KAUNAS UNIVERSITY OF TECHNOLOGY
INFORMATICS FACULTY

SHUBHAM JUNEJA

Investigation on Moving Object Tracking and Recognition Problems

Final project for Master degree

Supervisor
Assoc. Prof. Dr. Armantas Ostreika

KAUNAS, 2016

KAUNAS UNIVERSITY OF TECHNOLOGY
INFORMATICS FACULTY

Investigation of Moving Object Tracking and Recognition Problems

Final project for Master degree
Informatics (M4016M21)

Supervisor
Assoc. Prof. Dr. Armantas Ostreika

Reviewer
Assoc. Prof. Dr. Sigitas Drąsutis

Project made by
Shubham Juneja

KAUNAS, 2016

KAUNAS UNIVERSITY OF TECHNOLOGY

Informatics

(Faculty)

Shubham Juneja

(Student's name, surname)

Informatics M4016M21

(Title and code of study programme)

Investigation on Moving Object Tracking and Recognition Problems

DECLARATION OF ACADEMIC HONESTY

____ ________________ 2016

Kaunas

I confirm that a final project by me, **Shubham Juneja**, on the subject " Investigation on Moving Objects and Recognition Problems" is written completely by myself; all provided data and research results are correct and obtained honestly. None of the parts of this thesis have been plagiarized from any printed or Internet sources, all direct and indirect quotations from other resources are indicated in literature references. No monetary amounts not provided for by law have been paid to anyone for this thesis.

I understand that in case of a resurfaced fact of dishonesty penalties will be applied to me according to the procedure effective at Kaunas University of Technology.

(name and surname filled in by hand) *(signature)*

Shubham Juneja, Investigation on Moving Objects and Recognition Problems. *Masters in Informatics* final project / supervisor Assoc. Prof. Dr. Armantas Ostreika; Kaunas University of Technology, Informatics faculty.

Kaunas, 2016. XX p.

SUMMARY

Object tracking and recognition is a major part of computer vision and artificial intelligence fields. With decades of research time put into it, there have been a wide variety of algorithms and methods proposals so far. With high variance in the approaches to track and recognize objects in videos, there are several applications to each type of approach for different kinds of objects. This master's thesis analyses selected state-of-the-art methods which are used for detecting motion in videos for tracking objects & keypoints based feature extraction methods, proposes a method for a specific application domain, implements it to perform experiments. Finally, evaluates its performance to demonstrate the challenges in object tracking and recognition for the chosen application domain.

Keywords— ***Object tracking, recognition, computer vision, artificial interlligence, keypoints, feature extraction***

Shubham Juneja, Investigation on Moving Objects and Recognition Problems. *Masters in Informatics* final project / supervisor Assoc. Prof. Dr. Armantas Ostreika; Kaunas University of Technology, Informatics faculty.

Kaunas, 2016. XX p.

SUMMARY

Objektų atpažinimas ir jų sekimas sudaro svarbia dalį kompiuterijos ir dirbtinio intelekto srityse. Su dešimtmečiais tyrimų metų įdėtų į juos, iki šiol yra pasiūlyta daugybe algoritmų ir metodų šioje srityje. Su aukštos dispersijos atitinkamais metodais, siekiant stebėti ir atpažinti objektus filmuotoje medžiagoje, yra kelios programos kiekvienam iš požiūrio į skirtingus objektų tipus. Šiuo magistriniu darbu nagrinėjama pasrinktus state-of-the-art metodus kurie yra naudojami filmuotoje medžiagoje atpažinti judančius objektus ir atskirti keypoints pagrįstus atvaizdo požymių išskyrimo metodai, siūlo būdus konkrečioje taikymo srityse, juos įgyvendina atliktant eksperimentus.

Galiausiai, įvertina savo veiklos rezultatus parodant objektų stebėjimo ir atpažinimo iškylančius iššūkius pasirinktoje taikymo srityje.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Object tracking and recognition are important parts of computer vision. With the increasing need of automated video analysis, with availability of high powered computer and inexpensive high quality video cameras, object tacking algorithms and recognition algorithms are being looked into with great interest. In order to track a specific object, the object needs to be recognized to make sure if tracking is required for it. The two tasks are may not be similar but are quite related. With well defined algorithms and methods, tracking and recognition can be performed in order to automatically extract information out of pre-recorded videos or real time feeds of video cameras.

With decades of research carried out in the field of computer vision, various kinds of methods have been proposed to solve different kinds of problems. Some methods work for specific applications, depending on the way the methods extract information out of the video. Information such as motion occurring in a video, can be extracted based on the changes in pixel values. Also, the parts of the video where the motion occurs can further be investigated with understand what is present at those parts. To understand what lies in that part of the video, feature extraction can be used.

Feature extraction is a highly vital part when it comes to tracking and recognition. Objects have features, some may be common between certain objects, but the most distinguishable features of an object are extracted. The features can be shape, colour, size, and so on. In previous years, researches performed [2] [3] attempt to recognize clothing in images, by extracting features like the articulated pose of a human in an image and by segmenting the body parts based on features extracted furthermore. Another way of extracting features can be finding keypoints in the region of interest, which are considered as highly distinguishable.

This mater's thesis investigates state-of-the-art methods used to track and recognize objects. A method is then proposed to robustly track and recognize moving objects with the use of adaptive Gaussian mixture models of detecting motion and scale invariant feature transform (SIFT) method for recognition, and attempts to recognize patterns embedded on clothes and track them as they move in a video. The method is then implemented and experiments are performed to analyse and evaluate the proposed method. And finally conclusions are drawn for to give an overview of the results of the thesis.

1.1 Aim of the thesis

This thesis aims to study and analyse existing state-to-the-art methods for tracking and recognizing moving objects, then proposes a method for the same, further demonstrates the proposed method with experimentation and result evaluation.

1.2 Structure

The thesis starts off with introducing what is to be carried out within this study, in Chapter 1. Chapter 2 analysis the existing methods used to perform object tracking and recognition. Chapter 3 proposes a method to track and recognize patterns in clothes. Chapter 4 performs experiments on the proposed method and an existing method, and collects results. Chapter 5 evaluates the results based on performance and comparisons. Finally, Chapter 6 concludes on the findings of the thesis.

2. ANALYSIS

2.1 Adaptive background mixture models

The adaptive background mixture models method [4] defines a model called the background model, which represents the pixels residing behind an object or expected to remain static. This method models each pixel as a mixture of Gaussians and uses an online approximation so as to update the model. The Gaussian distributions are evaluated to determine which ones are most likely to be a part of the background process. Then every pixel is classified based on the Gaussian distributions which may or may not represent the background process.

Instead of modelling values of all pixels with one Gaussian distribution, this makes use of a mixture of Gaussians. The Gaussians in the mixture differ in terms of persistence and variance, which also determines if a Gaussian distribution represents the background or not. The pixel values which do not get classified into the background distributions are then considered as the foreground until they get classified by one of the Gaussian distributions with sufficient and consistent evidence to support it.

This approach adapts robustly with lighting changes, repetitive motions of scene elements, tracking through cluttered regions, slow-moving objects, and introduction or removal of objects from the scene.

The method for realizing the background contains two significant parameters, which are α and T. α represents the learning constant and T represents the proportion of the data which is to be accounted for background.

Method

This method takes into account the practical possibility of multiple surfaces being affected by unpredictable changes in the lighting conditions. In this scenario a single adaptive Gaussian may not be enough. To overcome this problem, this method uses a mixture of adaptive Gaussians. To make the Gaussians adaptive, the parameters of the Gaussians are updated regularly, also the Gaussians are evaluated using a simple heuristic to hypothesize which are a part of the background process. The pixels which are not classified under the background pixels are then grouped into clusters called connected components. These components are then tracked from frame to frame using a multiple hypothesis tracker.

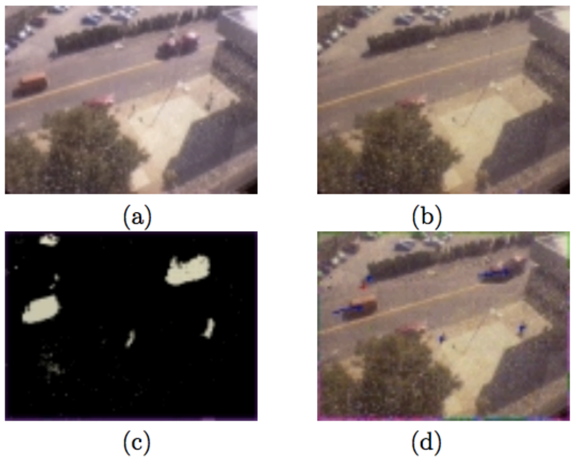


Figure 1: (a) An image from a video, (b) Image consists of the means of the most probable Gaussian means of background model, (c) Foreground pixels (d) Image from video with tracking information. [4]

Online mixture model

The mixture model considers values of pixels over time, which is referred to as a "pixel process". It is a time series of pixel values. The history of pixel values is tracked with respect to time, that is

$$\{X_1, ..., X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$$

Eq. 1

where (x_0, y_0) is a pixel in image *I*, and *i* represents a particular time *t*. The value of each pixel here represents a measurement of radiance in the direction of the sensor. This value is affected by the the lighting changes occurring in the scene, which requires to be tracked by the Gaussians. If there is a recent observation, it is considered more important in determining the parameter estimates.

The model and update procedures require some guiding factors for which, the recent history of every pixel, { X_1, ..., X_t }, is modelled by a mixture of *K* Gaussian distributions. Therefore, the probability of observing each pixel is calculated by

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

Eq. 2

where *K* is the number of Gaussian distributions, $\omega_{i,t}$ is an estimate of the weight of the *i*th Gaussian distribution at the time *t*, while $\mu_{i,t}$ being the mean of the *i*th Gaussian at time *t*, $\Sigma_{i,t}$ being the covariance matrix of the *i*th Gaussian at time *t*, where η is the Gaussian probability density function. The probability density function is given by

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T \Sigma^{-1} (X_t-\mu_t)}$$

Eq. 3

K is determined with respect to memory and computation power available. Also for computation reasons covariance matrix can be taken of the form

$$\Sigma_{k,t} = \sigma_k^2 \mathbf{I}$$

Eq. 4

which assumes that the pixel values for red, green and blue are independent and have same variances.

