Integrated Algorithm for Robust Visual Tracking

A Report submitted

in partial fulfillment for the award of the Degree of

MASTER OF TECHNOLOGY

in

AVIONICS

by

MADAN KUMAR RAPURU

pursued in

DEPARTMENT OF AVIONICS

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

to

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

Thiruvananthapuram

April 2016

Integrated Algorithm for Robust Visual Tracking

A Report submitted

in partial fulfillment for the award of the Degree of

MASTER OF TECHNOLOGY

in

AVIONICS

by

MADAN KUMAR RAPURU

pursued in

DEPARTMENT OF AVIONICS

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

to

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

Thiruvananthapuram

April 2016

CERTIFICATE

This is to certify that the project report entitled **Integrated Algorithm for Robust Visual Tracking** submitted by **Madan Kumar Rapuru**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfilment for the award of the degree **Master of Technology** in **Avionics**, is a bona fide record of the project research work carried out by him under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Deepak Mishra
Supervisor-I
Department of Avionics

Dr. R. K. Sai Subrahmanyam Gorthi
Supervisor-II
Department of Earth and Space Sciences

Dr. N. Selvaganesan
Head of the Department, Avionics
IIST, Trivandrum

Place: Thiruvananthapuram
April 2016

DECLARATION

I declare that this report titled **Integrated Algorithm for Robust Visual Tracking** submitted in partial fulfilment of the Degree of **Master of Technology** in **Avionics** is a record of original work carried out by me under the supervision of **Dr. Deepak Mishra** and **Dr. R. K. Sai Subrahmanyam Gorthi** , and has not formed the basis for the award of any degree, diploma, associateship, fellowship or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Place: Thiruvananthapuram
April 2016

Madan Kumar Rapuru
SC14M022

ACKNOWLEDGMENTS

I would like to take this opportunity to express my immense gratitude to my supervisors Dr. Deepak Mishra and Dr. R. K. Sai Subrahmanyam Gorthi for giving me this invaluable opportunity to work under their expert guidance. I would like to primarily thank them for taking time from their busy schedule and spending long hours for brain storming discussions which has aided me to successfully complete the project.

Their unflinching support, suggestions and directions have helped me sail smoothly throughout the project duration. Most importantly, they have been a constant source of inspiration in all possible ways for the successful completion of my project. I am highly indebted to them for all the care and concern they have shown towards me.

Madan Kumar Rapuru

SC14M022

ABSTRACT

This thesis aims at studying and understanding the state of the art trackers and various challenges involved in visual object tracking. Tracking as a field, has seen massive research in previous decades and now algorithms are being developed which have shown success even in challenging scenarios. Still there is a lot of scope for doing research in tracking since there is no single algorithm or method which can handle all the issues in it. This work started with Tracking Learning Detection (TLD), an unique tracker which have tracking resumption ability and outperforms other trackers in many challenging situations. It is observed that it can not handle fast motions, Out of plane rotation and Non rigid body tracking, primarily due to the tracker component in it. To replace the tracker component in TLD we have carried out studies on Struck tracker (STR), Compressive Tracker (CT) and Kernelized correlation Filter tracker (KCF).

The main idea of the thesis is based on the augmentation of tracker component in TLD with the state of the art trackers as mentioned above. After experimenting with different combinations it is found that KCF tracker would be a better solution because of its high speed (150fps) and precision in handling complementary issues with TLD. The proposed algorithm efficiently combines the two tracking algorithms, which takes advantages of both and outperforms them on their short-ends by virtue of other.

