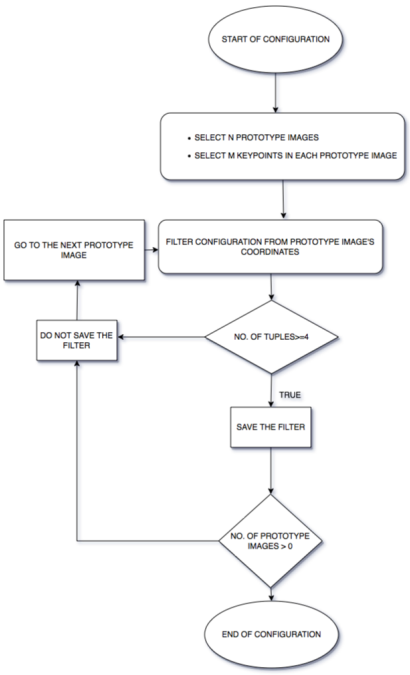


Figure 2: Steps in the Algorithm of Trainable COSFIRE Filters