Mixture of Gaussians is responsible for characterization of distribution of recently observed values. When a new pixel value is observed by the model, it is represented by one of the major components and used to further update the model. Since, each pixel process varies over time, an approximate method is used which treats every new observation as a sample set of size 1 and uses standard learning rules for integration with new data.

Due to the presence of multiple mixture models in the system, to classify the pixel values K-means approximation is used. Every pixel value, X_t, is checked for a match with existing *K* Gaussian distributions until a match is found. A match is defined within 2.5 standard deviation of a distribution, that is a threshold which can be perturbed. This plays an important role since different regions may have different lightings.

If K-means approximation does not match with any of the *K* Gaussian distributions, the least probable distribution is replaced with a distribution as its mean value, with initially high variance and a low prior weight.

The prior weights of the *K* Gaussians at a specific time *t*, are adjusted as below

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t})$$

Eq. 5

where α is the learning rate, and $M_{k,t}$ is set to 1 for a model where there is a match and set to 0 for the models there is no match. After the approximation, the weights are normalized. The speed at which distribution parameters change is determined by $1/\ \alpha$ resultant. $\omega_{k,t}$ is a low pass filtered average of the posterior probability with a condition that states, the pixel values have matched with model *k,* considering given observations made from time 1 to *t*.

The parameters μ and σ for unmatched Gaussian distributions remain same but in case of the matched Gaussian distributions with new observations, a process to update is followed. The updating is done based on the following equations:

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t$$

Eq. 6

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t)$$

Eq. 7

where

$$\rho = \alpha\eta(X_t|\mu_k, \sigma_k)$$

Eq. 8

which is the same type of low pass filter as in the previous case, but with inclusion of the data which is matched with the model, into the estimation.

The strength of this method is that if something is allowed to become a part of the background, it does not destroy the existing model, instead it will be moved near the K^{th} most probable Gaussian distributions and, if it holds the last position and a new value is observed then it may be eliminated. Hence, if an object is stationary long enough to become a part of the background and then it moves, the distribution describing the previous backgrounds still exists with the same values of μ and σ^2 , but with a lower ω.

Background model estimation

The background model estimation aims to determine which of the Gaussian distributions are most likely to produce background processes. This relies on the Gaussian distributions which have the most supporting evidence and the least variance. Hence, by the end of this process it concludes which portion of the mixture models represent the background process.

To start off, the Gaussians are ordered by the value of ω/σ. This value increases with higher gains and decreases with higher variance. After re-estimating the parameters, they are sorted starting from the recent matches to the most probable distributions. Hence most probable background distributions remain on top and less probable transient distributions move towards the bottom and may get replaced by new distributions.

The first *B* distributions are selected as the background model, where

$$B = argmin_b \left(\sum_{k=1}^{b} \omega_k > T \right) .$$

Eq. 9

Here, *T* is a measure of minimum portion of the data that should be accounted for the background.

Connected components

The above method identifies the pixels if they are a part of the background or the foreground. This is done by running through the pixel processes. The labelled foreground pixels are then accumulated together as connected components. This procedure is effecting in determining the entire moving object. The moving objects can be determined by their positions, directions, size and other characteristics, which can be useful for later processing and classification.

Multiple hypothesis tracking

Calculating the correspondence of connected components between frames is done using linear predictive multiple hypotheses tracking algorithm which takes into account the position and the size. This algorithm maintains creates and maintains sets of Kalman filters.

At every frame, there is an available pool of Kalman models and a new pool of connected components. Available models are matched with connected components, and if the connected components are not matched at first, then new Kalman models are found and matched. The modes which fall outside the threshold are removed.

2.2 Optical Flow Calculation Algorithm

Optical flow resembles the translation of each pixel in a dense field of displacement vectors. To compute the translations, the brightness constancy of corresponding pixels is observed and calculated with respect to consecutive frames. [1] mentions this concept in a survey on Object tracking methods.

Optical flow is referred to as one of the state-of-the-art concepts in computer vision and has been proposed several times with different techniques over multiple decades. The most famous methods based on optical flow have been proposed by Horn and Schunck [5], Lucas and Kanade [6], Black and Anandan [7], and Szeliski and Couglan [8], as mention by [1]. Out of these, Lucas and Kanade's proposed method stands out with its multiple revisions and other methods citing it to build upon. One of the very famous revision of this methods is the pyramidal implementation of Lukas and Kanade feature tracker.

Pyramids are multi-scale representation of signals. This signal is an image in the field of image processing and computer vision. A pyramidal representation defines multiple levels of a single image. At each level the resolution is halved with the help of an appropriate filter, which is sometimes referred to as image smoothing filter. This technique helps in compression of an image and also in image analysis [9].

Concept of the method

The method is based on finding out differences between images, i.e. considering *I* and *J* to be two 2D grayscale images. $I(x) = I(x,y)$ and $J(x) = \mathrm{J}(x,y)$ being the grayscale values of two images at a location $\mathrm{x} = [x\ y]^T$, where *x* and *y* are two pixel coordinates at any image point x. The first image is considered as *I* while the second as *J*, where n_x and n_y are the width and height of the images.

The method considers a point $\mathrm{u} = [u_x\ u_y]^T$ on the first image *I*, and then aims to find $\mathrm{v} = \mathrm{u} + \mathrm{d} = [u_x + d_x\ u_y + d_y]^T$ on the second image *J*, with the goal of $I(\mathrm{u})$ and $J(\mathrm{v})$ being similar. The vector $\mathrm{d} = [d_x\ d_y]^T$, being the vector velocity at point x, is basically the optical flow at point x. Two integers ω_x and ω_y , are initialized which resemble the window size, in the method. Within the window, image velocity d is defined, as the vector which minimises the residual function:

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \left(I(x,y) - J(x+d_x, y+d_y)\right)^2 .$$

Eq. 10

Tracking algorithm

The algorithm aims to find a balance between two curial components, accuracy and robustness. The accuracy relates not to clear out crucial parts of the image so details are not lost, which requires a small window size. Whereas, the robustness component is related to sensitivity of the tracking outcomes. To track and handle large motions, the window size should be big. Hence, to overcome this trade off, pyramids are used.

Image pyramid representation

For the pyramid representation of an image *I*, would be defined with multiple levels as an image pyramid contains. An image at level 0 would be denoted as $I^0 = I$, since the level zero represents the actual image (raw image). Every level *L*, has its width and height, therefore at level zero the width and height would be denoted by $n^0_x = n_x$ and $n^0_y = n_y$. Every level of the pyramid is derived from its lower level, that is, I^1 from I^0, I^2 from I^1, and so on. Hence, the image I^{L-1} is defined as :

$$\begin{aligned} I^L(x,y) \quad = \quad & \frac{1}{4} I^{L-1}(2x,2y) + \\ & \frac{1}{8}\left(I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)\right) + \\ & \frac{1}{16}\left(I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y+1)\right). \end{aligned}$$

Eq. 11

The dimensions of the next level of the same image depends on the dimensions of the previous level. Therefor there is a need to satisfy two equations for values *x* and *y*, such that $0 \leq 2x \leq n^{L-1}{}_{x-1}$ and $0 \leq 2y \leq n^{L-1}{}_{y-1}$. The equations are as follows:

$$\begin{aligned} n^L_x \quad & \leq \quad \frac{n^{L-1}_x + 1}{2}, \\ n^L_y \quad & \leq \quad \frac{n^{L-1}_y + 1}{2}. \end{aligned}$$

Eq. 12

By following the above three equations, , image pyramids are constructed. The levels of the pyramids are usually constructed up to levels 2,3 or 4. Constructing beyond level 4 the would not make sense as suggested by [6]. For example an image of size 640 x 480, the levels I^1, I^2, I^3 and I^4 would scale down to 320 × 240, 160 × 120, 80 × 60 and 40 × 30, respectively.

Pyramid feature tracking

For all the levels of a pyramid, u^L is defined corresponding to the point *u* on the pyramidal images. The vectors u^L are calculated as:

$$\mathbf{u}^L = \frac{\mathbf{u}}{2^L}.$$

Eq. 13

The vector calculation for pyramidal tracking starts from the deepest pyramid level. Once the pixel displacement is calculated for pyramid level *L*-2 or the deepest pyramid, the value is propagated to calculate pixel displacement for pyramid level *L*-1. This process is followed until the 0th level is reached, that is, the original image.