Extensive evaluation of the proposed tracker has been carried out on the datasets ALOV300++ and Visual Tracker Benchmark. The precision plots, Success plots for Spatial Robustness Evaluation(SRE), Temporal Robustness Evaluation(TRE) and One Pass Evaluation(OPE) have been generated. Through experimentation, both qualitatively and quantitatively it is proved that the proposed method outperforms the state of the art tracker algorithms both in terms of robustness and success rate.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

HCI	Human Computer Interaction
KCF	Kernelized Correlation Filters
TLD	Tracking Learning Detection
HOG	Histogram of Oriented Gradients
NN	Nearest Neighbourhood
DFT	Discrete Fourier Transform
RVT	Robust Visual Tracker
STR	STRuctured output tracking with kernels
SURF	Speeded Up Robust Features
SRE	Spatial Robustness Evaluation
TRE	Temporal Robustness Evaluation
OPE	One Pass Evaluation
LBP	Local Binary Pattern
FPS	Frames Per Second
SVM	Support Vector Machine
S^c	Conservative Similarity

CHAPTER 1

INTRODUCTION

1.0.1 Tracking Challenges

Illumination Variations

The object changes its appearance under different illumination conditions. This makes tracking task difficult. By keeping the local background along with a model of the target this can be overcome. Fig 1.2 illustrates the scenario

Figure 1.1: Challenges in tracking: illumination changes.

Scale Variations

Due to zooming of camera or because of the motion of the object the object is subject to scale variations. While implementing a tracker scale estimation brings an additional degree of freedom to the tracking process and as such increases vulnerability of the tracker to failure. Fig 1.3 illustrates the scenario.

Figure 1.2: Challenges in tracking: Scale changes.

Occlusion

The object may disappear from the camera view or it may be occluded for an arbitrarily length of time. The same object may reappear at any time and at any location. Since occlusions cover and uncover the object it is challenging to track objects in those cases. Fig 1.4 illustrates the scenario.

Figure 1.3: Challenges in tracking: Occlusion.

Clutter and Confusion

The object may appear in cluttered environments or may be surrounded by other objects of the same visual class. Hence it is challenging to not get distracted by background clutter and be able correctly distinguish the object of interest from other objects of the same class in the scene. Fig 1.5 illustrates the scenario.

Figure 1.4: Challenges in tracking: Clutter.

Motion blur

Due to the motion of target or camera the target region is blurred. Hence the appearance model of the target is greatly effected. So it becomes a challenge for tracking.

Fig 1.6 illustrates the scenario

Figure 1.5: Challenges in tracking: Motion Blur.

Appearance changes

The object of interest may change its appearance and view point throughout the sequence. This can be due to in plane rotation, out of plane rotation or deformation of the object. These appearance changes complicates the tracking process as the only information given to the tracker is a single patch from the initial frame, which may not be relevant throughout the entire sequence.

Other tracking challenges include Fast motion, Low Resolution. A video is said to be having fast motion if the motion of the ground truth is larger than t_m pixels (t_r=20). Here t_m is the threshold. A video is considered to be of low resolution if the number of pixels inside the ground-truth bounding box is less than t_r (t_r =400).Here t_r is the threshold.

1.1 Objective of the work

Assume a video stream depicting various objects moving in and out of the camera field of view. Given a bounding box defining the object of interest in a single frame, a robust tracker should be able to automatically determine the object's bounding box or indicate that the object is not visible in every frame that follows. It should be able to handle all the tracking challenges like illumination variations, occlusions, scale changes, background clutters, deformations, in-plane and out-of- plane rotations etc. The video stream is to be processed at full frame-rate and the process should run indefinitely.

A number of algorithms related to tracking have been proposed in the past. Each algorithm has its merits and demerits. This work started with Tracking Learning Detection (TLD), an unique tracker which have tracking resumption ability and outperforms other trackers in many challenging situations. But this tracker can not handle all the challenges, So to improve its performance we explored other state of the art trackers like Compressive tracker [2], Structured output with kernels : STR tracker [4] and Kernelized correlation filters tracker [5]. A detailed analysis of the functioning has been presented. It also presents the cases where these algorithms fail and in which these are robust. The primary objective of this project is to come up with an efficient robust tracking algorithm by integrating the state-of-the-art algorithms KCF and TLD. It should have all the good virtues that KCF and TLD possess. It should be able to handle the challenges which KCF and TLD alone cannot handle. It should be more precise and accurate.

Another objective involves the extensive evaluation of the proposed algorithm over the datasets ALOV300++ and Visual Tracker Benchmark. Precision values and success rates averaged over all the videos present in the dataset have been generated. Generation of precision plots and success plots to measure the overall performance is of necessity to compare the robustness of the proposed algorithm with the other eminent trackers available. Spatial robustness evaluation, temporal robustness evaluation and one pass evaluation of the proposed tracker has also been done.

