NORTH SOUTH UNIVERSITY

UNDERGRADUATE THESIS

Robust Fusion Tracking

Author:
Md. Rasheduzzaman
Md. Amirul Islam

Supervisor:
Mirza Mohammad Lutfe Elahi

A thesis submitted in fulfilment of the requirements for the degree of The Bachelor of Computer Science

in the

Electrical Engineering and Computer Science Department
North South University

October 2013

Declaration of Authorship

We are, Md. Rasheduzzaman and Md. Amirul Islam, declare that this thesis titled, Robust Fusion Tracking and the work presented in it are our own. We confirm that:

- This work was done wholly or mainly while in candidature for a undergraduate degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- We have acknowledged all main sources of help.
- Where the thesis is based on work done by ourselves jointly with others, We have made clear exactly what was done by others and what we have contributed myself.

Signed: ____

Date: ____

The Chairman
Department of Electrical Engineering and Computer Science
North South University
Bashundhara, Dhaka
Date: October 10, 2013

Sir,

With due respect we are, Md. Rasheduzzaman (ID: 0910647040) and Md. Amirul Islam (ID: 0920507042) would like to state that, we have completed our thesis project of CSE 499 which is titled "Robust Fusion Tracking" under the supervision of Mirza Mohammad Lutfe Elahi. We are submitting our report.

Therefore hereby declare that no part of this project was plagiarized or borrowed and all the referenced materials are cited accordingly. Therefore, we would be grateful if you would kindly accept the submission of our report.

Sincerely,
Md. Rasheduzzaman (ID: 0910647040)

Md. Amirul Islam (ID: 0920507042)

NORTH SOUTH UNIVERSITY

Abstract

Mirza Mohammad Lutfe Elahi
Electrical Engineering and Computer Science Department

Undergraduate Thesis

Robust Fusion Tracking

by
Md. Rasheduzzaman
and
Md. Amirul Islam

Our thesis addresses the problem of long-term object tracking in video sequences. The tracked object is defined by it's position in the first frame, the goal is to track this object in a video where it changes shape, gets partially occluded . This task is crucial for many real life applications. This method consists of object tracking, detecting and fusion. These three sub-systems are connected in one functional system, which addresses problem of long-term object tracking. The goal is to implement novel, but robust fusion tracking system in C++ programming language. First experiment tries to find out the best key-point descriptor for Lucas-Kanade tracking algorithm, where we device three different experiment. We evaluate 9 different state-of-the-art key-point detector algorithm using 4 challenging image sequences which are publicly available. We use the most challenging sequences available in online and widely used by many other tracking algorithm for evaluation. Second experiment tests the complete tracking method for purposes of online detector learning. Result of experiments show that proposed fusion tracking functionality improve tracking performance in various subsets of testing sequences. Experiment provides comparison with result published in "Robust Object Tracking with Online Multiple Instance Learning [1]. Outcome of this thesis is an introduction to problem of long-term object tracking, documentation of implemented fusion tracking system extended by new object tracking method.

Acknowledgements

We would like to thank our supervisor, Mirza Mohammad Lutfe Elahi, for the guidance and countless numbers of advice he has provided. We would also like to thank our good friend Mahabub Akram for helping us with comprehension of technical thesis related issues. Finally we thanks our family for supporting us during studies. . . .

Contents

Declaration of Authorship **i**

Letter **ii**

Abstract **iii**

Acknowledgements **iv**

List of Figures **vii**

List of Tables **viii**

Abbreviations **ix**

Symbols **x**

1 Introduction **1**
- 1.1 Problem Definition 3
- 1.2 Related Work 4
- 1.3 Scope of Work 5
- 1.4 Contribution 6
- 1.5 Organization 7

2 Tracking **8**
- 2.1 Lucas-Kanade Tracking 9
- 2.2 Median Flow Tracker 12
- 2.3 Point Detectors 13
  - 2.3.1 Harris Corner Detection 14
  - 2.3.2 Good Feature Track 14
  - 2.3.3 SIFT 15
  - 2.3.4 SURF 16

2.3.5 MSER . . . 16
2.3.6 ORB . . . 16
2.3.7 FAST . . . 17

3 Detection **18**
3.1 Radial Based Approach . . . 20
3.2 Variance Filter . . . 20
3.2.1 Integral Image . . . 21
3.3 Machine Learning . . . 23
3.3.1 Haar-like Features . . . 24
3.3.2 Naive Bayes Classification . . . 25

4 Fusion **29**
4.1 Template matching . . . 29
4.2 Fusion . . . 31
4.3 Main Loop . . . 31

5 Experiment **33**
5.1 Evaluation Methodology . . . 33
5.2 Experiment on Key-points . . . 34
5.3 Experiment on MIL Data-set . . . 38
5.3.1 Sylvester and David indoor . . . 38
5.3.2 Occluded Face and Cola . . . 38
5.3.3 Tiger . . . 39
5.3.4 Dollar . . . 39

6 Conclusion **42**
6.1 Conclusion . . . 42

A Appendix **44**
A.1 Variance . . . 44

Bibliography **45**

List of Figures

1.1 An object is selected by a bounding box. 2
1.2 An example of clutter. 3
1.3 Illustration of the Tracking system. 6

2.1 Process of Lucas-Kanade Tracking. 10
2.2 Forward-Backward Tracking. 11
2.3 Process of FAST key-point method. 17

3.1 Sub-window Searching. 19
3.2 Illustration of detection process. 19
3.3 An example of integral image calculation. 22
3.4 An example of variance filter. 23
3.5 An example of Haar-like features. 25
3.6 Diagram of naive Bayes classification. 26

4.1 Find template in image. 30
4.2 An example Normalized Cross-Correlation and Coordinates of Peak. . . . 30

5.1 First frame of test image sequences 34
5.2 Key-points found on test image sequences 35
5.3 Forward-Backward error on test image sequences 36
5.4 Center location error on test image sequences 37
5.5 Ouput of MIL Track and Fusion tracking 40
5.6 Precision plot of test image sequences 41

List of Tables

5.1 Center location error 39

Abbreviations

LK	**L**ucas **K**anade
TLD	**T**racking **De**tection **L**earning
MIL	**M**ultiple **I**nstance **L**earning
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SURF	**S**peeded **U**p **R**bust **F**eatures
GFTT	**G**ood **F**eature **T**o **T**rack
MSER	**M**aximally **S**table **E**xtremal **R**egions
ORB	**O**riented **BRIEF**
NCC	**N**ormalized **C**ross **C**orrelation

Symbols

I	Image
T	Template
W	Patch
A	auto-correlation
H	Harris matrix
P	probability
ΔX	Forward difference
∇X	Backward difference
$\sum X$	Summation of value X
∂X	partial derivative of X
ϵ	error threshold

1 Introduction

Object tracking is an important task within the field of computer vision. The availability of high quality and inexpensive video camera, high powered computers, and the increasing need for automated video analysis have generated a great deal of interest in object tracking algorithms. There are three main steps in video analysis: tracking of object from frame to frame, detection of interested moving objects in the frame, and analysis of object tracks to determine their behavior. Therefore, the use of object tracking is applicable in the tasks of: motion based recognition, video indexing, human-computer interaction, traffic monitoring, vehicle navigation etc. Tracking can be defined as the problem of esteeming the trajectory of an object in the image plane as it moves around a scene. In its simplest from, track an object from one frame to the next frame. The object is defined by a bounding box in a frame and our objective is to find out the bounding box in the next frame or indicate that the object is not visible in the next frame. Tracking objects can be complex due to noise in the image, complex object motion, partial and full object occlusions, complex object shapes, real-time processing requirements etc. Tracking can be simplified by imposing constraints on the motion or appearance of the object. Most of the tracking algorithms assume that the object motion is smooth with no unexpected changes. Prior knowledge about the size and shape of the object can also be used to simplify the tracking problem. There are numerous approaches for object tracking have been proposed. These methods are different from each other based on the way they approach. Selecting the right features plays a critical role in tracking. In general, the most desirable property of a visual feature is its uniqueness so that the objects can be easily distinguished in the feature space. Feature selection is related to the object representation. For example, color is used as a feature for histogram based appearance representations, while for contour-based representations, object edges are usually used as features. Most features are chosen manually by users. In general, many tracking algorithms use a combination of these features. In this paper we proposed a new approach to the Long term tracking method which is used to solve the problem of tracking object with minimum prior information. Minimum prior information means, initially the object is selected by the user but this object is not known to the user. So all the information related to the object ,which is selected initially, must be retrieved from the footage processing. This method is decomposed into three components: tracking, learning, and detection. The initial object is tracked by an adaptive tracker which follow the Lucas-Kanade method. This tracker follow the object from frame to frame. The tracker is corrected by the detector when the detector localizes all the appearance observed so far. The detector's error are indicated by the learning component and updates all the errors in the next frame. Tracking failures are detected by the forward-backward error detection method and then follow the median flow method for object tracking. The median flow tracker takes a pair of image and a bounding box in the image. The points inside the bounding box are tracked by the median flow tracker and tracking errors are indicated, outliers are filtered out from the bounding box.

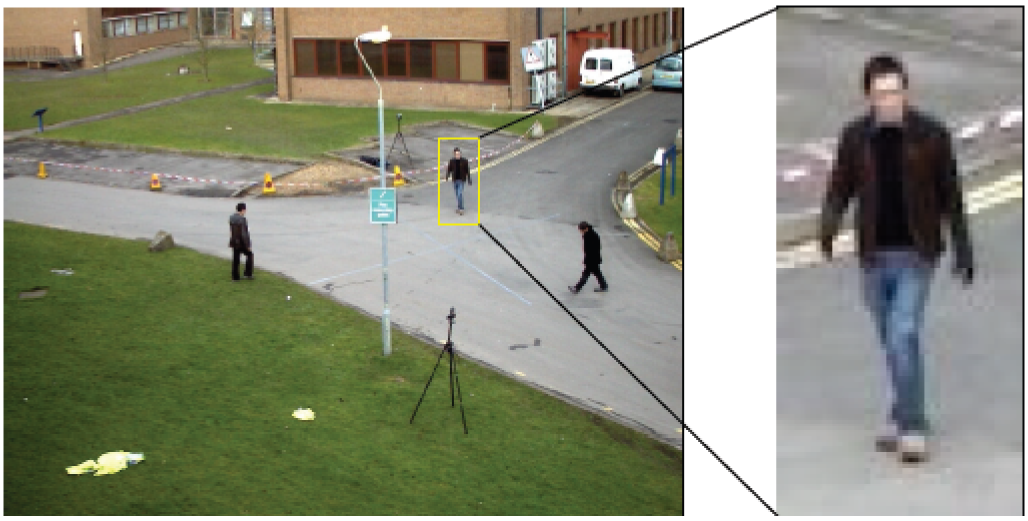


Figure 1.1: An object is selected by a bounding box.

1.1 Problem Definition

In this work we primarily focuses on semi-supervised short time single-tracking object. Consider a image with sequences $I_1, I_2, ..., I_n$. For each frame I_k estimate the stage X_k of the target. State X_k is predetermined as centroid, bounding box, bounding ellipses in the method of object tracking. In case of figure 1.1, in the image we consider a single object and a boundary box is shown around the object we are interested in. Considering the image, all the parameters of X_k consist of the upper left corner of the extracted rectangle(whose length and width is x, y respectively), and the length and width of the rectangle. Interaction with the user is required in each frame if we consider the manual tracking, whereas, priori information is needed in case of automated tracking. But if we consider semi-automated tracking then, user input is needed to initialize the tracking process automatically.