There exists a recursive operation between levels of pyramid, which is responsible to calculate the pixel displacements. Given the initial guess optical flow from particular levels *L* and *L*+1, g^{L}, to calculate the optical flow at level *L*, the residual pixel displacement vector d^L needs to be calculated to minimise the image matching error function:

$$\epsilon^L(\mathbf{d}^L) = \epsilon^L(d_x^L, d_y^L) = \sum_{x=u_x^L-\omega_x}^{u_x^L+\omega_x} \sum_{y=u_y^L-\omega_y}^{u_y^L+\omega_y} \left(I^L(x,y) - J^L(x+g_x^L+d_x^L, y+g_y^L+d_y^L)\right)^2.$$

Eq. 14

The guess flow vector g^L is used to pre-translate patches in the second image *J*, hence the residual flow vector d^L is small and easy to calculate through a Lucas and Kanade step.
Assuming that the vector d^L has been calculated, then the result of the this calculation is propagated to the next level by passing the new initial guess.

$$\mathbf{g}^{L-1} = 2\,(\mathbf{g}^{\mathbf{L}} + \mathbf{d}^L).$$

Eq. 15

Once the lowest level of the pyramid has been reached, the algorithm sets the initial guess for the that level to zero:

$$\mathbf{g}^{L_m} = [0 \;\; 0]^T.$$

Eq. 16

The final solution to the lowest level is written as:

$$\mathbf{d} = \mathbf{g}^{\mathbf{0}} + \mathbf{d}^0.$$

Eq. 17

Therefore, the final solution for the entire pyramid can be given by the following:

$$\mathbf{d} = \sum_{L=0}^{L_m} 2^L \, \mathbf{d}^L .$$

Eq. 18

Iterative Lucas Kanade

This computation is referred to as the core optical flow computation in this algorithm. It is aimed to find the vector d^L that minimizes the function ε^L, on the entry level L. This function generates new images A and B, and is defined as:

$$\forall (x,y) \in [p_x - \omega_x - 1, p_x + \omega_x + 1] \times [p_y - \omega_y - 1, p_y + \omega_y + 1],$$

$$A(x,y) \doteq I^L(x,y),$$

Eq. 19

$$\forall (x,y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y],$$

$$B(x,y) \doteq J^L(x + g_x^L, y + g_y^L).$$

Eq. 20

Based on the previous equations, the goal is to find the new displacement vectors which minimises the matching function:

$$\varepsilon(\overline{\nu}) = \varepsilon(\nu_x, \nu_y) = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(A(x,y) - B(x+\nu_x, y+\nu_y)\right)^2 .$$

Eq. 21

The first derivative of ε gives zero, where the expansion of the derivative gives:

$$\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} = -2 \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(A(x,y) - B(x+\nu_x, y+\nu_y)\right) . \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} .$$

Eq. 22

Substituting B(x + v_x, y + v_y) by it's first order Taylor expansion about the point v = [0 0]^T, for the purpose of approximation, gives:

$$\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} \approx -2 \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(A(x,y) - B(x,y) - \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \overline{\nu}\right) . \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} .$$

Eq. 23

Where A(*x, y*) − B(*x, y*) can be interpreted by temporal image derivative at point [*x y*]^T:

$$\forall (x,y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y],$$

$$\delta I(x,y) \doteq A(x,y) - B(x,y).$$

Eq. 24

The matrix [∂B/∂x ∂B/∂y] is the image gradient vector:

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \doteq \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}^T.$$

Eq. 25

I$_x$ and I$_y$ are calculated with the use of central difference operators:

$$\forall (x,y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y],$$

$$\begin{aligned} I_x(x,y) &= \frac{\partial A(x,y)}{\partial x} = \frac{A(x+1,y) - A(x-1,y)}{2}, \\ I_y(x,y) &= \frac{\partial A(x,y)}{\partial y} = \frac{A(x,y+1) - A(x,y-1)}{2}. \end{aligned}$$

Eq. 26

And hence, the optical flow vector is calculated with:

$$\overline{\nu}_{\text{opt}} = G^{-1}\,\overline{b}.$$

Eq. 27

The expression is only valid if matrix G is invertible.

2.3 Scale Invariant Feature Transform

The Scale Invariant Feature Transform (SIFT) algorithm is designed to be invariant to:

- Scale,
- Rotation,
- Illumination,
- Viewpoint.

The algorithm relies on 4 major stages, which are as follows:

- **Scale-space extrema detection:**

This is the first stage where search is performed in all in all scales and locations of the image. It is carried out by calculating the difference of Gaussians to find interest points which are invariant to scale and orientation.

- **Keypoint localization:**

At every location a detailed model is fit to determine the the scales of the point. The keypoints are selected based on their stability.

- **Orientation assignment:**

Based on local image gradient directions, one of more orientations are assigned to each keypoint locations.

- **Keypoint descriptor:**

Local gradients are measured at a selected scale, around the selected keypoints. These are then transformed into a representation which allows a fixed amount of changes in the local shape distortion and illumination.

The algorithm has been named Scale Invariant Feature Transform because it transforms image data into scale-invariant coordinates relative to local features.

Scale space extrema detection

This stage involves finding locations and scales that can be assigned repeatedly under varying views of the same object, which is to be recognized. To identify the features which are invariant to highest number of scales, the search is performed under all possible scales of the image. The set of all possible scales is known as the scale space.

The scale space of an image is defined as a function:

$$L(x, y, \sigma)$$

Eq. 28

Which is obtained by performing convolution of a variable scale Gaussian $G(x, y, \sigma)$, with an input image $\mathrm{I}(x, y)$:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y),$$

Eq. 29

where

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}.$$

Eq. 30

To detect table keypoint locations in the scale space, scale space extrema is involved in within the difference of Gaussian function, while being convolved with the input image.

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma). \end{aligned}$$

Eq. 31

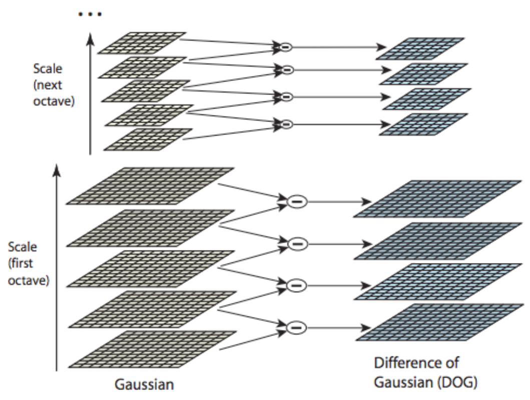


Figure 2: Calculation of difference of Gaussian [10]

Figure 2 shows an efficient way to construct D(x, y, σ) where the images are incrementally convolved with a constant factor k in a scale space. Each octave of a scale space is divided into an integer number of intervals, represented as s, so $k = 2^{1/s}$. Adjacent image scales are subtracted to produce the difference of Gaussians.

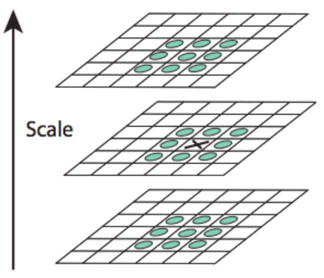


Figure 3 Shows the pixels considered for comparison [10]

To find the local maxima and minima of D(x, y, σ), each point is compared to its eight neighbours in the current image and 9 neighbours in the scale above and below. The point is selected only if the point is larger or smaller than all of the points.

Keypoint localization

A keypoint candidate once found by comparing to its neighbours, is then evaluated to find a detailed fit the nearby data for location, scale and ratio of principal curvatures. This helps in rejection of weak points, i.e. points which have low contrast or are poorly localized along the edge.

The approach used to perform this task, uses Taylor expansion of the scape space function, D(x, y, σ), shifted so that the origin is located at the sample point:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathbf{T}}\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

Eq. 32

where D and its derivatives are evaluated at a sample point and x = $(x, y, \sigma)^{\mathrm{T}}$ is the offset from this point, due to the shift. The location of the detected extremum, is determined by calculating the derivative of this function with respect to x and setting it to zero, as

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1}\frac{\partial D}{\partial \mathbf{x}}.$$

Eq. 33

If the offset is larger than 0.5 in any of the dimensions, then it would mean that the extremum lies closer to a different sample point. In this situation, the sample point is then changed and then interpolation is performed instead. The final offset is then added to the location of its sample point, to get the interpolated estimate for the required location of the extremum.

The following function is used to reject unstable extrema with low contrast, which is given as:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T\hat{\mathbf{x}}.$$

Eq. 34

Orientation assignment

With assignment of orientation to each keypoint based on it's local image properties, the keypoint descriptor can be represented relative to the assigned orientation and therefore achieve invariance to image rotation. For this, the scale of the keypoint is used to select the smooth Gaussian image, L, with the closest scale, in order to perform all the calculations in a scale invariant manner. For every image sample, $L(x,y)$, at a scale, the gradient magnitude, $m(x,y)$ and orientation, $\theta(x, y)$, is computed in advance using pixel difference:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Eq. 35

From the gradient orientations of sample points within a region around keypoints, a orientation histogram is formed. This orientation histogram has 36 bins which represent 360 degrees range of orientations. Every sample which is added to the histogram is weighted by its gradient magnitude and by a Gaussian weighted circular window with a σ that is 1.5 times the size of scale of the keypoint.

In the orientation histogram, peaks are considered as corresponding to the dominant directions of local gradients. With respect to the highest peak's size in the histogram, any other local peak being more than 80% of the highest peak is used to also create a keypoint with that orientation. Hence, locations with multiple peaks of similar magnitude will have multiple keypoints created with same scale and location but different orientations. Only 15% of the keypoints will have multiple orientations of, but these will be significant for matching.

Keypoint descriptor

Further step involves the computation of a descriptor for the local image region that is highly distinctive and also invariant to variations such as illumination and 3D viewpoint. To generate a keypoint descriptor, firstly the image gradients are sampled around the keypoint location, using the scale of the keypoint the Gaussian blur is applied to the image. To achieve orientation invariance, the coordinates of the descriptor along with the gradient orientations are rotated with respect to the keypoint orientation.

To avoid sudden changes in descriptor with small changes in the position of the window, and to give more emphasis to the center of the descriptor instead of the points far away from the center, a Gaussian weighting function is used. This function is provided with σ as a parameter, with the value equal to one half the width of the descriptor window which is used to assign a weight to the magnitude of each sample point.