1.2 Organisation of thesis

The present chapter dealt with the literature behind tracking. It gave a brief description about the objective of the work. Chapter 2 provides a detailed insight into the working of TLD. It analyses the performance of its different components and the novelty in approach. Chapter 3 describes KCF algorithm in detail. The concepts behind KCF like circulant matrices, kernel etc., are explained very clearly. Chapter 4 have detailed

description and algorithm of Compressive Tracker. Chapter 5 compares critically both algorithms based on different challenges faced during tracking such as illumination change, occlusion, scale change etc. Such critical analysis has helped to come up with a robust algorithm for tracking. A clear description of the proposed algorithm has been presented in that chapter. chapter 6 deals with extensive evaluation of the proposed algorithm on ALOV300++ dataset and Trackers benchmark dataset. Precision plots and success plots for OPE, SRE, TRE have been generated. The results have been compared with the state of the art trackers KCF, TLD and STRuck. Chapter 7 concludes the report and provides future insight for further improvement of the proposed tracker.

CHAPTER 2

Tracking learning detection

Tracking-Learning-Detection (TLD) is tracking algorithm for long-term tracking of unknown objects in real-time. The task includes identifying the location of object via bbox or indicating its absence. It uses three components for the task: The tracker follows the object from frame to frame. The detector localizes all appearances that have been observed so far and corrects the tracker if necessary. The learning estimates detector's errors and updates it to avoid these errors in the future. Novel learning method (P-N learning) is used which estimates the errors by a pair of "experts": (i) P-expert estimates missed detections and (ii) N-expert estimates false alarms.
TLD decomposes the tracking task into three different simultaneously operating components as shown in Fig. 2.1

- **Tracking**: Estimates the object's motion between consecutive frames under the assumption that the frame-to-frame motion is limited and the object is visible. The tracker is likely to fail and never recover if the object moves out of the camera view.

- **Detection**: Treats every frame as independent and performs full scanning of the image to localize all appearances that have been observed and learned in the past. As any other detector, the detector makes two types of errors: false positives and false negative.

- **Learning**: Observes performance of the detector, estimates its errors and generates training examples to avoid these errors in the future. The learning component assumes that both the tracker and the detector can fail. By the virtue of the learning, the detector generalizes to more object appearances and discriminates against background.

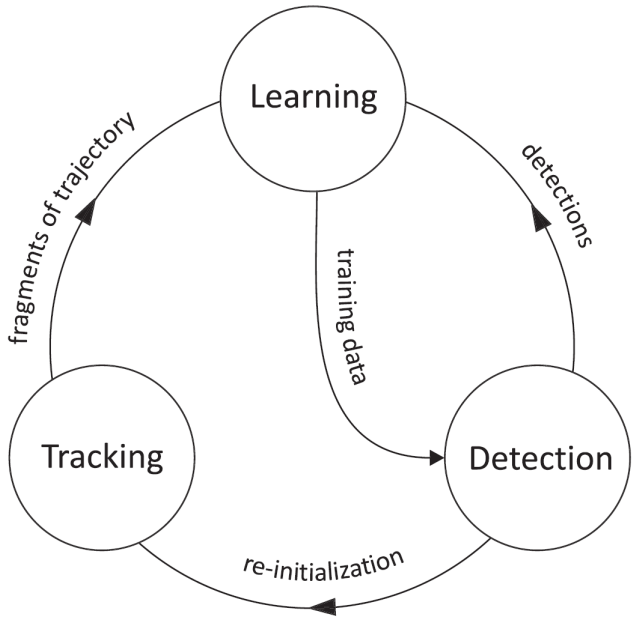


Figure 2.1: Block diagram of TLD framework[1]

2.0.1 Tracking

The tracking component of TLD is based on Median-Flow tracker [6] extended with failure detection. Median-Flow tracker represents the object by a bbox and estimates its motion between consecutive frames. Internally, the tracker estimates displacements of points within the object's bbox, estimates their reliability, and votes with 50% of the most reliable displacements for the motion of the bbox using median. A grid of 10X10 points is used and their motion is estimated using pyramidal Lucas-Kanade tracker [7].