One of the main challenge in object tracking is clutter. Clutter is basically some observed facts or a situation where the features expected from the object of interest are difficult to differentiate from the features which are not from the object of interest. In some images several objects are present in the same place so it's tough to extract the exact feature from the object of interest. In figure 1.2, an example of clutter is shown in which several cars are present in the same place. So when the object

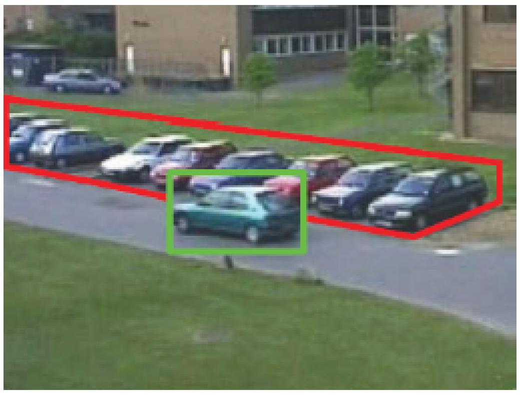


FIGURE 1.2: An example of clutter.

of interest is the car that is surrounded by the bounding box, there is a chance that some features from the car which is just behind it will be extracted. So. there will be contradiction to differentiate the exact feature. A proper method is needed to re-detect the object independently.

1.2 Related Work

Kalal Z.[2] explore the long term tracking of unknown objects in a video frame where objects are identified by its location in a single frame. Their task is to track each frame and determine whether object is present or not. Finally, they propose a noble method which is consisted of three parts: Tracking-Learning-Detection (TLD). Function of the tracker is to follow the object frame to frame whereas the detector initially focuses on all the appearances that have been observed so far and finally correct the tracker if needed. The learning part uses a P-N experts method which basically completes the whole tracking system by estimating the detectors error and it also updates all the errors so that the detector can avoid those errors in the next frame when it detect the tracked object. Zhang K., Zhang L. & Yang M.H. [3] proposed a method for real time compressive tracking. They basically come up with an efficient tracking algorithm with an appearance model which is based on features extracted from the multi-scale image feature space. Their model employs non adaptive random projections and a very sparse measurement matrix is adopted to extract the features for the appearance model. Li H., Shen C. & Shi Q. [4] propose real time compressive sensing tracking by developing the signal recovery power of compressive sensing (CS). In their approach OMP(Orthogonal Matching pursuit) are adopted to accelerate the CS tracking and their algorithm achieves a real time speed that is 5000 times faster than the conventional tracker. Babenko B., Yang M. & Belongie S. [1] actually address the problem of tracking with minimum prior information. Initially they follow the method of 'Tracking by detection' and find some limitations or inaccuracies related to the classifier which used in this method. Further they show that using Multiple instance learning (MIL) instead of conventional supervised learning avoids these limitation and can therefore come up with a more robust tracker. Finally they propose the MIL algorithm for tracking object which achieves better result in real life performances. Avidan S. [5] proposed a ensemble tracking to differentiate between the object and the background. They used the AdaBoost algorithm to combine weak classifier into strong classifier. Then the strong classifiers are used to label a pixel in the next frame as either belonging to the object or the background, giving a confidence map. Yang F., Lu H. & Chen Y. [6] propose a visual tracking method based on "Bags of Features"(BaF) algorithm. They use RGB LBP features instead of just one codebook in traditional BoF. They combine path based approaches and global template based approach into a unified framework and finally get a robust method for handling occlusions, scaling and rotation. Grabner H., Leistner C. & Bischof H. [7] work on the drifting problem in tracking applications and they proposed a noble online semi-supervised method for addressing this problem. Using SemiBoost tracker they demonstrate real time tracking on several challenging test sequences where this method do better than other online tracking methods. Godec M., Grabner H. & Bischof H. [8] focuses on almost similar approaches as [7] . They just added a more sophisticated search-space sampling, and an improved update sample selection. Also a review of semi-supervised online boosting algorithm is given. Adam A., Rivlin E. & Shimshoni. Paper [3] present a noble algorithm which is called ”Frag-Track” for tracking an object in a video sequence. In this process, the histogram of the image patch of current frame is compared to the histogram of the corresponding image patch. This algorithm come up with proper solutions of some difficulties which can't be handled in many histogram based approach. Finally they found extensive experimental results which demonstrate robust tracking. Santner J. , Leistner C. , Saffari A. , Pock T.& Bischof H. [9] focus on some limitations of [7]. They combine three trackers(a simple template model as a non-adaptive, a novel optical flow based mean-shift tracker, and an online random forest as moderately adaptive appearance based learner) in a cascade and all of the components run on GPUs or similar multi-core systems. They justified their method over current state-of-art method and find superior result. Yu Q. , Dinh T. B. & Medioni G. [10] discusses initially about the limitations of visual tracking. They propose a co-training based approach to continuously label incoming data and online update a hybrid discriminative generative model. They show that this method has strong ability and robustness to distracters in background.

1.3 Scope of Work

We use the approach of Kalal et al.[2] for recursive tracking initially. Then perform compressive tracking to get better prediction and to condense the execution time of the tracker. The Kalal et al. [2] approach is based on the method of Lucas Kanade [11–13]. In the detection part we use the approach of [1]. In every frame of the video the detector is evaluated. All the sub-windows searched by the detector are evaluated in two step filter. First, a variance filter is applied on each sub-windows. Sub-window that has variance value greater than a certain threshold are pass down to the second stage filter.

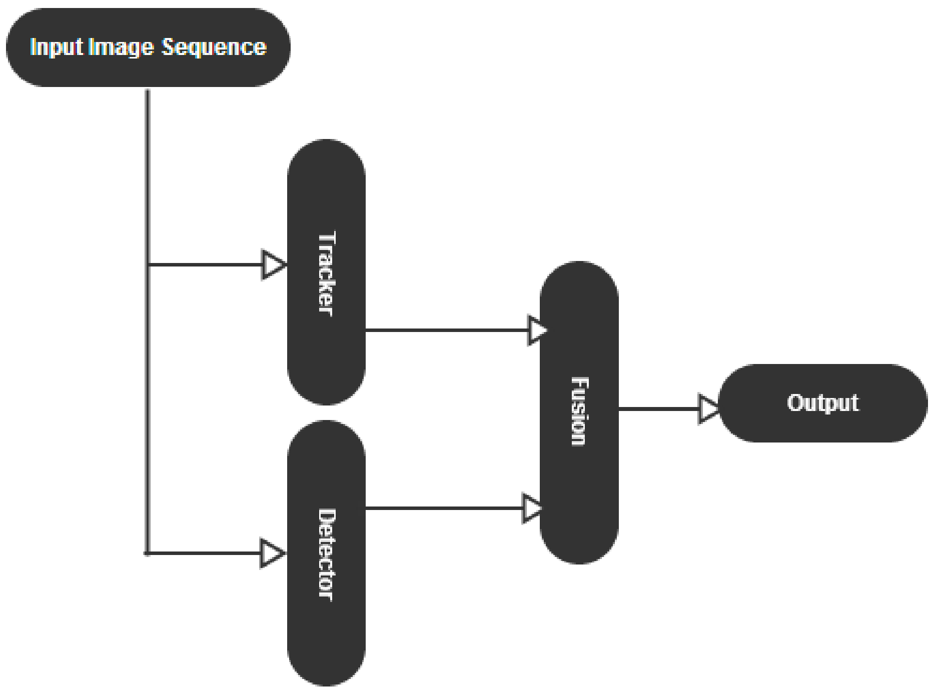


Figure 1.3: Illustration of the Tracking system.

In the second stage a naive Bayes classifier classify the remaining sub-window and finds the best sub-window. Then a template matching method is used to combine these two output of tracker and detector and finds the best output. A training set is formed by all the estimated errors in each frame. The detector is retrained in each frame to avoid these errors.

1.4 Contribution

In our project we initially follow the Tracking-Learning-Detection approach. We added compressive tracking part to increase the efficiency of the tracker. To avoid the difficulty with clutter we proposed a method which can easily detect a clutter situation. Execution time is reduced by using different features to test which one gives better prediction. We implement our project in C++(using OpenCV 2.4.6) and test it in sample data.

1.5 Organization

The Paper is organised as follows. In Chapter 2, the tracking method based on the estimation of optical flow is described. In Chapter 3, the cascaded approach to object detection is explained. Chapter 4 describes the combining process of tracker and detector. Chapter 5 shows experimental results on test data as well as a comparison to other tracking methods. Chapter 6 gives a conclusion and final remarks.

2 Tracking

In this chapter we introduce a recursive tracking method for tracking object of interest. No prior information without the location of the object in the previous frame is required about the object. We just initialize the object in the first frame and track the object in the consecutive frames. We follow the approach of Kalal et al. for recursive tracking. The tracking is performed on an input bounding box, where the underlying patch is tracked from one camera frame and to the next. This is done by computing the optical flow (pixel displacement from one frame to the next) with the Lucas-Kanade(LK) algorithm on the pixels belonging to the patch which is a uniform grid extracted from the patch. To filter out track points which are likely to be erroneous, we use information from the Lucas-Kanade method as well as different error measures based on Normalized cross correlation and forward-backward error.

We also consider here how to select the feature points in an image from the set of image sequences, such that further they can be tracked well through the sequences. Usually, tracking is performed by some sort of local search methods searching for a similar patch in the next image from the sequence. For the Lucas-Kanade (LK) tracking method we try to find which feature points descriptor is better.

Chapter 1 organized as follows: Sec. 2.1 formulate Lucas-Kanade tracker. Section 2.2 describes the median flow tracker used in the project. Finally, Sections 2.3 describes the point descriptor used as the parameter to find the best key-point detector.

2.1 Lucas-Kanade Tracking

The Lucas-Kanade(LK) tracker was first introduced in 1981, at International Joint Conference of Artificial Intelligence. It is a very popular computer vision algorithm for tracking. Many state-of-the-art tracking system has been developed by integrating other techniques with LK tracker. Lucas-Kanade (LK) tracker is iterative differential method for optical flow estimation. Optical flow is the pattern of apparent motion of objects in a scene caused by the relative motion between an observer and the scene. For two given images I(x) and J(x) and a template image T (x) in image I(x), LK tracker estimate position of this template T(x) in image J. Where image I and J are grayscaling images. The image I will sometimes be referenced as the first image, and the image J as the second image. First, we select an image patch template $T(x)$ from image I at time t. Here x is column vector of coordinates in image $[x, y]^T$. LK measure the similarity using square of $L2$ norm, which defines quadratic error between images. Error minimization is done by using iterative gradient method. Using the gradient method the window $I(W(x; p))$ shifted in such a way in each iteration such that the quadratic error between patch template and input patch is minimized. The wrap matrix $W(x; p)$ can be expressed as:

$$W(x; p) = \begin{pmatrix} x + p1 \\ y + p2 \end{pmatrix} \tag{2.1}$$

Here $p1$ and $p2$ can be considered as the optical flow along x and y respectively. The optimal window shift $P*$ minimizes dissimilarity of the patch template and input patch. The Lucas-Kanade algorithm will minimize the sum of squared difference between the template T and the image I:

$$\sum_x (I(W(x; p)) - T(x))^2 \tag{2.2}$$

Consider the best shift is Δp, from Eq.(2.1) we get

$$\sum_x (I(W(x; p + \Delta p)) - T(x))^2 \tag{2.3}$$

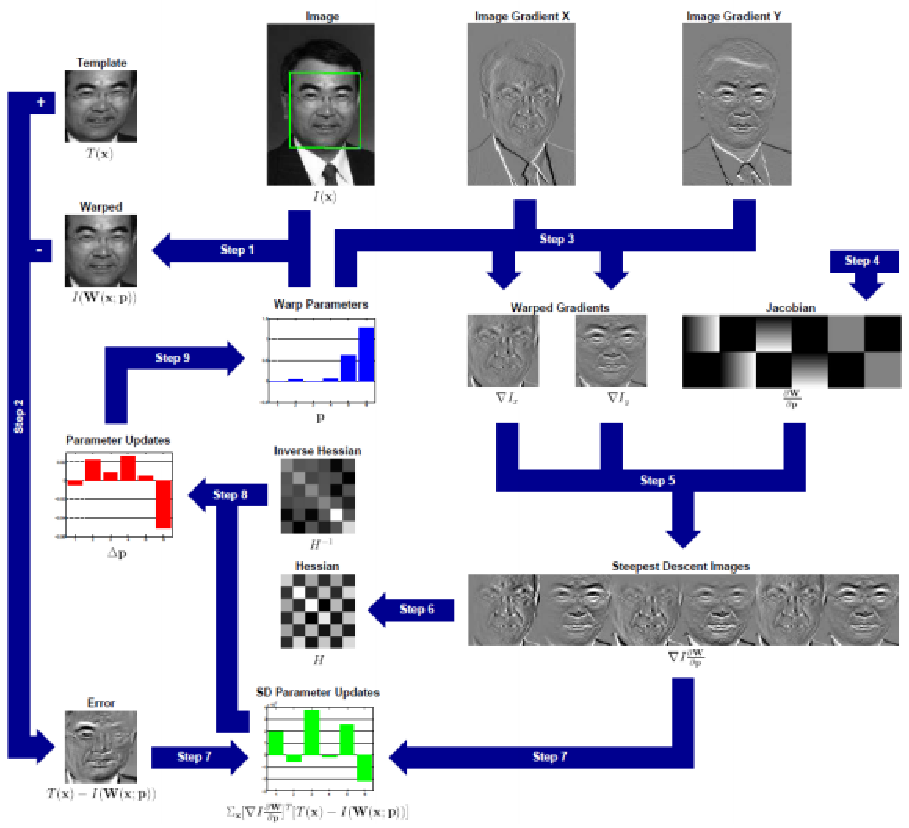


FIGURE 2.1: Process of Lucas-Kanade Tracking.