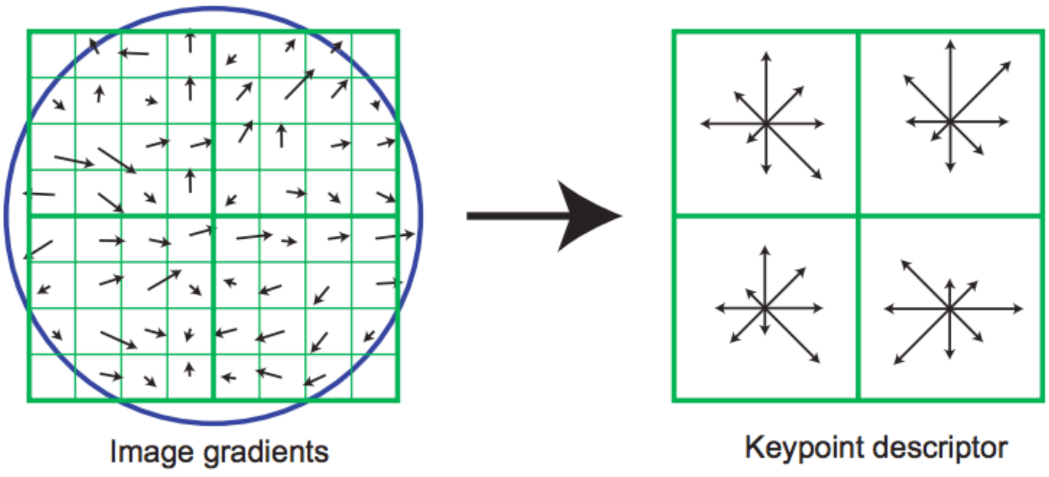


Figure 4 Image gradient and keypoint descriptor representation [10]

The keypoint descriptor allows shift in gradient positions by creating orientation histograms over 4x4 sample regions. Each orientation consists of 8 arrows in different directions for each orientation histograms, with sizes of the arrow corresponding to the magnitude of the histograms.

Every new entry in the bin is multiplied by a value of the bin as measured in units of histogram bin spacing. Finally some modifications are performed to reduce the effects of illumination changes. The feature vector is normalized to to unit length and changes in image contrast and brightness are adjusted. Therefore, the descriptor becomes invariant to affine changes in illumination.

2.4 Shi-Tomasi Corner Detector

The Shi-Tomasi detector is a corner detector which is designed to track good features, as mention in the paper "Good features to track" [11]. It is based on Harris corner detector [12] with some improvements. Majority of the concept remains the same, but Shi & Tomasi propose improvements which give better results than the Harris detector.
The algorithm aims to identify the patches where there is large variation when moved around.
To find the variations, the following equation is utilized:

$$E(u,v) = \sum_{x,y} w(x,y)\,[I(x+u,y+v) - I(x,y)]^2$$

Eq. 36

Where E is the difference between original and moved window, u and v are window's displacements in x and y directions, w(x,y) represents the window at point (x,y), I being the intensity of the image at position (x,y), I(x+u, y+v) being the intensity of the moved window and I(x,y) as the intensity of the original.

The windows proving large E are considered, so in order to find the maximum E value, the equation is maximized and expand it using Taylor's series, as follows:

$$\sum_{x,y}[I(x+u, y+v) - I(x,y)]^2$$

Eq. 37

$$E(u,v) \approx \sum_{x,y}[I(x,y) + uI_x + vI_y - I(x,y)]^2$$

Eq. 38

With application of Taylor series an approximation is achieved. Further, the terms inside the brackets are expanded due to the square.

$$E(u,v) \approx \sum_{x,y}[I(x,y) + uI_x + vI_y - I(x,y)]^2$$

Eq. 39

The same equation is re-written as

$$E(u,v) \approx [u \ \ v](\sum \begin{bmatrix} {I_x}^2 & I_xI_y \\ I_xI_y & {I_y}^2 \end{bmatrix}) \begin{bmatrix} u \\ v \end{bmatrix}$$

Eq. 40

The summed matrix is then renamed as M,

$$M = \sum w(x,y) \begin{bmatrix} {I_x}^2 & I_xI_y \\ I_xI_y & {I_y}^2 \end{bmatrix}$$

Eq. 41

The final equation with the above value becomes,

$$E(u,v) \approx [u \ \ v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

Eq. 42

To determine what interest windows to be used, Harris's [12] proposed technique calculates an R value and compares with a threshold, if calculated value meets expectation the points are considered as good tracking points.

The calculation is done as follows:

R=det M-k (trace M^2)

det M = $\boldsymbol{\lambda}_1\boldsymbol{\lambda}_2$

trace M = $\boldsymbol{\lambda}_1 + \boldsymbol{\lambda}_2$

Eq. 43

Where, λ 1 AND λ 2 are eigenvalues of the matrix M.

Shi & Tomasi demonstrate that the same can be done with the following equation:

$$R = \min(\lambda_1 \lambda_2)$$

Eq. 44

Where R has to be greater than a certain pre-defined value, then it can be marked as a corner. Therefore, the effect region can be seen in Figure 5.

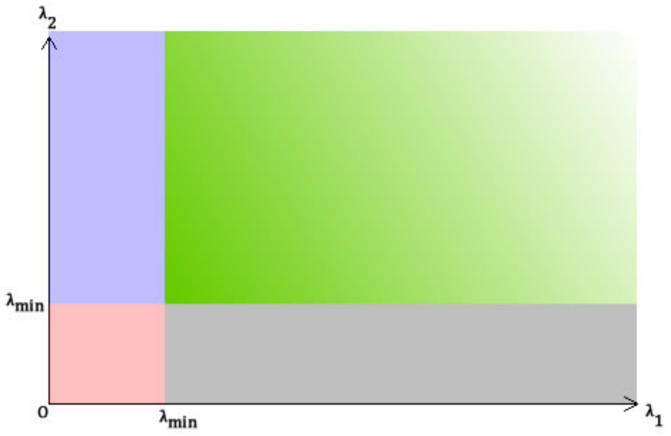


Figure 5 Acceptable Eigen values regions [13]

Where only the green region is acceptable for corners, while blue and grey region show either λ 1 or λ 2 are less than required, and red shows that both are less than required.

3. DESIGN AND IMPLEMENTATION

3.1 Aim

This study aims to design a method for object tracking and recognition. The object which has been chosen to recognize and track is a pattern that is embedded on clothes. Therefore, this method is an attempt to track humans based on their the patterns of their clothes.

3.2 Problem to solve

Lucas Kanade algorithm is based on the idea of identifying the pixels within consecutive images where a motion can be detected. Once the motion is detected the Shi-Tomasi algorithm is then responsible for identifying interest regions where there is a significant variation to be noticed. These interest regions create keypoints which can be tracked as per the detected motion.

This combination results in loss of keypoints while being tracked. When there is a motion, there may be some kind of transformation causing image regions to change shape. Shi-Tomasi tend to find the corner point in the interest region. Corner points can be tracked only when seen or with a constant shape. Due to movements such as rotation, these points may be lost or the points will never appear in order to be selected as a feature. This makes the tracker weak and unreliable for tracking patterns.

The strength of the Lukas Kanade algorithm is it's ability to track minute pixel displacements as well as major displacements. This sensitivity is achieved with the help of pyramidal implementation of the algorithm, by [6]. This strength may not be a requirement for all kinds of applications, that is, applications where the objects which are needed to be tracked consume pixels enough to be labelled as a big object.

The problem to be solved here requires detection of motions which are not as small as 10 pixels. Another requirement is to track a pattern, which should be determined if the visible pattern is the target pattern needed to be tracked or it is not. To do so, the right features should be extracted and be recognized as a they were extracted.

3.3 Approach

Adaptive background mixture models method proposes a robust way of segmenting the part of the image where there is no motion, from the part of the image where motion exists, i.e. segmenting background from the foreground as in [4]. These parts are recognized and segmented in pixel units. With the use of connected components the segmented foreground pixels can be connect and formed as a single object. Connecting components adds flexibility in selection of the size of the motion detection. As per requirement small motions can be ignored and the larger motions can be considered.

If the motion detection is limited to larger motions, then the motion being detected within another moving object which may be in another direction and smaller according to the selected threshold, it can be avoided.

Once the motion has been detected, recognition of the pattern would confirm if it should be tracked or not. In order to determine if the pattern is the one that should be tracked, a robust feature extraction algorithm will make the process reliable. The SIFT algorithm proves to be immune against rotation, translation and most importantly scale variation. The features extracted can then be compared to the required pattern, if the match is successful the matching features can then be displayed on that particular frame. This task being performed iteratively may form a tracking system with recognition.

To reduce the number of SIFT calculations, SIFT keypoints may only be calculated when there is a detection. Once the SIFT keypoints are calculated for a detection, next would be the matching process. Recognition would be confirmed with successful matches. To make SIFT more accurate, it may only find points be run on the connected components representing a detection.

3.4 Implementation

Tools

To implement the proposed method of detection, Matlab was used. Matlab provides an environment for rapid scripting to create and experiment with prototypes. Also, Matlab is loosely typed and gives simple access to mathematical functions. Along with Matlab, the Computer Vision Toolbox [14] was utilized for video related functions, and the VL Feat Library which implement's David Lowe's SIFT algorithm for feature extraction [15].

Implementation

In order train the program with the features that should be detected, we start the implementation with feature extraction of training images. SIFT feature extraction creates features and their descriptors which are stored in variables, for the sake of simplicity.

Further we initialize the Gaussian mixture models to be used and a function for detection of the foreground pixels, followed by training a blob analyser which is responsible for connecting components on the foreground. In order to remove noise we initialize a matrix which would act as a filter for noise in the foreground pixels.

These initializations are then used on every frame in a sequence. A video which is to be read, is sliced into frames. On every frame of the video the foreground is detected. This foreground gives pixels, which may contain noise, so we apply the filter for noise removal. Now that the foreground consists of more pixels representing motion and less pixels representing noise. The neighbouring components can be connected with the the initialized blob analyser. This completes the implementation regarding detection of motion.

To recognize if the motion contains the object which is to be tracked and recognized, the stored features are compared with the detected motion area's features with the VL Feat's SIFT feature matching function. But this is only done if the blob analyser returns any detected blobs (components). Every detected blob's location on the frame is extracted and a rectangular frame area (sub-image) is calculated to for extraction and comparison of SIFT features.

On successful extraction, the matched features are highlighted with red asterisks. Hence wherever there is a motion and the object to be recognized exists, the implemented program highlights it.