We minimize the expression with respect to Δp. Nonlinear expression Eq.(2.2) is linearized using Taylor series of first order to

$$\sum_x \left(I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right)^2 \tag{2.4}$$

where $\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right]$ is gradient in image estimated from $W(x;p)$. Term $\frac{\partial W}{\partial p}$ is Jacobian matrix (symmetric matrix of first partial derivation) of translation in the direction x and y. Resulting Jacobian matrix is

$$\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} = \frac{\partial \left[x + p_x; y + p_y \right]^T}{\partial \left[p_x; p_y \right]} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.5}$$

To find the minimum of Eq.(2.4) we use partial derivation of Δp. After that we explicitly express the shift

$$\Delta p = H^{-1} \sum_{x} \left[\nabla I \frac{\partial W}{\partial p} \right]^{T} [T(x) - I(W(x;p))] \tag{2.6}$$

Where H is approximation of Hessian matrix used in Gaussian-Newton gradient method. It is necessary to tell that Gaussian-Newton method solve the non-linear regression and minimizes expression. Note that the approximation of Hessian matrix is in this non-linear regression is equal to autocorrelation matrix (Harris matrix).

$$H = \sum_{x} \left[\nabla I \frac{\partial W}{\partial p} \right]^{T} \left[\nabla I \frac{\partial W}{\partial p} \right] \tag{2.7}$$

Use of expression Eq.(2.4) in Eq.(2.5) and Eq.(2.6) simplifies $\nabla I \frac{\partial W}{\partial p} = \nabla I$. Eq.(2.5) is computed in each iteration and for next iteration is the overall shift update.

$$p \leftarrow p + \Delta p \tag{2.8}$$

Convergence can be terminated by setting the maximal number of iteration steps or by condition in from

$$||\Delta p|| < \epsilon \tag{2.9}$$

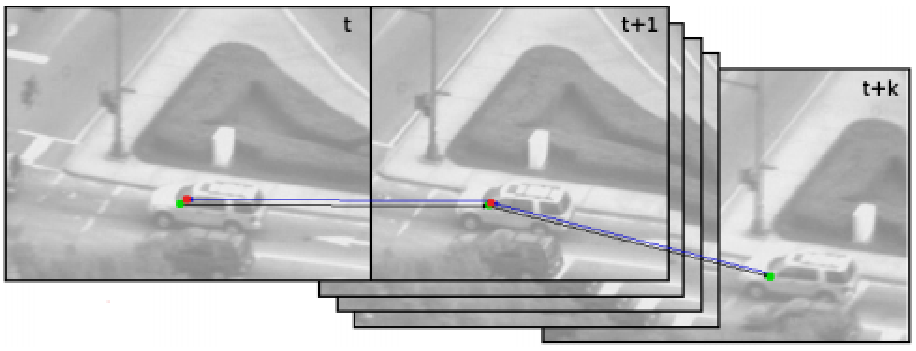


FIGURE 2.2: Forward-Backward Tracking.

2.2 Median Flow Tracker

The tracking solution is based on a previously released paper by Kalal et al.[2], and is also extended with a failure detector. The tracking is performed on an input bounding box, where the underlying patch is tracked from one camera frame and to the next. This is done by computing the optical flow(pixel displacement from one frame to the next) with the Lucas-Kanade(LK) algorithm on the pixels belonging to the patch(a 10 x 10) uniform grid of pixels is extracted from the patch). To ensure robustness, the LK algorithm is performed in two directions (first frame ! second frame and second frame ! first frame), and a reliability measure is computed for each displacement. The median displacement of 50% of the most reliable displacements is considered as the object's motion from the first frame to the second frame. The Median Flow Tracker algorithm is described in the following subsection.

```
Input: Current Image $I_i$
Input: Previous Image $I_{i-1}$
Input: Bounding Box $BB_i$
Output: New Bounding Box $BB_{i+1}$

for $k \leftarrow 1$ to All image sequence do
    $p_1....p_n \leftarrow selectPoints(BB_i)$
    for $p_i \leftarrow 1$ to n points do
        $p_1 \leftarrow LK_{FORWOARD}(p_i)$
        $p_2 \leftarrow LK_{BACKWOARD}(p_1)$
        $e \leftarrow |p_i - p_1|$
        $d \leftarrow NCC(W(p_i), W(p_1))$
    end
    $med_{NCC} \leftarrow median(d_1....d_n)$
    $med_{FB} \leftarrow median(e_1....e_n)$
    for $i \leftarrow 1$ to n points do
        if $d_i <= med_{NCC}$ and $e_i <= med_{FB}$ then
            $target \leftarrow p_i$
        end
    end
    $predictBB(traget, BB_{i+1})$
end
```

Algorithm 1: Median Flow Tracking Algorithm

For median flow tracker Kalal et al. [2]. devised two different method for filtering out the unwanted key-points. The first method is Euclidean distance measurement. The Euclidean distance calculate the distance between two given points, but instead of comparing two point we compare a grid of points in the algorithm. This process gives better approximation for tracking. Euclidean distance for two given points $a = (a_1, a_2)$ and $b = (b_1, b_2)$ is computed as $d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$.

$$Dist - ratio_{i,j} = \frac{d(LK_{t_i}, LK_{t_j})}{d(LK_{(t-1)_i}, LK_{(t-1)_j}} \tag{2.10}$$

Second method for filtering out the unwanted key-points is normalized cross-correlation (NCC). Normalized cross-correlation between object templates, obtained from first image , and object within bounding box is defined as

$$\frac{\sum_k \sum_l (T(k,l) - T)I(x+k, y+l) - I(x,y))}{\sqrt{\sum_k \sum_l (T(k,l) - T)^2} \sqrt{\sum_k \sum_l (I(x+k, y+l) - I(x,y))^2}} \tag{2.11}$$

Template from the image is generated by this processes. After filtering out the key-points, which do not satisfy the normalized cross-correlation and Euclidian distance condition the remaining key-points are tracked key-points. Using these remaining key-points the new bounding box for next iteration is computed.

2.3 Point Detectors

One of the part of object detection and tracking is to calculate a set of robust features which describes the object and improves the tracking result. Local invariant features is a popular method in object detection task because of its robustness. Invariant Local features allows efficient matching of sub structure in images. The advantage of using local invariant feature is that they encode key information about image that are invariant to number of image transformation such as, scale, rotation, translation and affine transformation. Also some pixels are easier to follow than others. A lots of invariant feature selection method have been proposed in the last decades. All of these method have their advantages and disadvantages in respective application. One of the part of our thesis work is to implement the following paper (TLD). Zdenek Kalal et al. used the 2-bit pattern features (similar to integral image) in their project. We believe that different feature calculation method can bring improvement in Object tracking performance. As a part of our thesis work we would like to test all this methods mentioned and test which one is good for tracking. Through this test what we want to achieve whether the

Forward-Backward error depends on feature calculation method. The following are short description of some well known method for key-points detection used in our project:

2.3.1 Harris Corner Detection

Harris Corner is basically finds the difference in intensity for a displacement of (x, y)in all directions. This is expressed as below:

$$E(X) = \sum_{x_i \in W} [I(x_i) - I(x_i + \Delta x)]^2 \tag{2.12}$$

We have to maximize this function $E(x, y)$ for corner detection. That means, we have to maximize the second term. Applying Taylor Expansion to above equation and using some mathematical steps, we get the final equation as:

$$E(X) = (\Delta X)^T A(X) \Delta X \tag{2.13}$$

where the auto-correlation matrix A is given by

$$A = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix} \tag{2.14}$$

A captures intensity pattern in W. Response $R(X)$ of Harris corner detector is given by

Two ways to define corners: (1) Large response. The locations X with $R(X)$ greater than certain threshold. (2) Local maximum . The locations X where $R(X)$ are greater than those of their neighbors, i.e., apply non-maximum suppression.

2.3.2 Good Feature Track

Also known as Shi and Tomasi Good Features considered weighted auto-correlation

$$E(X) = \sum_{x_i \in w} w(x_i) [I(x_i) - I(x_i + \Delta x)]^2 \tag{2.15}$$

where the weighted auto-correlation matrix A is given by

$$A = \begin{bmatrix} \sum_W wI_x^2 & \sum_W wI_xI_y \\ \sum_W wI_xI_y & \sum_W wI_y^2 \end{bmatrix} \tag{2.16}$$

This means there exist scalar values e_1, e_2 and vectors v_1, v_2. Where v_i are the orthonormal eigenvectors and e_i are the eigenvalues.

- If both e_i are small, then feature does not vary much in any direction. Uniform region (bad feature).
- If the larger eigenvalue $e_i >> e_2$, then the feature varies mainly in the direction of v_1. Edge (bad feature)
- If both eigenvalues are large, then the feature varies significantly in both directions. Corner or corner-like (good feature)
- In practice, image I has a maximum value (e.g., 255).So, e_1, e_2 also have an upper bound. So, only have to check that $min(e_1, e_2)$ is large enough.

2.3.3 SIFT

Scale Invariant Feature Transform (SIFT) is an image descriptor for image-based matching developed by David Lowe. This descriptor as well as related image descriptors are used for a large number of purposes in computer vision related to point matching between different views of a 3-D scene and view-based object recognition. The SIFT descriptor is invariant to translations, rotations and scaling transformations in the image domain and robust to moderate perspective transformations and illumination variations. In its original formulation, the SIFT descriptor comprised a method for detecting interest points from a grey-level image at which statistics of local gradient directions of image intensities were accumulated to give a summarizing description of the local image structures in a local neighbourhood around each interest point, with the intention that this descriptor should be used for matching corresponding interest points between different images. Later, the SIFT descriptor has also been applied at dense grids (dense SIFT) which have been shown to lead to better performance for tasks such as object categorization, texture classification, image alignment and biometrics .

2.3.4 SURF

In SIFT, Lowe approximated Laplacian of Gaussian with Difference of Gaussian for finding scale-space. SURF goes a little further and approximates LoG with Box Filter. One big advantage of this approximation is that, convolution with box filter can be easily calculated with the help of integral images. And it can be done in parallel for different scales. Also the SURF rely on determinant of Hessian matrix for both scale and location. For orientation assignment, SURF uses wavelet responses in horizontal and vertical direction for a neighbourhood of size 6s. Adequate guassian weights are also applied to it. Then they are plotted in a space as given in below image. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window of angle 60 degrees. Interesting thing is that, wavelet response can be found out using integral images very easily at any scale. For many applications, rotation invariance is not required, so no need of finding this orientation, which speeds up the process. SURF provides such a functionality called Upright-SURF or U-SURF.