3.5 Algorithm for the proposed method

```
1. Initialize training SIFT keypoints
2. Initialize Gaussian mixture models
3. Initialize foreground detector
4. Initialize foreground noise filter
5. Initialize blob analyser
6. Run through all frames
7. For every frame:
        Extract foreground and remove noise
        Extract components
        If a component exists
            Calculate sub-image
            Calculate SIFT of sub-image
            Calculate matches with training keypoints
            If matches exist
                Plot matches
            End if
        End if
```

3.6 Flowchart

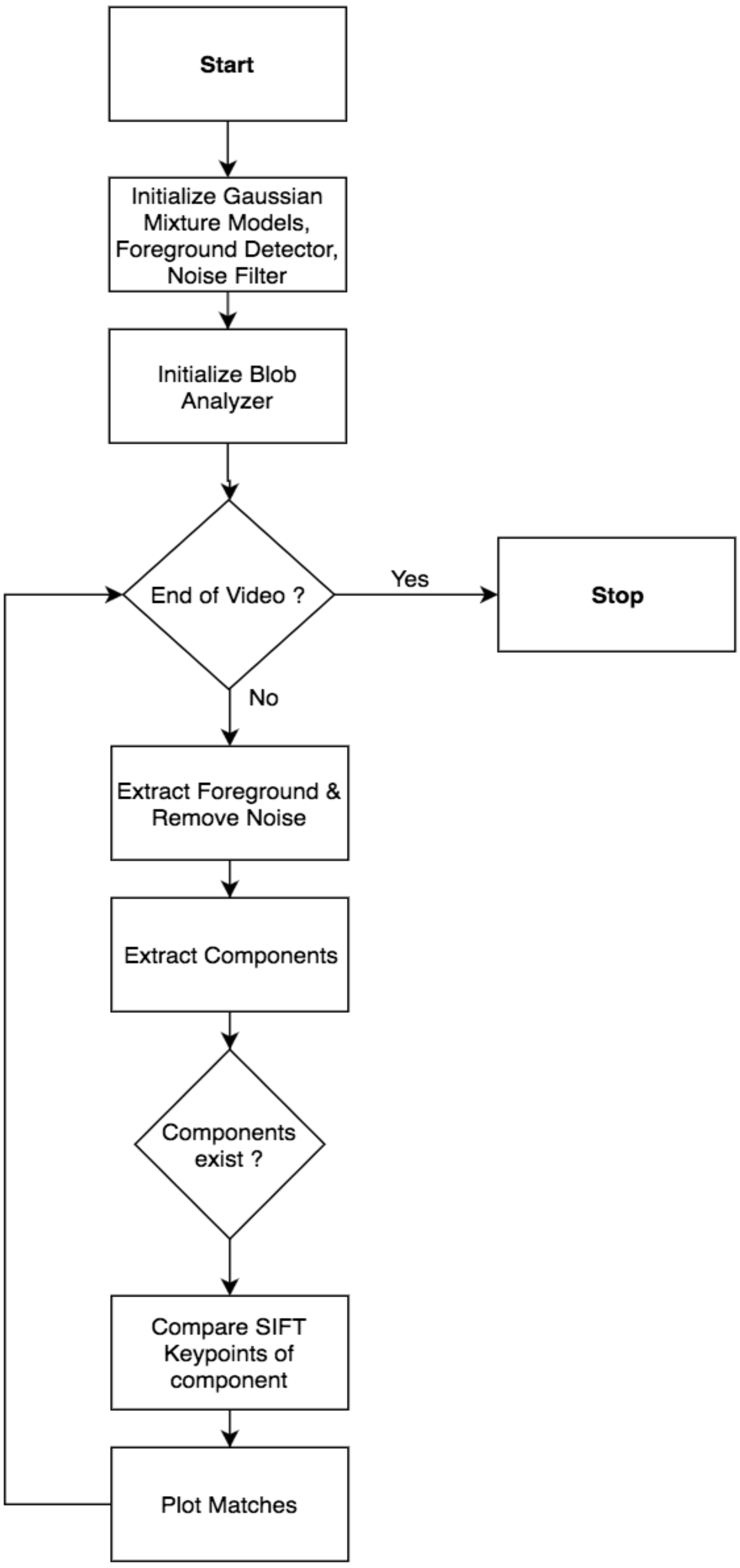

Start
Initialize Gaussian Mixture Models, Foreground Detector, Noise Filter
Initialize Blob Analyzer
End of Video ?
Yes
Stop
No
Extract Foreground & Remove Noise
Extract Components
Components exist ?
Compare SIFT Keypoints of component
Plot Matches

Figure 6: Flow chart of the proposed method.

4. EXPERIMENTATION

Experiments are performed with the proposed method and the best performing values are obtained. Further calculations are performed based on predefined metrics to determine how the method performs on its own. Lucas Kanade Pyramidal method with the use of Shi-Tomasi feature extraction, is compared to the proposed method in terms of speed and visual performances.

4.1 Parametric experimentation

We list the important parameters to be considered during for this method:

- Number of Gaussian Mixture Models,
- Adaptive Learning,
- Minimum Background Ratio,
- SIFT matching threshold.

To experiment with these parameter we choose the following pattern to be recognized with five different images of the same pattern:

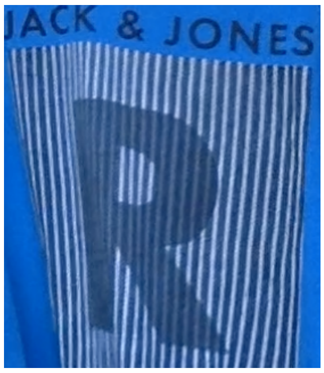


Figure 7 Candidate image for training

We choose three videos, where the first video does not have the required pattern to be recognized, while the second video contains it, and the third video also contains it but with partial visibility.

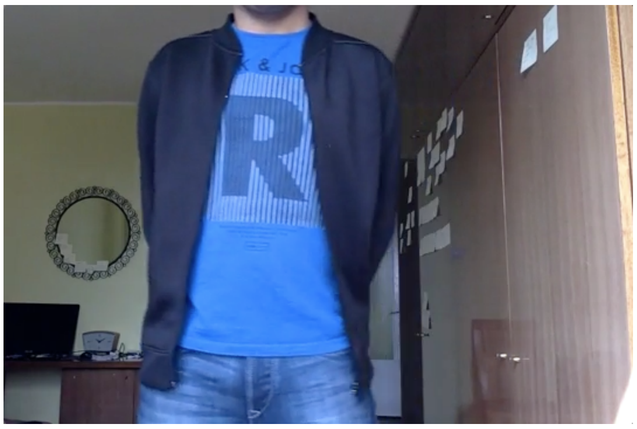


Figure 8 Screenshot from a test video, with pattern to be recognized partially occuluded.

All 3 videos are of similar length, are taken from the same view point and have a stable camera.

The method is tested with varying parameter values on each video specified above. The selection of the parameter values is done progressively, i.e., if a parameter gives good results with respect to all the test videos, it is selected to test further parameters.
The for parameters are experimented with as follows:

Number of Gaussian Mixture Models.
As the concept of adaptive mixture models provides with the flexibility of having adaptive mixture models, the count of these adaptive mixture models is also adjustable. For this experiment, we try variations in the number of mixture models with 2, 3, 4 and 5 mixture models.

The experiment is carried out with SIFT matching threshold 2.5, adaptive learning enabled and minimum background ratio 0.7.
The following results are generated with varying number of mixture models:

Table 1. Specifies the results of experiments with varying number of Gaussian Mixture Models.

Number of Gaussian Mixture Models	Pattern Missing		Pattern Present		Pattern Occluded	
	Matches	Detections	Matches	Detections	Matches	Detections
2	1796	1212	22032	1914	11240	1917
3	2272	854	5986	930	6715	1050
4	2070	708	5219	766	5289	927
5	2073	700	4032	764	5286	913

The result of two mixture models shows detections of motion in the videos, where as the other values take a major part of the motion as a static surface and hence a lot of motion stays undetected, leaving out few opportunities for recognition and tracking.

Adaptive learning

The feature of adaptive learning can be made static, where the weight adjustment would show any changes throughout the process, also the newly introduced objects into the frames will be treated the same as existing objects. We test with adaptive learning turned on and off.

We progress with previously used values of SIFT matching threshold and minimum background ratio, with using 2 Gaussian mixture models.

The following results are generated:

Table 2. Specifies the results of experiments with adaptive learning enabled and disabled.

Adaptive learning	Pattern Missing		Pattern Present		Pattern Occluded	
	Matches	Detections	Matches	Detections	Matches	Detections
Enabled	1796	1212	22032	1914	11240	1917
Disabled	8729	2553	19570	1573	7586	1524

With adaptive learning disabled the detection of motion in the video with pattern missing rises to a very high number, showing instability in understanding of motion in the video. This gives rise to performing unwanted identification and matching of SIFT descriptors

Minimum Background ratio

The minimum background ratio represents the ratio of the pixels to be considered as part of the background. The default value suggested for the used implementation is 0.7. We experiment with 0.5 and 0.7.

This experiment is carried out with previously used SIFT matching threshold, adaptive learning enabled and 2 Gaussian mixture models.

The following results are generated:

Table 3. Specifies the results of experiments with different Minimum Background Ratio.

Minimum Background Ratio	Pattern Missing		Pattern Present		Pattern Occluded	
	Matches	Detections	Matches	Detections	Matches	Detections
0.5	1874	1259	25560	1983	10397	1799
0.7	1796	1212	22032	1914	11240	1917

With 0.5 as the selected ratio, there is additional detection of unwanted motion, as the pattern in the video consumes less than 50% of the pixels.

SIFT matching threshold

This parameter is required to restrict the variation in keypoint matches. With a high value, the matches with only closer scores will be listed as matches. Therefore, the value needs to specified in order avoid misleading feature pairs for the object to be recognized with accuracy. The default value for SIFT is 1.5, we experiment with increasing values to reduce the number of matches where the object to be recognized does not exist in the test video.

We perform this experiment with adaptive learning enabled, number of Gaussian mixture models set to 2 and minimum background ratio set to 0.7.