2.3.5 MSER

Maximally Stable Extremal Regions (MSER) is a feature detector; Like the SIFT detector, the MSER algorithm extracts from an image I a number of co-variant regions, called MSERs. An MSER is a stable connected component of some level sets of the image I. Optionally, elliptical frames are attached to the MSERs by fitting ellipses to the regions.

2.3.6 ORB

ORB is basically a fusion of FAST key-point detector and BRIEF descriptor with many modifications to enhance the performance. Firstly, it use FAST to find key-points, then uses the Harris corner measure to find top N points among them. It also use pyramid to produce multi-scale-features. But one problem is that, FAST does not compute the orientation. So what about rotation invariance? Authors came up with following modification.

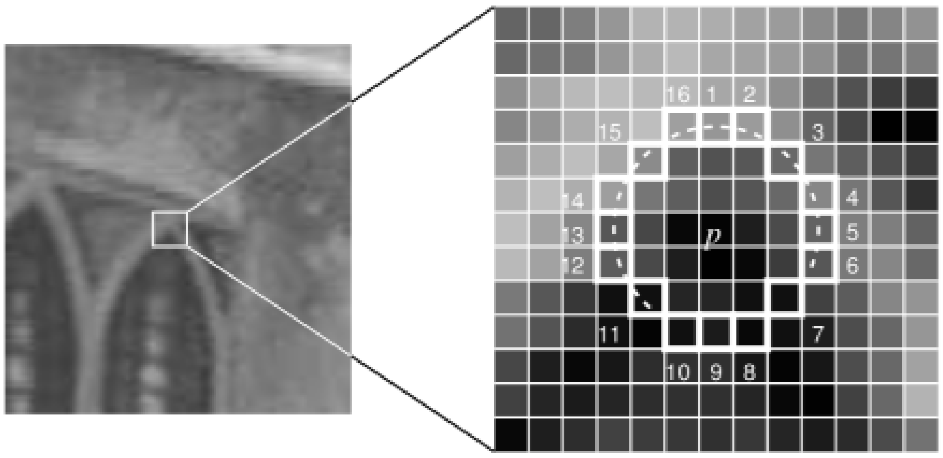


FIGURE 2.3: Process of FAST key-point method.

2.3.7 FAST

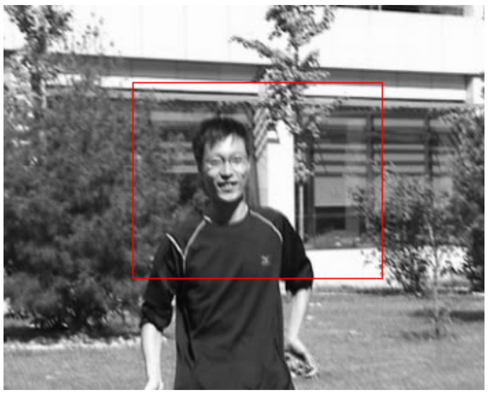
Select a pixel P in the image which is to be identified as an interest point or not. Let its intensity be I_p. Select appropriate threshold value t. Consider a circle of 16 pixels around the pixel under test. (See the image below)

Now the pixel P is a corner if there exists a set of n contiguous pixels in the circle (of 16 pixels) which are all brighter than $I_p + t$, or all darker than $I_p - t$. (Shown as white dash lines in the above image). was chosen to be 12. A high-speed test was proposed to exclude a large number of non-corners. This test examines only the four pixels at 1, 9, 5 and 13 (First 1 and 9 are tested if they are too brighter or darker. If so, then checks 5 and 13). If P is a corner, then at least three of these must all be brighter than $I_p + t$ or darker than $I_p - t$. If neither of these is the case, then cannot be a corner. The full segment test criterion can then be applied to the passed candidates by examining all pixels in the circle. This detector in itself exhibits high performance, but there are several weaknesses: It does not reject as many candidates for n ¡ 12. The choice of pixels is not optimal because its efficiency depends on ordering of the questions and distribution of corner appearances. Results of high-speed tests are thrown away. Multiple features are detected adjacent to one another. First 3 points are addressed with a machine learning approach. Last one is addressed using non-maximal suppression.

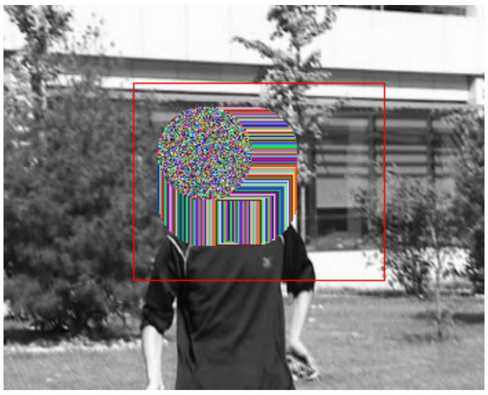
3 Detection

In this chapter we present the idea and the techniques that has been adopted for object detection. Object detection is an important part in our tracking system for multiple reason. Firstly, simple motion based tracking are not good for long term tracking and even for a short term tracking their performance aren't up to the mark. That's why we integrated this detection model in our tracking so that both performance can be improved radically. One of the popular method for detection is sliding window method, but sliding window method is time consuming and computationally heavy. For these reasons we employed a radius based image search method. For this method we assume that some parts of the object is always visible in input frame.

Our redial based image window search method is based on []. The figure 3.1 shows that output of the image window search. For this search method we only consider one fixed size width and height for every sub-windows. In the original paper [1] they have evaluated roughly 50,000 patches for every frame which effect the performance of the system heavily. In our method we only evaluate roughly 2,000 patch in every frame. In each frame these detected sub-windows are evaluated independently. Our detection system is composed of two different filtering systems. Each sub-windows have to go through these filters. Once a sub-window passes a filter it moves on to the next filter. Sub-window that fails the filter test are discarded. First method of our filtering system is variance filter. In this stage each sub-windows variance is compared with threshold variance. The second stage is simple naive bayes classification method with Haar-like feature and third stage consist of template matching method that is based on normalized cross correlation as similarity measure.



(a) Radius



(b) Search

FIGURE 3.1: Sub-window Searching.

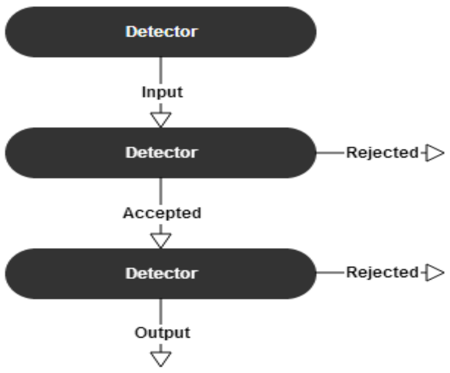


FIGURE 3.2: Illustration of detection process.

Chapter 2 organized as follows: Sec. 3.1 formulates the radial based approach for sub-window search. Section 3.2 describes the variance filter used in the project. Finally, Sections 3.3 describes the feature and classification algorithm used as the final filtering process.

3.1 Radial Based Approach

In the radial based sub-window search method the desired object is searched from near the position where the previous object is defined. Assume we are dealing with objects that have a relatively well-behaved appearance, and do not deform much. Then we can detect them with a very simple recipe. Each detected image patch might contain the desire object because of these carefully evaluation of every patch is much needed. The search method is formulated as follows: For an image I at time t, our tracker maintain the object location l_t^*. Let $l(x)$ denote the location of image patch x with only coordinate (x, y) and with fixed scale. For every new frame, we crop out a set image patches $X^s = x : ||l(x) - l_{t-1}^*|| < s$. Where s is some predefined radius of the tracker and compute $p(y|x)$ for all xEX^s. We update the new tracker location with classification method explain in section 4.

$$l_t^* = l\left(argmax_{xEX^s} p(y|x)\right) \tag{3.1}$$

Where l_t^* is new location with maximum probability. In this method the radius is chosen arbitrarily. In our case $s = 25$. The number of detected patches depend on s. When the radius is $s = 25$ we have around $2,000$ samples selected by detection method. As we increase the radius these number increases by factor of 2. The whole process is computed on a single threaded machine.

3.2 Variance Filter

The main context of measuring the similarity between patches is to understand the scene. There are lots of different approach available for patch similarity measurement. Comparison of patch variance is one such method. The variance gives you an idea how the pixel values are spread. In this section we present the mathematical definition of conventional way of calculating variance. We introduce a new method of calculating variance with integral image which requires only few memory lookups. The variance filter is the first filtering mechanism in our detection system. At First, we calculate the threshold variance from first frame of input image sequences. If any patches in subsequent frame has higher variance then the threshold variance then it moves on the next stage of the filtering system, otherwise we discard the patch. The variance σ^2 of a discrete set of random variables $X = \{x_1, ..., x_n\}$ is defined as

$$\sigma^2 = \sum_{i=1}^{n} p_i \left(x_i - \mu\right)^2 \quad with \quad \mu = \sum_{i=1}^{n} p_i x_i \tag{3.2}$$

where μ denoted as mean of the set. Let X be the set of pixel values in a rectangular block with all pixel values having the same probability $p_i = \frac{1}{n}$, then

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} \left(x_i - \mu\right)^2 \quad with \quad \mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.3}$$

which expands to

$$\sigma^2 = \frac{1}{n}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right) \tag{3.4}$$

The mathematical proof of equation (3.4) is given in appendix A. Both sums can be generated from two Integral Images over $I(x)$ and $I(x)^2$ respectively. For a two-dimensional image, both tables can be generated in a single loop with, on average, 1 product and 6 sums for calculation and 5 sums for data access per pixel. Using those, the variance of an arbitrary rectangular block of pixels can be generated in constant time with 3 products and 9 sums for calculation and 2 products and 6 sums for data access.

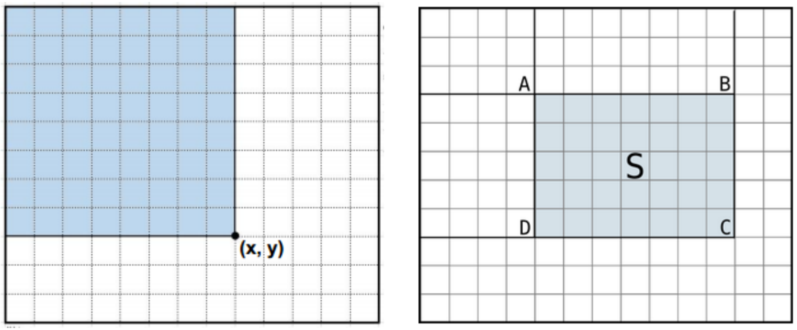
3.2.1 Integral Image

The integral Image (summed area table) was first properly introduced by Viola and Jones in 2001. They developed a very efficient object detection system using integral image. The Integral Image is one of the widely used technique in computer vision. The Integral Image is used as a quick and effective way of calculating the sum of values (pixel values) in a given image – or a rectangular subset of a grid (the given image). Each pixel in the integral image corresponds to the pixel sum between the origin and the pixel in the original image. Therefore the pixel sum of a rectangle can be calculated with just a few array references when using the integral image. The integral image itself can easily be computed in a single pass. The value of the integral image at point (x, y) is the sum of all the pixels above and to the left of (x, y) inclusive:

$$I'(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \tag{3.5}$$

where $I'(x, y)$ is the integral image and $I(x', y')$ is the original image. The Summed Area Table at (x, y) is simply calculated by

$$I'(x, y) = I(x, y) + I'(x-1, y) + I'(x, y-1) - I(x-1, y-1) \tag{3.6}$$



(a) Summed table up to (x, y) (b) Sum of patch ABCD

FIGURE 3.3: An example of integral image calculation.