The following results are generated:

Table 4. Specifies the results of experiments with varying SIFT matching threshold.

SIFT matching threshold	Pattern Missing	Pattern Present	Pattern Occluded
2.5	1796	22032	11240
3	705	17336	8406
3.5	337	14584	7202
4	193	12660	6532
4.5	105	11306	6069
5	53	10293	5702
5.5	31	9499	5240
6	17	8841	5204

It can be seen that with 6 as the threshold value, results closest to the ground truth are obtained.

4.2 Performance based results calculation

We determine metric based results of the proposed method. The right metric to consider for this scenario would be frame-based metrics since we are processing the video on every frame and we have information available regarding object tracking and recognition with respect to frames. The frame based metrics have been specified by Bashir et. al [16].

To calculate the based on the frame based metrics, we assume that all the matches in the video with the pattern present are correct, and all the matches in the video with the pattern absent are incorrect. Then we assume that both the videos have been concatenated together and considered as one single video.

To get the closest approximations, we use the parameters which give the best results, which are same as the parameters considered for the experiment for SIFT matching threshold and considering SIFT matching threshold itself as 6. We also consider the results generated with these parameters with the proposed method.

Based on observations, results and the input video we have,
Total Frames (TF) = 2391
Ground Truth (TG) = 1091
True Positive (TP) = 591
False Positive (FP) = 17
True Negative (TN) = TF – GT = 1300

According to [16] we have,
Specificity = TN / (FP + TN)
False Alarm Rate = FP / (TP + FP)
Tracker Detection Rate = TP / TG
Accuracy = (TP + TN) / TF
Positive Prediction = TP / (TP + FP)
False Positive Rate = FP / (FP + TN)

Hence, we obtain the values by substituting what we have.
Specificity = 0.99

False Alarm Rate = 0.03

Tracker Detection Rate = 0.54

Accuracy = 0.79

Positive Prediction = 0.97

False Positive Rate = 0.01

4.3 Cross-method experimentation

Once the right parameters have been identified, they are then initialized and the proposed method is tested again. This time the proposed method is evaluated for the time required to process the input. The same video inputs are provided to the Lucas Kanade pyramidal implementation with Shi-Tomasi, and its time required process the inputs is noted. An existing implementation has been used for Lucas Kanade with pyramids and Shi-Tomasi.

Along with time comparisons, observations over the tracking are made to compare how well do the methods perform when features are in motion. The tests are run on a Macbook Air 2015 model which runs over a 1.6 gigahertz dual core Intel i5 processor with 4 gigabytes of random access memory.

Results

The results are as follows:

Table 5. The table shows time taken by Lucas Kanade method and the proposed method, to process given number of frames.

Number of frames to process	Lucas Kanade method (s)	Proposed method (s)
1025	391	206
1366	747	141
1307	646	178

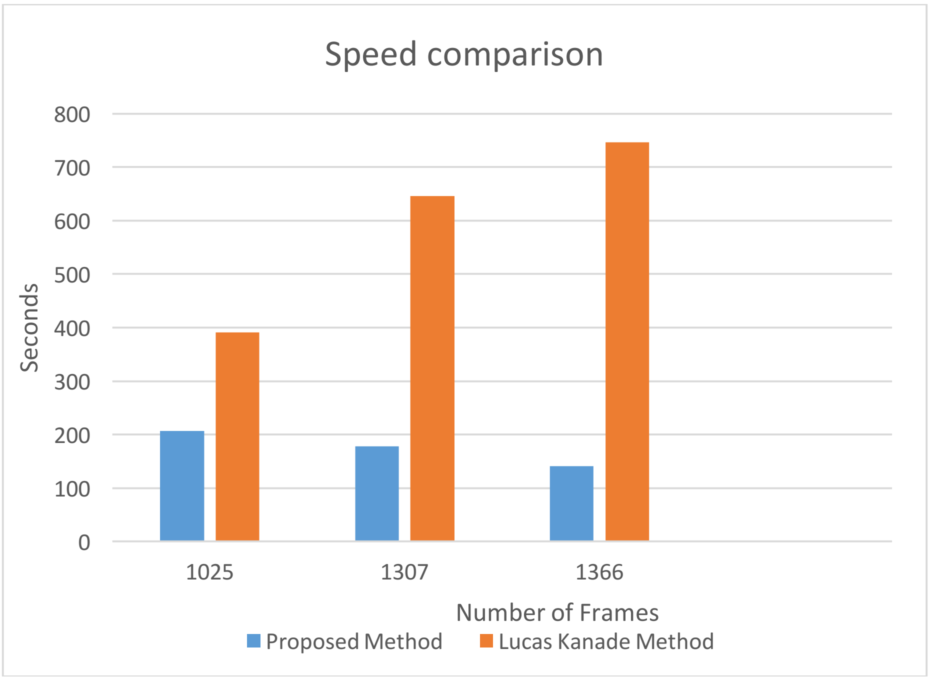


Figure 9 Demonstrates speed differences between prosed method & Lucas Kanade method

The result outputs are as follows:

The Lukas Kanade method starts off with Shi-Tomasi corner points at the corner regions in the image, as in Figure 10.

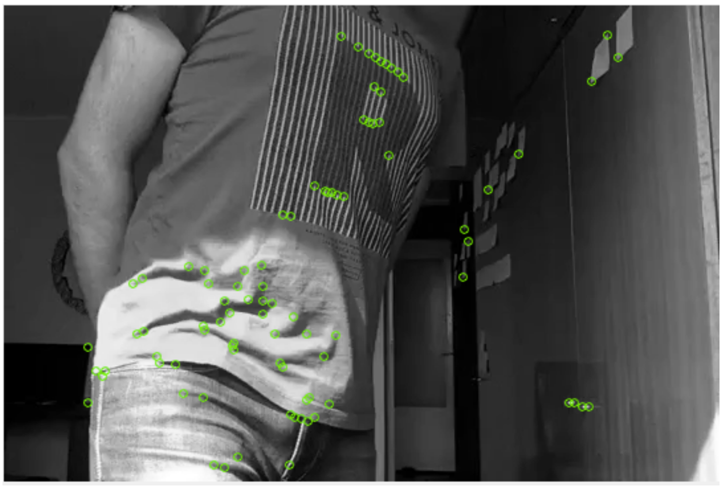


Figure 10 This image shows initial stage of Shi-Tomasi corner points

After certain frames, the corner points migrate to other corners to find image features to represent, because of the changes in patterns due to non-rigidity (as in Figure 11).

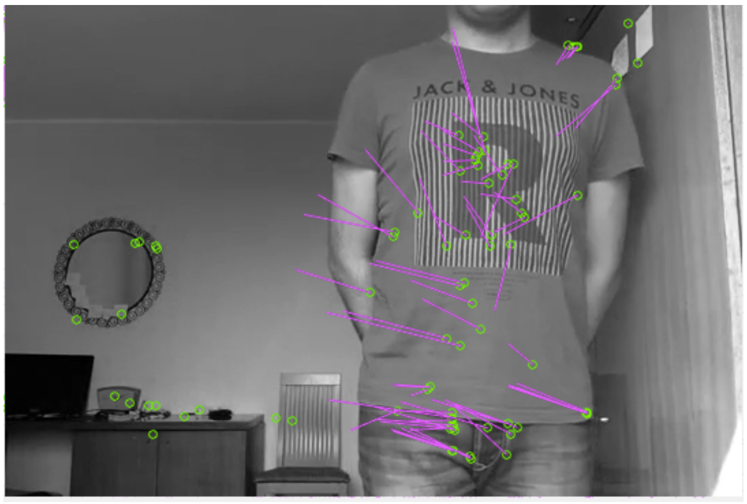


Figure 11 This image shows Shi-Tomasi corner points adapting due to changes in the image.

Whereas, the proposed method calculate SIFT points which are detected and matched in every frame, in order to track the pattern in every frame.

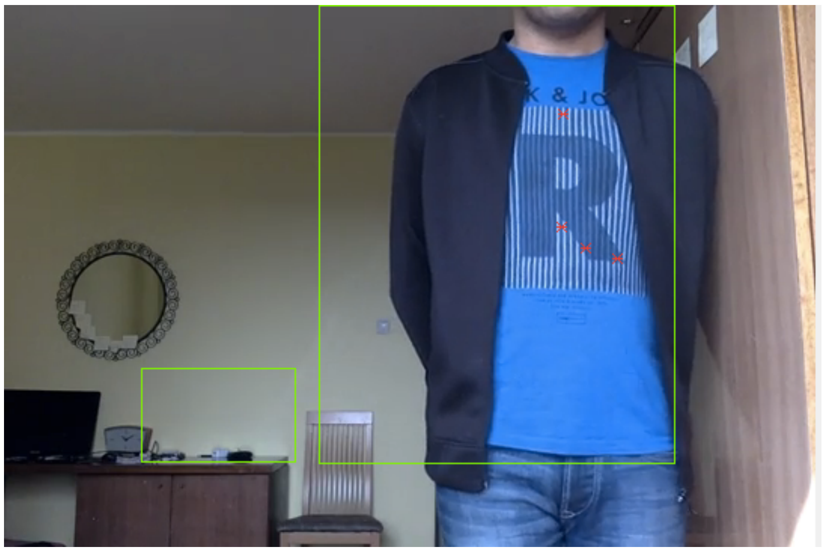


Figure 12 This image shows the pattern being recognized when partially occluded.

Motion detection is represented by the green rectangles in the image, and the recognition of keypoints with SIFT is represented by red asterisks.