The sum within S in figure 3.3(b) can be computed as

$$S = I'(A)I'(B) - I'(C) + I(D) \tag{3.7}$$

Using these integral image we calculate the mean $(\sum_{i=1}^{n} x_i)$ as follows:

$$\sum_{i=1}^{n} x_i = I'(x, y) - I'(x+w, y) - I'(x, y+h) + I'(x+h, y+w) \tag{3.8}$$

where w and h are the width and height of the bounding box B. In order to calculate the $\sum_{i=1}^{n} x_i^2$ we use the squared of integral image, denoted by $I''(x, y)$

$$\sum_{i=1}^{n} x_i^2 = \mu = I''(x, y) - I''(x+w, y) - I''(x, y+h) + I''(x+h, y+w) \tag{3.9}$$

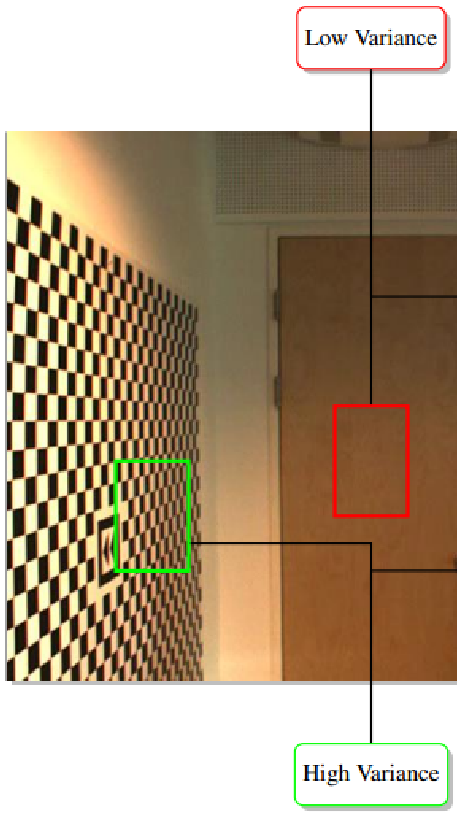


FIGURE 3.4: An example of variance filter.

By combining Eq , we get

$$\sigma^2 = \frac{1}{n} I''(B) - \left[\frac{1}{n} I'(B) \right]^2 \tag{3.10}$$

This process allow us very fast variance calculation, which is very important for robust tracking system.

3.3 Machine Learning

Machine learning, a branch of artificial intelligence, has several applications. It can concerns the construction and study of a system that can learn from data. The core of machine learning deals with representation and generalization. There is a wide variety of machine learning tasks and successful applications. Machine learning algorithm can be organized into a classification. Supervised Learning algorithms are trained on label examples, i.e. input where output is known. The supervised learning algorithm attempts to generalize a function or mapping from inputs to outputs. Unsupervised Learning algorithms operate on unlabeled dates, i.e. input, where the desired output is unknown. The main objective of unsupervised machine learning algorithm is to discover structure in the data. For example, a cluster analysis groups data into different cluster based on the characteristics of the data. Semi-supervised Learning algorithms combine both labelled and unlabeled examples to generate an appropriate classifier. Learning to Learn learns its own inductive bias based on previous experiences.

Feature selection has been an active and fruitful field of research area in object tracking, pattern reorganization, machine learning, statistics and data mining communities. The main objective of feature selection is to choose a subset of input variables by eliminating irrelevant features. Feature selection is also known as variable selection, attribute selection or variable subset selection in machine learning and statistics. Feature selection method is basically the process of selecting a subset of relevant features for use in a model construction. Features selection method are necessary to use when we assume that there are too many redundant features appear in the data. Redundant data means data without any information. By using efficient feature selection method it is possible to remove redundant data from the data set. Feature extraction create new features where feature selection method returns a subset of the features. Selecting the right features plays a critical role in tracking. In general, the most desirable property of a visual feature is its uniqueness so that the objects can be easily distinguished in the feature space. Feature selection is related to the object representation. For example, color is used as a feature for histogram based appearance representations, while for contour-based representations, object edges are usually used as features. Most features are chosen manually by users. In general, many tracking algorithms use a combination of these features.

3.3.1 Haar-like Features

Haar-like features are rectangular features, that can indicate specific characteristics in an image. The idea behind Haar-like features is to recognize objects or features based on the value of simple features, instead of pixel values directly. The Haar-like features have the advantage of very fast computation, because it depends only on the sum of pixels

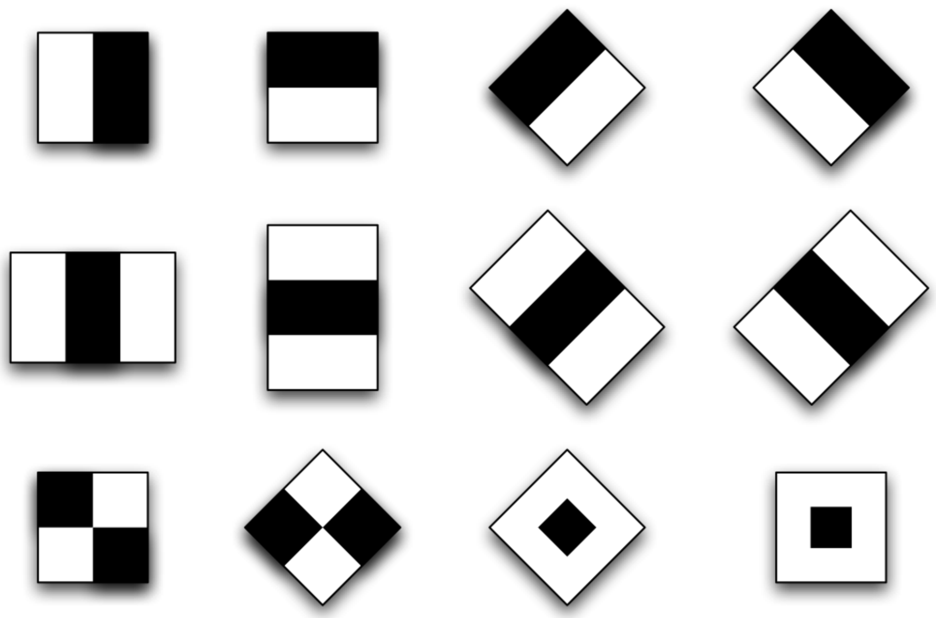


FIGURE 3.5: An example of Haar-like features.

within a rectangle instead of every pixel value. Using an integral image for calculating the sum, one rectangle can be computed with only four references, independent of the size of the feature. The simplest Haar-like feature is a two-rectangle feature. The value of this is calculated as the difference between the sum of the pixels within the two rectangles. This will indicate characteristics like edges or borders between light and dark regions. Three-rectangle features indicates for instance a dark line or a dark thin area lying between light regions, depending on the size of the middle rectangle. Four-rectangle features computes the difference between diagonal pairs of rectangles, and so on. The features shown below are Haar-like rectangular features.

3.3.2 Naive Bayes Classification

Naive Bayes is based on "Bayes Rule" and "naively" assumes independence given the label. Naive Bayes consider as simplistic assumption in real life but it works well on actual data-set. naive Bayes classifier is simple probabilistic classifier based on applying Bayes theorem which is derived from Bayesian Statistics with naive(strong) independence assumption. A more descriptive term for the underlying probability model would be "independent feature model". A naive Bayes classifier assume that the presence of a particular feature of a class is unrelated to the presence of any other feature. Naive Bayes

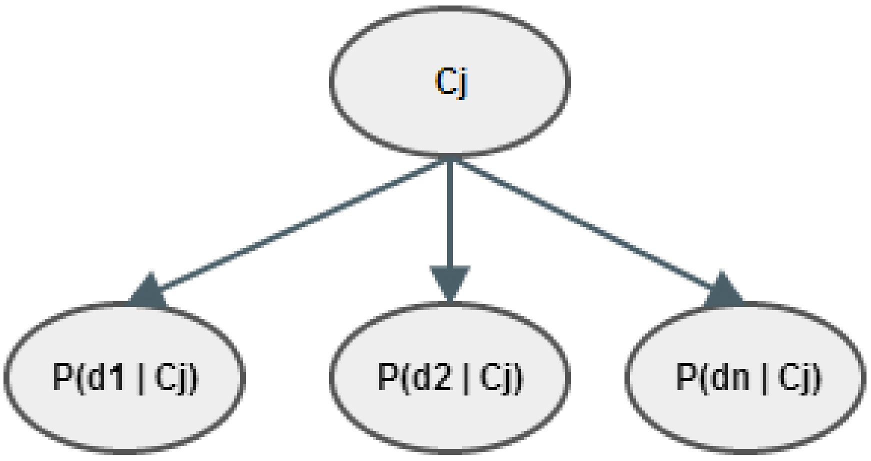


FIGURE 3.6: Diagram of naive Bayes classification.

classifiers can be trained efficiently in a supervised learning setting Bayes classifiers use Bayes theorem, which says

$$P(c_j|d) = \frac{P(d|c_j)}{P(d)} P(c_j) \tag{3.11}$$

Where,

- $P(c_j|d)$ = probability of instance d being c_j
- $P(d|c_j)$ = probability of generating instance d given classc_j
- $P(c_j)$ = probability of occurrence of class c_j
- $P(d)$ = probability of instance d

Naive Bayes classifiers assume attributes have independent distributions, and thereby estimate

$$P(d|c_j) = P(d_1|c_j) * P(d_2|c_j) * P(d_3|c_j) * .... * P(d_n|c_j) \tag{3.12}$$

The Naive Bayes classifiers is often represented as above

Naive Bayes is fast, space efficient, and sensitive to irrelevant features. In many real world situations bayes classifier works quiet well. To estimate the parameters (means and the variance of the variables) for classification, Naive Bayes only requires a small amount of training data which is considered as a huge advantage of Naive Bayes. The necessity to train the classifier online, Naive Bayes algorithm is a good choice. In case of object tracking its really challenging to track the object of interest exactly because of the appearance of the object in frame to frame. The appearance of the object is likely to be changed if the object move very fast in frame to frame. A learning rate is used to do the relative weighting the present and past examples. Model update is performed in every frame during tracking. Each descriptor is denoted by $x^i = (x_1^i, x_2^i, x_3^i, ..., x_n^i)$. A label $y^i = 1$ if the patch corresponds to the object of interest , $y^i = 0$ otherwise. The $x_j^i = 1$'s for $j = 1, 2, ..., n$ are supposed to be independent given y^i with

$$x_j^i | y^i = 1 - \ N(\mu_j^1, \sigma_j^1) \tag{3.13}$$

$$x_j^i | y^i = 0 - \ N(\mu_j^0, \sigma_j^0) \tag{3.14}$$

Also for $k \in 0, 1$, the mean and variance of the positive (k=1) and negative (k=0) training examples are extracted from each frame during tracking. If we give a weight λ to the training examples from the previous frame , and a weight $1 - \lambda$ to the training examples from the current frame, we can easily derive the mean and the variance of the resulting descriptor distribution. Then we find the following updated formulas:

$$\mu^k = \lambda \mu^k + (1 - \lambda)\mu^{k*} \tag{3.15}$$

$$\sigma^k = \sqrt{\lambda(\sigma^k)^2 + (1 - \lambda)(\sigma^{k*}) + \lambda(1 - \lambda)(\sigma^k - \sigma^{k*})^2} \tag{3.16}$$

Using the variance filter, Haar-like feature and naive Bayes classifier we formulate the detection algorithm. The detection algorithm is described in the following subsection:

```
**Input**: Current Image $I_i$
**Output**: Bounding Box $D_t$

$I' \leftarrow integral_{image}(I)$
$I'' \leftarrow integral_{image}(I^2)$

for $minrow$ to $maxrow$ do
    for $mincol$ to $maxcol$ do
        $box_x \leftarrow mincol$
        $box_y \leftarrow minrow$
        $box_{width} \leftarrow width$
        $box_{height} \leftarrow height$

        if $variance_{filter}(I', I'', box)$ then
            $B_i \leftarrow box$
        end
    end
end
$D_t \leftarrow Bayes_{classifier}(B)$
```

Algorithm 2: Detection Algorithm

4 Fusion

Fusion is the last stage of our tracking system. In this chapter we describe how we combine the output of Lucas-Kanade Tracker and the radial sub-window based detector into single final result.