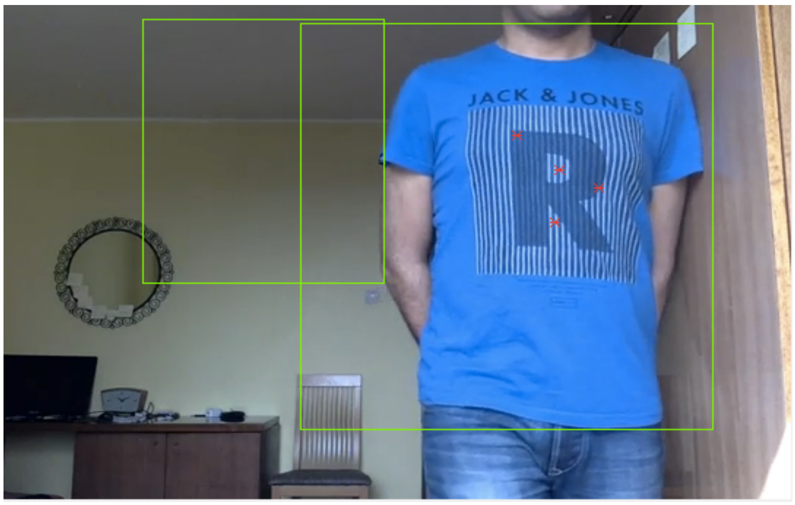


Figure 13 This image shows the pattern being recognized with no occlusions.

5. EVALUATION OF RESULTS

The experiments performed in section 4, demonstrate how the proposed method can perform with various parameters. The chosen parameters that can be adjusted allow the method to be dynamic or static with the given input to process.

The value of number of Gaussian mixture models is chosen as 2 since, with 2 chosen mixture models the video can select two surfaces. The video lacks variation in movements on multiple surfaces, hence the least possible number of Gaussian mixture models can observe the minimal motion where as with more than 2 mixture models the motion gets divided into unnecessary surfaces and the detection of motion is segmented. This segmentation reduces the chances of recognition.

The adaptive learning feature is enabled, as the method works best with adaptive learning when there are objects introduced into the video while it is running and as they remain for a longer time, the objects tend to become a part of the background.

Minimum background ratio helps guide the method to understand what ratio to expect between the background and the objects in motion. The default value 0.7 suggest that the object may consume approximately 30% of the pixels in the overall video. This fits best for the videos used for experiments.

The SIFT matching threshold is the restrict matches which are weak to be considered. The keypoint descriptors which do not lie close enough to each other, are reject with a high threshold value. With stronger matches, a reliable feature matching system can be created.

Domain Specific Evaluation

When motion in clothes, in most cases there is more than one motion appearing in a frame with clothes. This is due to the non-rigid nature of the object. The motion which needs to be detected here is the one with movements with large number of pixel displacement. The Lucas Kanade method is very specific and accurate in terms of detecting motions, which may be as minute as a single pixel. With the use of adaptive Gaussian mixture models to detect motion, the requirement is satisfied without the need of ignoring minute motions. With the involvement of the connected components algorithm, small motions can also be ignored by only connecting bigger foreground components.

With large number of motions taking place on clothes, the shape of embedded patterns also changes with respect to motions. With the use of keypoint and descriptors, a large number of features can be extracted and compared to perform matching. SIFT keypoints are scale invariant, which gives the possibility of extracting features which are more reliable than Shi-Tomasi corner points. With frequent motions occurring, SIFT keypoints still find matches since the points are valid under different scales, hence proving to be highly distinctive.

Considering an object, if the object is occluded partially, its corner may be lost. But, if the object's features are extracted which do not only lie on it's corner but also other parts of it, it would be more distinctive when recognizing it based on keypoints.

Due to the effect of non-rigidity in the object, the Shi-Tomasi corner points try to find distinctive features to extract and track, but with instability. With changing nature of the features, the corner points are either lost or moved another corner, demonstrating unreliability for this specific application.

The proposed method does not lose the moving feature to be tracked after the feature appears, in case of occlusions. Since the method actively scans the moving foreground for matches. This helps in distinguishing between the moving objects to be tracked and the objects which not to be tracked but are in motion.

Also the proposed method proves to consume less time in processing of videos as compared to Lucas Kanade pyramidal method.

Metric based evaluation

Based on the metrics, the approximated results state that the proposed method has an accuracy of 79% with the tracker detection rate of 54%. These results are low due to the unpredictability of the adaptive Gaussian mixture models based tracking.
But the recognition method influences the tracking in a positive way, as the overall method has a false alarm rate of only 3%, false positive rate of 1% and a 97% positive prediction.
The proposed method proves to be robust when detecting patterns in motion, but with a possibility of many motions remaining undetected. To generalize this method for wider applicability, it can be integrated with classifiers such as support vector machines, k-means, convolutional neural networks, etc

6. CONCLUSIONS

1. In separating the background and the foreground with use of adaptive Gaussian mixture models, the number of mixture models used plays a significant role in detecting motions, due to the method's concept of surfaces with varying pixel values and the impact of adaptive learning.

2. SIFT is a robust method for matching features, and also works well when the object to be recognized is partially occluded, with a high enough matching threshold.

3. The combination of adaptive Gaussian mixture models with SIFT method, processes videos faster than the pyramidal implementation of the Lukas and Kanade method for tracking.

4. The combination of the two methods, has a low tracker detection rate but a high positive prediction with a low false alarm rate.

5. This thesis accomplishes it's aim to analyse methods, propose a new method and evaluate it based on experimentation and results.

7. REFERENCES

[1]. Yilmaz, Alper, Omar Javed, and Mubarak Shah. "Object Tracking". *CSUR* 38.4 (2006): 13-es. Web.

[2]. Yamaguchi, Kota and Hadi Kiapour. "Parsing Clothing In Fashion Photographs". (2012): n. pag. Web. 3 Mar. 2016.

[3]. Kalantidis, Yannis and Lyndon Kennedy. "Getting The Look: Clothing Recognition And Segmentation For Automatic Product Suggestions In Everyday Photos". (2012): n. pag. Print.

[4]. Stauffer, Chris and W.E.L. Grimsson. "Adaptive Background Mixture Models For Real-Time Tracking". *CVPR98* (1998): n. pag. Print.

[5]. Horn, Berthold K., and Brian G. Schunck. "Determining optical flow." *1981 Technical symposium east*. International Society for Optics and Photonics, 1981.

[6]. Bouguet, Jean-Yves. "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm." *Intel Corporation* 5.1-10 (2001): 4

[7]. Black, Michael J. and P. Anandan. "The Robust Estimation Of Multiple Motions: Parametric And Piecewise-Smooth Flow Fields". *Computer Vision and Image Understanding* 63.1 (1996): 75-104. Web.

[8]. Szeliski, Richard, and James Coughlan. "Spline-based image registration."*International Journal of Computer Vision* 22.3 (1997): 199-218.

[9]. E.H. Andelson and C.H. Anderson and J.R. Bergen and P.J. Burt and J.M. Ogden. "Pyramid methods in image processing". 1984.

[10]. Lowe, David G. "Distinctive image features from scale-invariant keypoints."International journal of computer vision 60.2 (2004): 91-110.

[11]. Shi, Jianbo, and Carlo Tomasi. "Good features to track." Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on. IEEE, 1994.

[12]. C. Harris and M. Stephens (1988). "A combined corner and edge detector". Proceedings of the 4th Alvey Vision Conference. pp. 147–151.

[13]. Sinha, Utkarsh. "The Shi-Tomasi Corner Detector - AI Shack - Tutorials For Opencv, Computer Vision, Deep Learning, Image Processing, Neural Networks And Artificial Intelligence.". *Aishack.in*. N.p., 2016.

[14]. "Computer Vision System Toolbox - MATLAB & Simulink". *Se.mathworks.com*. N.p., 2016.

[15]. "Vlfeat - Home". *Vlfeat.org*. N.p., 2015. Web. 23 May 2016.

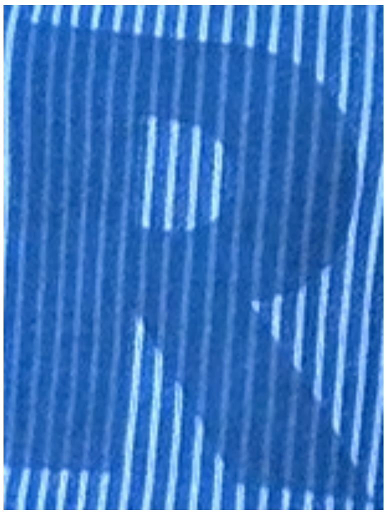
[16]. Bashir, Faisal, and Fatih Porikli. "Performance evaluation of object detection and tracking systems." *Proceedings 9th IEEE International Workshop on PETS*. 2006.

8. APPENDIX

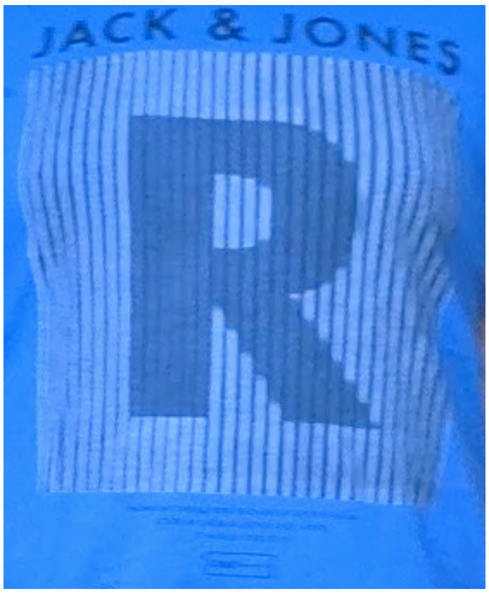
A. Images from training and raw video.

(a)

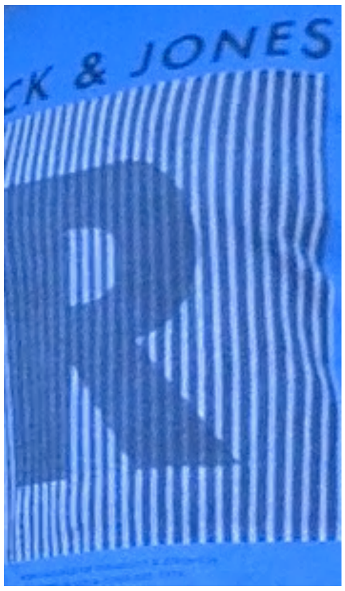
(b)



(c)



(d)



(e)

Figure 14 Figures (a)(b)(c)(d)(e) are training images for SIFT comparison.