The fusion stage of our system composed with simple template matching method. The result we got from the tracker with backward-forward method and result from detector, after processing the thousand of frame using variance filter and Naive Bayes classifier goes as input to our template matching system. Template matching method simply compare these two patches with each other and output the best patch.

This chapter is organised as follows. In Sec, 4.1 we shown the template matching method and Sec. 4.2 shows the Fusion method and finally we give the main method in Sec. 4.3.

4.1 Template matching

Template matching techniques are used in classifying objects as well as compare portion of images against one another. Templates are most often used to identify printed characters, numbers, and other small, simple objects. In template matching comparison is performed on a pixel by pixel level. The matching process moves the template image to all possible positions in a larger source image and computes a numerical index that indicates how well the template matches the image in that position. We use template matching in the last stage of our tracking system. Template matching is more restrictive than other classification method described in the detection process. To compare

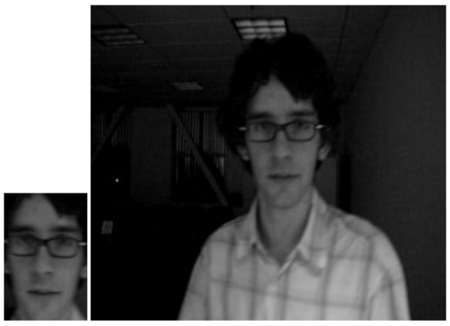


FIGURE 4.1: Find template in image.

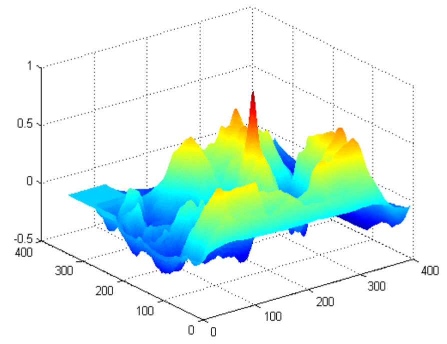


FIGURE 4.2: An example Normalized Cross-Correlation and Coordinates of Peak.

two patches $P1$ and $P2$, we employ the Normalized Correlation Coefficient (NCC) algorithm. NCC is used to determine the position of a given pattern in a two dimensional image I. For template matching process we do not resize the patches, because we only have two patches, one from tracking and another from detection part.

$$ncc(P1, P2) = \frac{1}{n-1} \sum_{x=1}^{n} \frac{((P_1(x) - \mu_1)(P_2(x) - \mu_2))}{\sigma_1 \sigma_2} \tag{4.1}$$

where $\mu_1, \mu_2, \sigma_1 and \sigma_2$ are the means and standard deviations of $P1 and P2$. NCC yields values between -1 and 1. The value is closer to 1 when two patches are similar. The distance between two patches is found using the following formula and the outcome is between 0 and 1.

$$d(P1, P2) = 1 - \frac{1}{2}(ncc(P1, P2) + 1) \tag{4.2}$$

Both the positive and negative class's template are considered. We consider positive class as P^+ and negative class as P^-. Distance of the positive class is defined by following equation:

$$d^+ = min_{P_i \in P^+} \; d(P_0, P_i) \tag{4.3}$$

Distance of negative class is defined by following equation:

$$d^- = min_{P_j \in P^-} \; d(P_0, P_j) \tag{4.4}$$

Finally using distance $d^+ and d^-$ we calculate the confidence value P^+. Because we have only two patches in our hand we do not need to use any clustering algorithm to decide which one is positive class. We just simply compare the two P^+ value and select the best patches.

4.2 Fusion

In this section we describe our simple fusion algorithm. Input for the fusion algorithm are the result of the recursive tracker R_t and the most confident detection D_t. Using these two $R_t and D_t$ patch template match decide which is our final B_t. The confidence value for detector is denoted as $P^+_{D_t}$ and the confidence value of tracker is denoted as $P^+_{R_t}$. If the detector yields the the higher $P^+_{D_t}$ value than tracker $P^+_{R_t}$ value, then the response from the detector is used as final result and also use to re-initialisation of the recursive tracker. If recursive tracker yields the higher $P^+_{R_t}$ value than detector $P^+_{D_t}$ value, then the response from the tracker is used as final result and also use to re-initialisation of the detector. The algorithm written below is the simple fusion used in our system,

4.3 Main Loop

Main loop is the last and main starting program in our tracking system. With this we finish describing our Robust Fusion Tracking system. In Algo. 3 the main loop of our implementation is given.

```
**Input**: Tracker $R_t$
**Input**: Detector $D_t$
**Output**: Final Bounding Box $B_t$

$P^+_{R_t} \leftarrow template_{matching}(R_t)$
$P^+_{D_t} \leftarrow template_{matching}(D_t)$

**if** $P^+_{R_t} > P^+_{D_t}$ **then**
    $B_t \leftarrow D_t$
**end**
**if** $P^+_{R_t} < P^+_{D_t}$ **then**
    $B_t \leftarrow R_t$
**end**
```

Algorithm 3: Fusion Algorithm

```
**Input**: Images $I_1, I_2, ....., I_n$
**Input**: Init Bounding Box $B_t$
$detector.init(I_1, B_t)$
**for** $k \leftarrow 2$ **to** $n$ **do**
    $R_t \leftarrow track(I_1, I_2, B_t)$
    $D_t \leftarrow detect(I_2, B_t)$
    $B_t \leftarrow fuse(R_t, D_t)$

    $imshow(B_t)$
**end**
```

Algorithm 4: Main Loop Algorithm

5 Experiment

In this chapter, we evaluated our fusion tracking system and compare with result published in "Robust Object Tracking with Online Multiple Instance Learning by Babenko, B. and Ming-Hsuan Yang and Belongie, S. We employ the standard center location error and precision for assessing performance. For this evaluation, a C++ implementation was created, where the calculation of the optical flow, the calculation of the normalised correlation coefficient, integral as well as all low-level image operations are implemented as function calls to the OpenCV 2.4.6 library. There was no multi threaded library used. In Sec. 5.2, we analysis 9 different key-point detection algorithm results are shown on Four video sequences. In Sec. 5.3 We experiment with eight different publicly available data-set and compare their result with us.

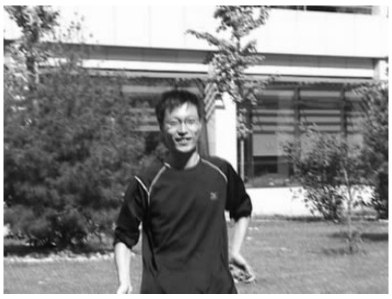
5.1 Evaluation Methodology

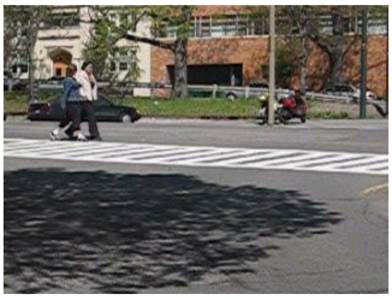
There is no universal method for evaluating fusion tracking system. But most common method used for evaluating tracking system is center location error and PASCAL overlap criteria. In our analysis we presented the screen shots of center location error analysis, we also include precision plots. These plots show the percentage of frames for which the estimated object location was within some threshold distance of the ground truth. To summarize these plots, we choose threshold 20 and report the precision at this point in the curve (e.g., this is the percent of frames for which the tracker was less than 20 pixels off from the ground truth); this threshold roughly corresponds to at least a 50 percent overlap between the tracker bounding box and the ground truth. For evaluating the Lucas-Kanade tracking system and the effect of the key-point detector we used the ground truth as our base. We only choose those point detector which can detect most key point in ground truth bounding box.

5.2 Experiment on Key-points

In order to find out the best key-point descriptor for Lucas-Kanade tracking algorithm, we device three different experiments. We evaluate 9 different state-of-the-art key-points detector algorithm using 4 challenging image sequences which publicly available. We used the most challenging sequences available in online and widely used by many other tracking algorithms for evaluation. Figure 5.1 shows a single frame of all test data-set.

(a) David


(b) Jumping


(c) Pedestrian

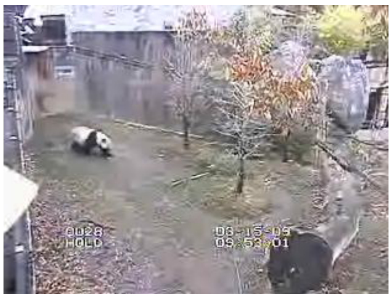
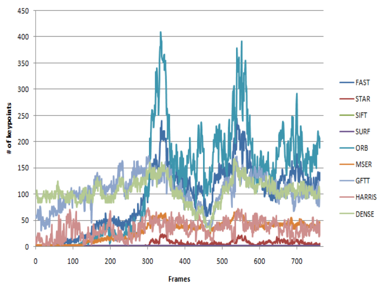

(d) Panda

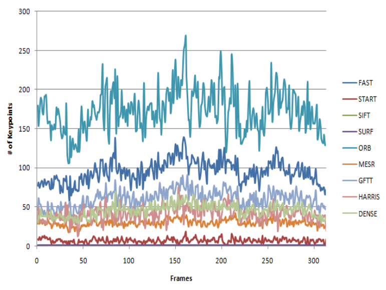
Figure 5.1: First frame of test image sequences

These image sequences are chosen because they are very challenging. For "David" image sequence the illumination and pose of the object both change gradually. For "Jumping" image sequence object moves very fast. For "Panda" and "Pedestrian" image sequence the object is too small and overlaps with each other.

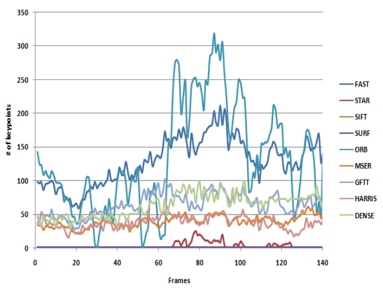
For our first experiment, we look for key-point's in ground truth (GT) bounding box in all image sequences, because if we don't find enough key-point then we won't be able



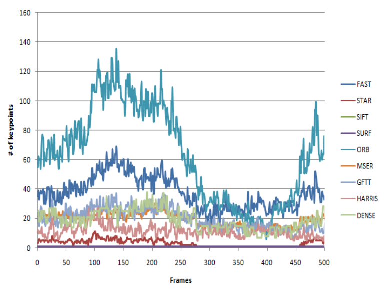
(a) David



(b) Jumping



(c) Pedestrian



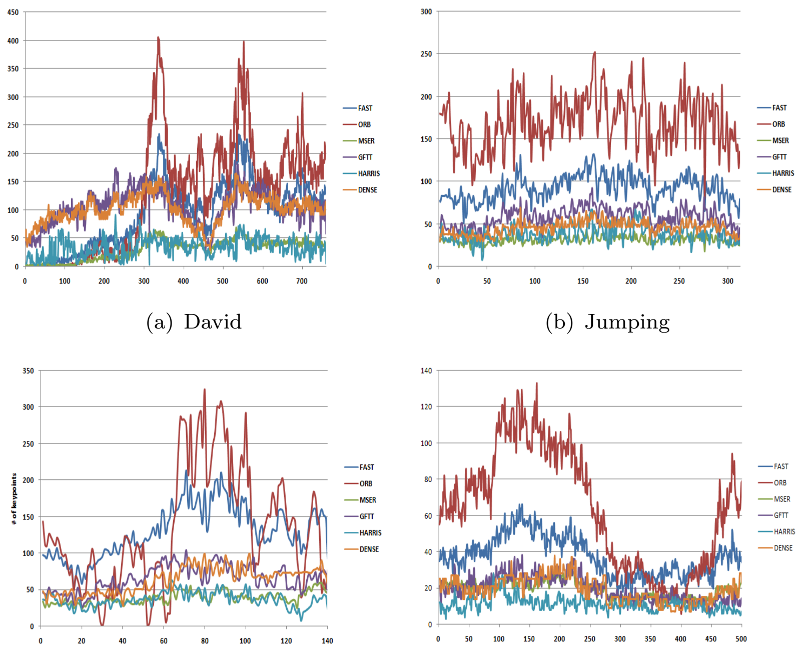
(d) Panda

Figure 5.2: Key-points found on test image sequences

to track the object. So we test which key-point detector finds the most key-point's in GT. we discard those descriptor which finds less than 20 key-point's in GT. Figure 5.2 shows the result of keypoint's find. We can see that the most key-point's are found by the ORB, FAST, GFTT, and Dense key-point descriptor. Even though SIFT and SURF our very robust and popular key-point descriptor, they were unable to find enough key-point's in the test image sequence. We discard the SIFT, SURF, and STAR key-point descriptor for our second experiment.

Foe our second experiment we only used the FAST, ORB, MSER, GFTT, HARRIS, and Dense key-point descriptor, because with these key-point descriptor we were able to find most key-point's. In the second experiment we look for those key-point's that satisfy the forward-backward condition. Forward-Backward condition state that if a key-point is found in image I_{i+1}, it should also be found in image I_{i-1} (Figure 2.2). If we can trace the key-point's back and fourth then those key-point's are good for tracking. From our experiment we found that using the forward-backward method we were able to find most of the key-point's. In some cases we found more key-point in forward move then the backward and vise versa.

Figure 5.3 shows the result of the forward-backward error method. We see that figure



(a) David (b) Jumping

(c) Pedestrian (d) Panda

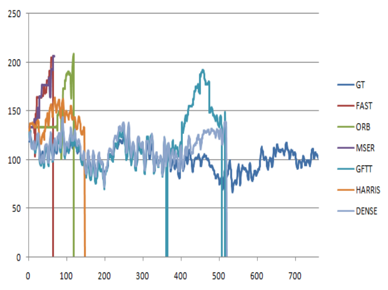
Figure 5.3: Forward-Backward error on test image sequences

5.2 and figure 5.3 is almost similar. We also see that for "David" image sequence most well known method failed for first 300 hundred frame. Once the illumination change to brighter background these method were able to pick most key-point's. For "Jumping" sequence the result were very consistent for FAST,ORB, and Dense method,. But for "Pedestrian" and "Panda" image sequence the all the method were fluctuating up and down. This because the object was overlapped and small.

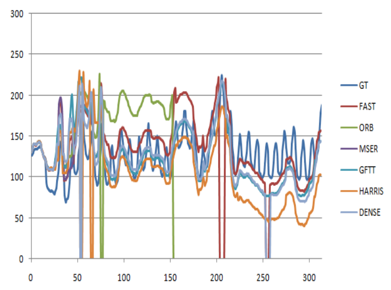
For our final experiment we calculate the center location error (CLE). The CLE is calculated by comparing the center location of predicted bounding box and ground truth bounding box. The best method should have the lowest center location error. Here we analysis all of our test image sequence performance one by one. First "David" image sequence.

In figure 5.4(a) we see that FAST, ORB, MSER, and HARRIS key-point detector method were unable to find key-point after 150 frame. This is because the sudden illumination and pose change in the image. Which results tracking failure. But GFTT and Dense key-point were able to track the object around 500 frame.

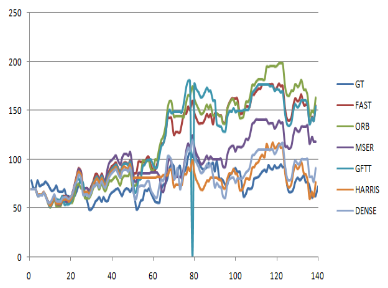
Our next image sequence is "Jumping". In this sequence the object is moving up and down continuingly. Which pose a grate difficulty for tracker.



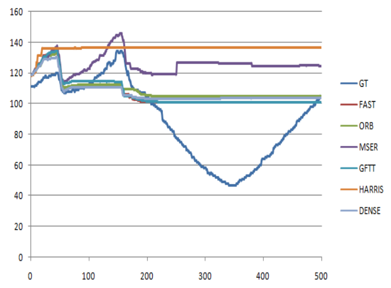
(a) David



(b) Jumping



(c) Pedestrian



(d) Panda

FIGURE 5.4: Center location error on test image sequences

Figure 5.4(b) shows the result of “Jumping” image sequence. From the figure we see that because of the fast change in object location all the key-point descriptor were unable to track the object perfectly.

The nest image sequence is “Pedestrian”. In this sequence the object is overlapped with another similar object. Which is difficult for tracker of distinguish.

From figure 5.4(c) we see that except for HARRIS and Dense key-point descriptor all other descriptor deviated from original ground truth location by a large margin.

Our final image sequence is “Panda”. Here the object was very small and most key-point descriptor were unable to find good key-point’s for tacking.

In this image sequence the result were much worse than the last three test data. From figure 5.4(d) we see that all of key-point descriptor stop tracking the object after 100 frame. The flat line in the figure means the object is not moving. Even though originally the object was moving whole time.

After completing the experiment with 9 different key-point descriptor with Lucas-Kanade tracker the result was not so convincing. Most of the well known method failed the experiment. Only the classic method like Good-Feature-To-Track and Dense key-point detector were able to performed much better than the other method. For long-term tracking only Lucas-Kanade tracking is not good enough. In order to increase the performance of tracking we need to integrate other method like object detection and template matching with it.

5.3 Experiment on MIL Data-set

We perform our experiments on eight publicly available video sequences. For all sequences, we labeled the ground truth center of the object for every five frames, and interpolated the location in the other frames. All video frames were converted to gray scale prior to processing. The quantitative results are summarized in Tables 1, and precision plots are shown in Figs.(5.6). Below is a more detailed discussion of the video sequences.

5.3.1 Sylvester and David indoor

These sylvester and david indoor video sequences have been used in several recent tracking papers, and they present challenging lighting, scale, and pose changes. Our fusion algorithm achieves the best performance.

5.3.2 Occluded Face and Cola

In the "Occluded Face 1" sequence, which comes from the authors of [], FragTrack performs the best because it is specifically designed to handle occlusions via a part-based model. Our method achieve third best result.However, on our similar, but more challenging clip, "Occluded Face 2", FragTrack performs poorly because it cannot handle appearance changes well. MIL Tracking performs best in this image sequence by using an adaptive appearance model. Our method achieve the second best result in this sequence with a very close result.

The cola can sequence contains a specular object, which adds some difficulty. These sequences exhibit many challenges and contain frequent occlusions and fast motion (which causes motion blur). For this sequence MIL track and fusion tracker have similar result.

5.3.3 Tiger

The two ”Tiger” sequences show the toy tiger in many different poses, and include out of plane rotations. Our fusion algorithm outperforms the others, often by a large margin. This clip illustrates a problem that arises when the tracker relies too heavily on the first frame.

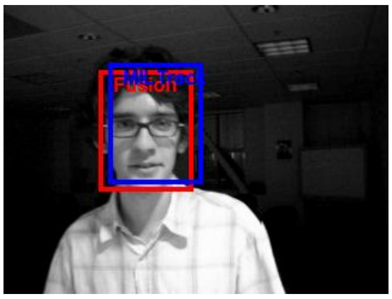
5.3.4 Dollar

The appearance of the coupon book is changed after about 50 frames by folding one of its pages; then an “imposter” coupon book is introduced to distract the trackers. MILTrack successfully tracks the correct coupon book, while FragTrack and the SemiBoost tracker are confused by the impostor object. Our fusion tracker achieve second best result in dollar sequence.

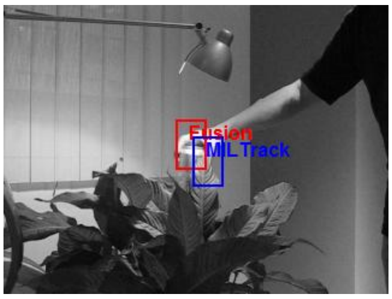
Table 5.1: Center location error

Video Clip	OAB	SemiBoost	Fragment	MILTrack	Fusion
Sylvester	25	16	🟩 11	🟩 11	🟥 9
David	49	39	46	🟩 23	🟥 15
Cola Can	25	🟥 13	63	🟩 20	🟩 20
Face 1	43	🟩 7	🟥 6	27	18
Face 2	21	23	45	🟥 20	🟩 21
Tiger 1	35	42	39	🟩 16	🟥 10
Tiger 2	33	61	37	🟩 18	🟥 15
Dollar	25	67	56	🟥 15	🟩 20

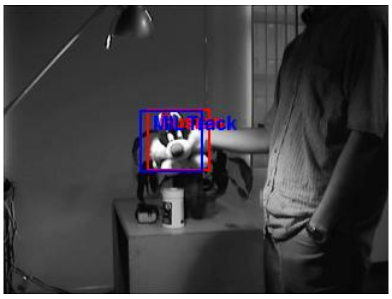
In the table 1 the red cell indicates the best result and green cell indicates the second best result. From the we that our tracking system achieve best result in four data-set and second best in two data-set. Figure 5.5 shows the output bounding box of MIL tracker and Fusion tracker. Figure 5.6 shows the precision plots. The higher the precision score, better the result.



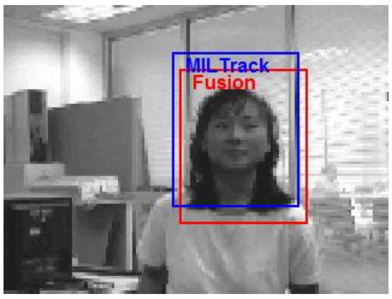
(a) David



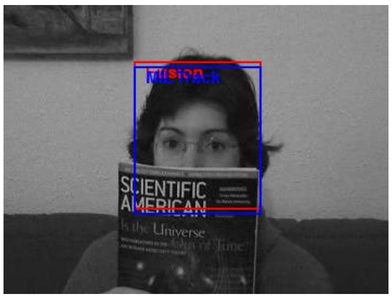
(b) Coke



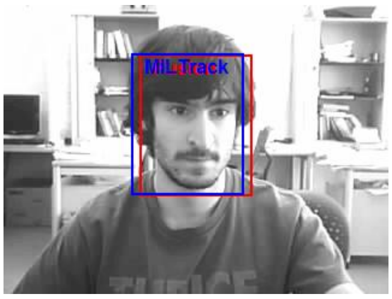
(c) Sylvester



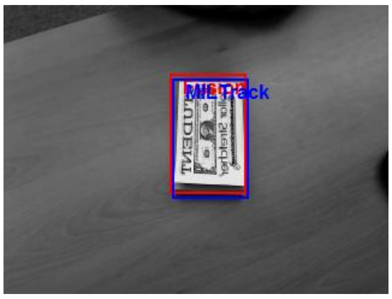
(d) Girl



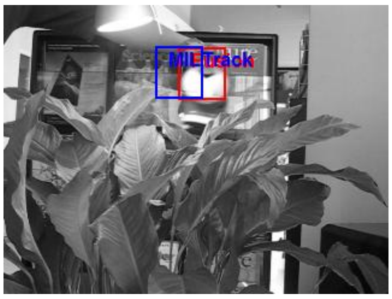
(e) Face1



(f) Face2

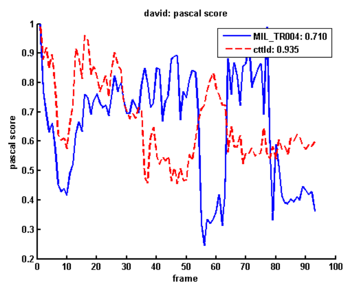


(g) Dollar

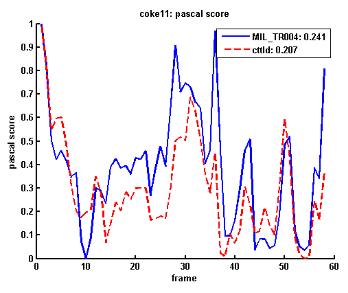


(h) Tiger

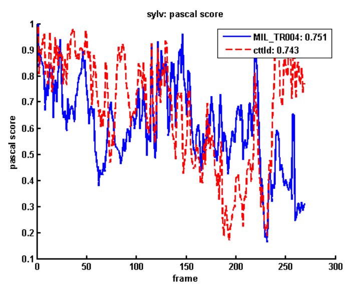
FIGURE 5.5: Ouput of MIL Track and Fusion tracking