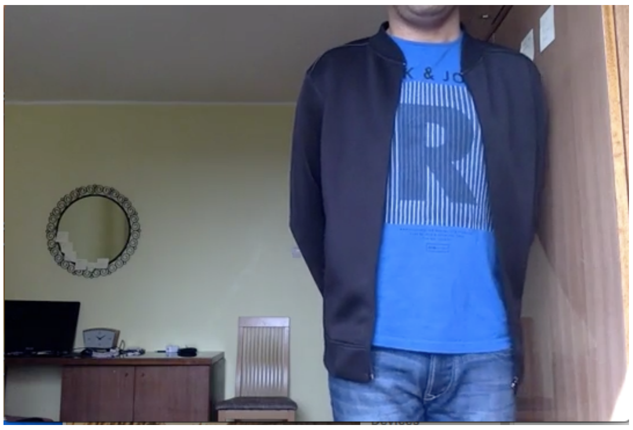


Figure 15 The figure shows video-frame with occluded pattern, without tracking and recognition

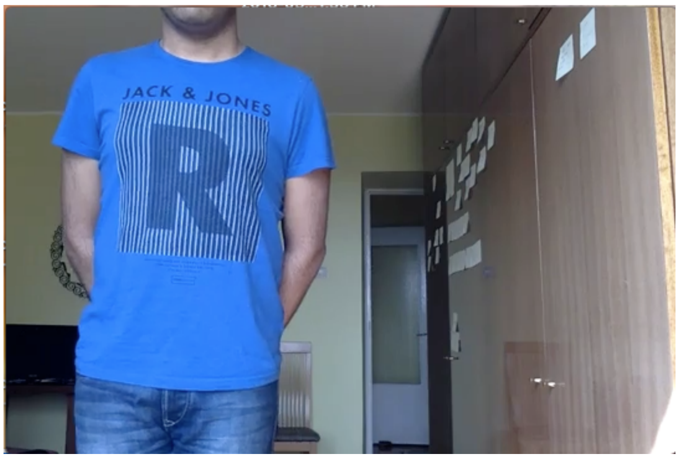


Figure 16 The figure shows video-frame with visible pattern, without recognition.

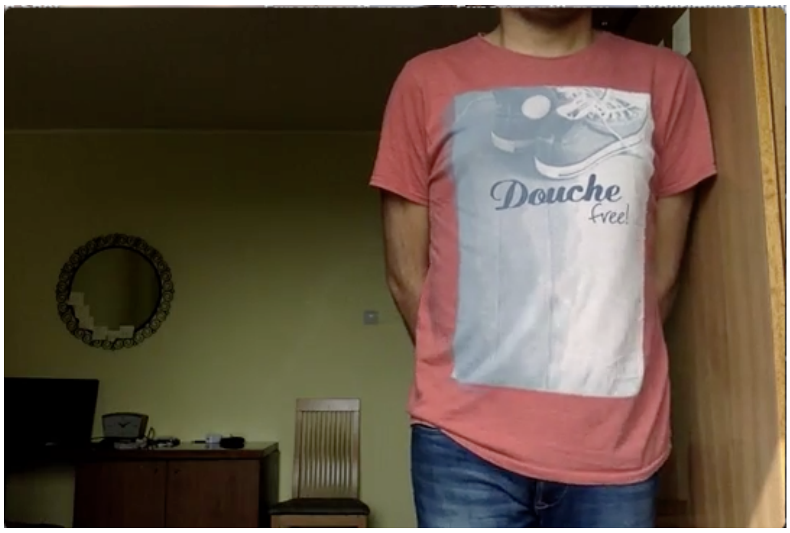


Figure 17 The figure shows video-frame with missing pattern, without recognition.

B. Code

The code requires, Computer Vision toolbox and VL Feat Library.

```
disp('New method start')
```

```
dt = datestr(now,'HH:MM:SS.FFF AM')

foregroundDetector = vision.ForegroundDetector('NumGaussians', 2, ...
   'NumTrainingFrames', 150, ...
   'MinimumBackgroundRatio', 0.7);

videoReader = vision.VideoFileReader('Experiment3.mp4');
for i = 1:100
   frame = step(videoReader);
   foreground = step(foregroundDetector, frame);
end


%%

se = strel('square', 3);
filteredForeground = imopen(foreground, se);

%%
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
   'AreaOutputPort', false, 'CentroidOutputPort', false, ...
   'MinimumBlobArea', 2000);
%%
   interestPic = imread('TrainingImage1.png');
   [Ft, Ds] = vl_sift(im2single(rgb2gray(interestPic)));
   interestPic1 = imread('TrainingImage2.png');
   [Ft1, Ds1] = vl_sift(im2single(rgb2gray(interestPic1)));
   interestPic2 = imread('TrainingImage3.png');
   [Ft2, Ds2] = vl_sift(im2single(rgb2gray(interestPic2)));
   interestPic3 = imread('TrainingImage4.png');
   [Ft3, Ds3] = vl_sift(im2single(rgb2gray(interestPic3)));
   interestPic4 = imread('TrainingImage5.png');
   [Ft4, Ds4] = vl_sift(im2single(rgb2gray(interestPic4)));

thres = 6; %SET THRESHOLD FOR SIFT HERE
%%
videoPlayer = vision.VideoPlayer('Name', 'GMMSIFT');
videoPlayer.Position(3:4) = [650,400];  % window size: [width, height]
se = strel('square', 3); % filter for noise removal

counter = 0;
totalMatches = 0;
totalDetections = 0;
totalFramesWithMatches = 0;
while ~isDone(videoReader)

   frame = step(videoReader); % read the next video frame

   % Detect the foreground in the current video frame
   foreground = step(foregroundDetector, frame);

   % remove noise in the foreground
   filteredForeground = imopen(foreground, se);

   % Detect the connected components with the specified minimum area, and
   % compute their bounding boxes
   bbox = step(blobAnalysis, filteredForeground);

   currentMatches = 0;
   currentDetections = 0;

   pos = [];
```

```
if(~isempty(bbox))
    %bbox
    currentMatches = 0;
    currentDetections = 0;
   [Brows, Bcols] = size(bbox);
   currentDetections = Brows;
   totalDetections = totalDetections + currentDetections;

   for b=1:Brows
       %b

       movingObjs = frame(bbox(b,2):bbox(b,2)+bbox(b,4)-5,bbox(b,1):bbox(b,1)+bbox(b,3));


       [Ftm, Dsm] = vl_sift(im2single(movingObjs));

       [MATCHES,SCORES] = vl_ubcmatch(Ds, Dsm, thres);
       [Mrows, Mcols] = size(MATCHES);

       if(~isempty(MATCHES))

           for i=1:Mcols
               posn = [Ftm(1,MATCHES(2,i))+bbox(b,1) Ftm(2,MATCHES(2,i))+bbox(b,2);];
               pos = [pos; posn];
           end
           round(pos);
       end

       [MATCHES1,SCORES1] = vl_ubcmatch(Ds1, Dsm, thres);
       [Mrows1, Mcols1] = size(MATCHES1);
       if(~isempty(MATCHES1))

           for i=1:Mcols1
               posn = [Ftm(1,MATCHES1(2,i))+bbox(b,1) Ftm(2,MATCHES1(2,i))+bbox(b,2);];
               pos = [pos; posn];
           end
           round(pos);
       end

       [MATCHES2,SCORES2] = vl_ubcmatch(Ds2, Dsm, thres);
       [Mrows2, Mcols2] = size(MATCHES2);
       if(~isempty(MATCHES2))
           %Ftm%MATCHES
           for i=1:Mcols2
               posn = [Ftm(1,MATCHES2(2,i))+bbox(b,1) Ftm(2,MATCHES2(2,i))+bbox(b,2);];
               pos = [pos; posn];
           end
           round(pos);
       end
       %4
       [MATCHES3,SCORES3] = vl_ubcmatch(Ds3, Dsm, thres);
       [Mrows3, Mcols3] = size(MATCHES3);
       if(~isempty(MATCHES3))
           %Ftm%MATCHES
           for i=1:Mcols3
               posn = [Ftm(1,MATCHES3(2,i))+bbox(b,1) Ftm(2,MATCHES3(2,i))+bbox(b,2);];
               pos = [pos; posn];
           end
           round(pos);
       end
       %5
       [MATCHES4,SCORES4] = vl_ubcmatch(Ds4, Dsm, thres);
```

```
            [Mrows4, Mcols4] = size(MATCHES4);
            if(~isempty(MATCHES4))
                %Ftm%MATCHES
                for i=1:Mcols4
                    posn = [Ftm(1,MATCHES4(2,i))+bbox(b,1) Ftm(2,MATCHES4(2,i))+bbox(b,2);];
                    pos = [pos; posn];
                end
                round(pos);
            end

            %RESULTING MATCHES
            currentMatches = Mcols + Mcols1 + Mcols2 + Mcols3 + Mcols4;

            if currentMatches>0
                totalFramesWithMatches = totalFramesWithMatches +1;
            end

            totalMatches = totalMatches + currentMatches;

        end


     end

    % Draw bounding boxes
    result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');
    if(~isempty(pos))
        result = insertMarker(result, round(pos),'*', 'Color', 'red');

    end

    step(videoPlayer, result);  % display the results

%     disp('counter');
%     disp(counter);
%     disp('currentMatches');
%     disp(currentMatches);
%     disp('currentDetections');
%     disp(currentDetections);
    counter = counter + 1;
end

release(videoReader); % close the video file
disp('New method end')
dt = datestr(now,'HH:MM:SS.FFF AM')


disp('totalMatches');
disp(totalMatches);
disp('totalDetections');
disp(totalDetections);
disp('totalFramesWithMatches');
disp(totalFramesWithMatches);
```