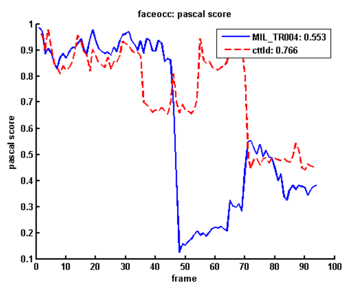
(a) David



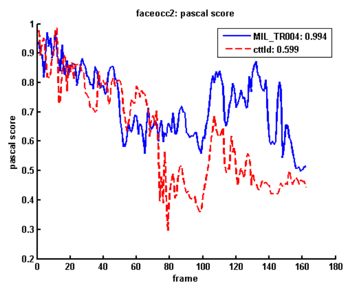
(b) Coke



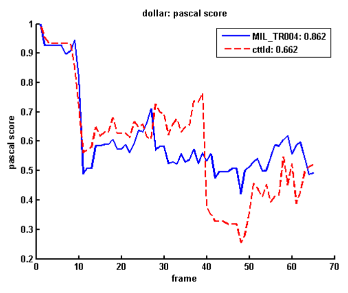
(c) Sylvester



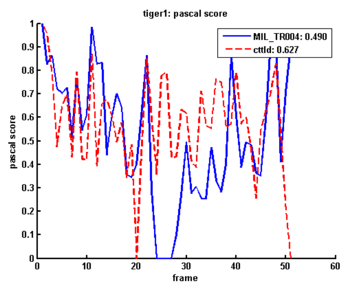
(d) Face1



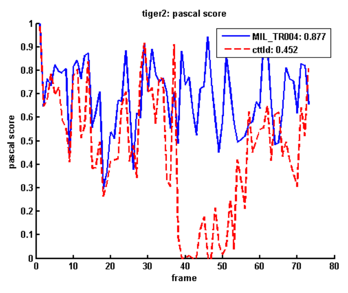
(e) Face2



(f) Dollar



(g) Tiger1



(h) Tiger2

Figure 5.6: Precision plot of test image sequences

6 Conclusion

6.1 Conclusion

In this paper we learn the problem of object tracking of an unknown object in a video stream. The task was really challenging because the object which is our object of interest changes its position frequently by moving in and out of the camera view. We follow the TLD [2] And MIL [1] Track method for robust object tracking which decomposes the task in three components: tracking, learning and detection. We design a new framework based on TLD and MIL Track which performs better for tracking and detect object efficiently. The system is capable of continuous tracking while building a reliable detector without using any prior information on the target. Initially we reproduce the results using Lucas-Kanade tracker based on different key point descriptor. The result was not convincing compared to Kalal et al. In order to make the tracker more efficient we implement the detector part in different stages. The performance based on Tracking-Learning-Detection is further improved by the automatic detection of tracking failures. Tracking failures are detected by the forward-backward error detection method and then follow the median flow method for object tracking. Our approaches basically depends on the outcomes of the tracker. We overcome one problem that is encountered during the experiments is that the object detector is unable to detect the object of interest because of multiple same objects. Specially in the TLD data-set some video have same objects which appear in the same frame. This problem occurred during the template matching on images with reduced size. Some shortcomings related to bounding box appears during tracking and detection. We assign a class label on bounding box level for which the appearance of background to be considered part of the object of interest.

But the object of interest is not recognized when the background is not similar compared to the initial background. As a result, when the tracker and the detector work together, the bounding boxes for both tracking and detection don't appear at the same object. Different segmentation techniques are used to overcome this problem.

The proposed system was extensively tested on a variety of objects and the results show clear improvement compared to standard trackers. Results of our approach with respect to the closest approaches was clearly demonstrated.

A Appendix

A.1 Variance

Mathematical derivation of variance

$$
\begin{aligned}
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\left(x_i^2 - 2x_i\mu + \mu^2\right)\\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} x_i^2 - \tfrac{1}{n}\sum_{i=1}^{n} 2x_i\mu + \mu^2\\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} x_i^2 - \tfrac{1}{n}\sum_{i=1}^{n} 2x_i\mu + \tfrac{1}{n^2}(\sum_{i=1}^{n} x_i)^2\\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} x_i^2 - \tfrac{2\mu}{n}\sum_{i=1}^{n} x_i + \tfrac{1}{n^2}(\sum_{i=1}^{n} x_i)^2\\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} x_i^2 - \tfrac{2}{n^2}(\sum_{i=1}^{n} x_i)^2 + \tfrac{1}{n^2}(\sum_{i=1}^{n} x_i)^2\\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} x_i^2 - \tfrac{1}{n^2}(\sum_{i=1}^{n} x_i)^2\\
&= \tfrac{1}{n}\textstyle\sum_{i=1}^{n} x_i^2 - \left(\tfrac{1}{n}\sum_{i=1}^{n} x_i\right)^2\\
&= \tfrac{1}{n}\left(\textstyle\sum_{i=1}^{n} x_i^2 - \tfrac{1}{n}(\sum_{i=1}^{n} x_i)^2\right)
\end{aligned}
$$

Bibliography

[1] B. Babenko, Ming-Hsuan Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1619–1632, 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.226.

[2] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, July 2012. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.239. URL `http://dx.doi.org/10.1109/TPAMI.2011.239`.

[3] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 798–805, 2006. doi: 10.1109/CVPR.2006.256.

[4] Hanxi Li, Chunhua Shen, and Qinfeng Shi. Real-time visual tracking using compressive sensing. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1305–1312, 2011. doi: 10.1109/CVPR.2011.5995483.

[5] S. Avidan. Ensemble tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):261–271, 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.35.

[6] Fan Yang, Huchuan Lu, and Yen wei Chen. Bag of features tracking. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 153–156, 2010. doi: 10.1109/ICPR.2010.46.

[7] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 234–247, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88681-5. doi: 10.1007/978-3-540-88682-2_19. URL http://dx.doi.org/10.1007/978-3-540-88682-2_19.

[8] Martin Godec, Helmut Grabner, Christian Leistner, and Horst Bischof. Speeding up semi-supervised on-line boosting for tracking.

[9] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 723–730, 2010. doi: 10.1109/CVPR.2010.5540145.

[10] Qian Yu, Thang Ba Dinh, and Gérard Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 678–691, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88685-3. doi: 10.1007/978-3-540-88688-4_50. URL http://dx.doi.org/10.1007/978-3-540-88688-4_50.

[11] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994. doi: 10.1109/CVPR.1994.323794.

[12] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm, 2000. URL http://robots.stanford.edu/cs223b04/algo_tracking.pdf.

[13] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vision*, 56(3):221–255, February 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000011205.11775.fd. URL http://dx.doi.org/10.1023/B:VISI.0000011205.11775.fd.

[14] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time compressive tracking. In *Proceedings of the 12th European conference on Computer Vision - Volume Part III*, ECCV'12, pages 864–877, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-33711-6. doi: 10.1007/978-3-642-33712-3_62. URL http://dx.doi.org/10.1007/978-3-642-33712-3_62.

[15] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proceedings of the British Machine Vision Conference*, pages 6.1–6.10. BMVA Press, 2006. ISBN 1-901725-32-4. doi:10.5244/C.20.6.

[16] David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *Int. J. Comput. Vision*, 77(1-3):125–141, May 2008. ISSN 0920-5691. doi: 10.1007/s11263-007-0075-7. URL http://dx.doi.org/10.1007/s11263-007-0075-7.

[17] Ruei-Sung Lin, David A. Ross, Jongwoo Lim, and Ming-Hsuan Yang. Adaptive discriminative generative model and its applications. In *NIPS*, 2004. URL http://dblp.uni-trier.de/db/conf/nips/nips2004.html#LinRLY04.

[18] Navid Nourani-Vatani, P Borges, and J Roberts. A study of feature extraction algorithms for optical flow tracking.

[19] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, June 2003. ISSN 0022-0000. doi: 10.1016/S0022-0000(03)00025-4. URL http://dx.doi.org/10.1016/S0022-0000(03)00025-4.

[20] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[21] S. Vijayanarasimhan and K. Grauman. Efficient region search for object detection. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1401–1408, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-0394-2. doi: 10.1109/CVPR.2011.5995545. URL http://dx.doi.org/10.1109/CVPR.2011.5995545.

[22] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2756–2759, 2010. doi: 10.1109/ICPR.2010.675.

[23] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007. doi: 10.1109/ICCV.2007.4409066.

[24] Yu Meng and Dr. Bernard Tiddeman(supervisor. Implementing the scale invariant feature transform(sift) method.

[25] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280, July 2008. ISSN 1572-2740. doi: 10.1561/0600000017. URL http://dx.doi.org/10.1561/0600000017.

[26] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2195–2202, 2006. doi: 10.1109/CVPR.2006.219.

[27] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto. Making good features track better. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '98, pages 178–, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8497-6. URL http://dl.acm.org/citation.cfm?id=794191.794739.

[28] Shengluan Huang and Jingxin Hong. Moving object tracking system based on camshift and kalman filter. In *Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on*, pages 1423–1426, 2011. doi: 10.1109/CECNET.2011.5769081.

[29] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), December 2006. ISSN 0360-0300. doi: 10.1145/1177352.1177355. URL http://doi.acm.org/10.1145/1177352.1177355.

[30] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1417–1424, 2009. doi: 10.1109/ICCVW.2009.5457446.

[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544.

[32] Z. Kalal, J. Matas, and K. Mikolajczyk. P-n learning: Bootstrapping binary classifiers by structural constraints. In *Computer Vision and Pattern Recognition*

(CVPR), 2010 IEEE Conference on, pages 49–56, 2010. doi: 10.1109/CVPR.2010.5540231.

[33] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 775–781 vol. 2, 2005. doi: 10.1109/CVPR.2005.288.

[34] S. Hinterstoisser, O. Kutter, N. Navab, P. Fua, and V. Lepetit. Real-time learning of accurate patch rectification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2945–2952, 2009. doi: 10.1109/CVPR.2009.5206794.

[35] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Yi Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.79.

[36] Charles Bibby and Ian Reid. Robust real-time visual tracking using pixel-wise posteriors. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 831–844, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88685-3. doi: 10.1007/978-3-540-88688-4_61. URL http://dx.doi.org/10.1007/978-3-540-88688-4_61.

[37] Vasileios Belagiannis, Falk Schubert, Nassir Navab, and Slobodan Ilic. Segmentation based particle filtering for real-time 2d object tracking. In *Proceedings of the 12th European conference on Computer Vision - Volume Part IV*, ECCV'12, pages 842–855, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-33764-2. doi: 10.1007/978-3-642-33765-9_60. URL http://dx.doi.org/10.1007/978-3-642-33765-9_60.

[38] N. D H Dowson and R. Bowden. Simultaneous modeling and tracking (smat) of feature sets. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 99–105 vol. 2, 2005. doi: 10.1109/CVPR.2005.324.

[39] Towards Recognizing Feature Points using Classification Trees - Infoscience. URL http://infoscience.epfl.ch/record/52666.

[40] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(3): 448–461, March 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2009.23. URL `http://dx.doi.org/10.1109/TPAMI.2009.23`.

[41] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007. doi: 10.1109/CVPR.2007.383123.

[42] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1296–1311, 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1233903.

[43] R.T. Collins, Yanxi Liu, and M. Leordeanu. Online selection of discriminative tracking features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1631–1643, 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.205.

[44] D.L. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52 (4):1289–1306, 2006. ISSN 0018-9448. doi: 10.1109/TIT.2006.871582.

[45] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001. doi: 10.1109/CVPR.2001.990517.

[46] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 841–848. MIT Press, 2001. URL `http://dblp.uni-trier.de/db/conf/nips/nips2001.html#NgJ01`.

[47] S. Hare, A. Saffari, and P. H S Torr. Struck: Structured output tracking with kernels. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 263–270, 2011. doi: 10.1109/ICCV.2011.6126251.