2.0.1.1 Median-Flow tracker and Forward-Backward(FB)error

Reliability of the points is estimated using forward-backward error calculation and normalized cross-correlation (NCC) [6]. Only the points above the median FB error and NCC are selected for predicting object bbox in next frame. The concept of FB error is as follows: Let $S = (I_t, I_{t+1}, I_{t+k})$ be the sequence of frames and x_t be the point to be tracked. Let the tracked trajectory through k frames is $T_k^f = (x_t, x_{t+1}, x_{t+k})$ where f stands for forward and k is length. Our goal is to estimate the error (reliability) of trajectory T_k^f given the image sequence S

For this purpose, the validation trajectory is first constructed by projecting $x_{(t+k)}$ backwards until first frame which produces $T_k^b = (\hat{x}_t, \hat{x}_{t+1}, \hat{x}_{t+k})$ where $\hat{x}_{(t+k)}$ corresponds to $x_{(t+k)}$ The FB error is defined as the distance between these two trajectories: $\mathbf{FB}(T_k^f|S) =$

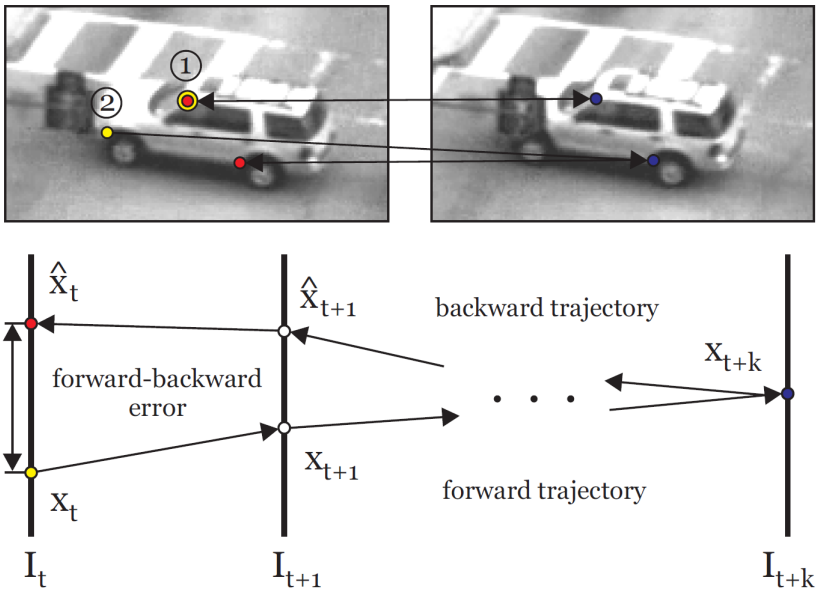


Figure 2.2: The Forward-Backward error penalizes in- consistent trajectories. Point 1 is visible in both im- ages, tracker works consistently forward and back- ward. Point 2 is occluded in the second image, forward and backward trajectories are inconsistent.[1]

$distance(T_k^f, T_k^b)$ Various distances can be calculated for comparison. Here, simple Euclidean distance is calculated, distance $(T_k^f, T_k^b) = ||x_t - \hat{x}_t||$

Along with FB error, NCC is calculated between the points from the two frames for each point. Reliable points are those which have FB error less than median FB error and NCC higher than median NCC. These points are used for predicting the bbox in next frame. It has been shown that combination of FB and NCC produce better results than individual or combination of FB+NCC+SSD. SSD refers to sum of squared differences[6]

Estimation of the bbox displacement from the reliable points is performed using median over each spatial dimension. Scale change is computed as follows: for each possible pair of points inside bbox, a ratio between current point distance and previous point distance is computed; bbox scale change is defined as the median over these ratios. An implicit assumption of the point-based representation is that the object is composed of small rigid patches i.e., object is rigid to a good extent. Parts of the objects that do not satisfy this assumption (object boundary, flexible parts) are not considered in the voting since they are rejected by the error measure.

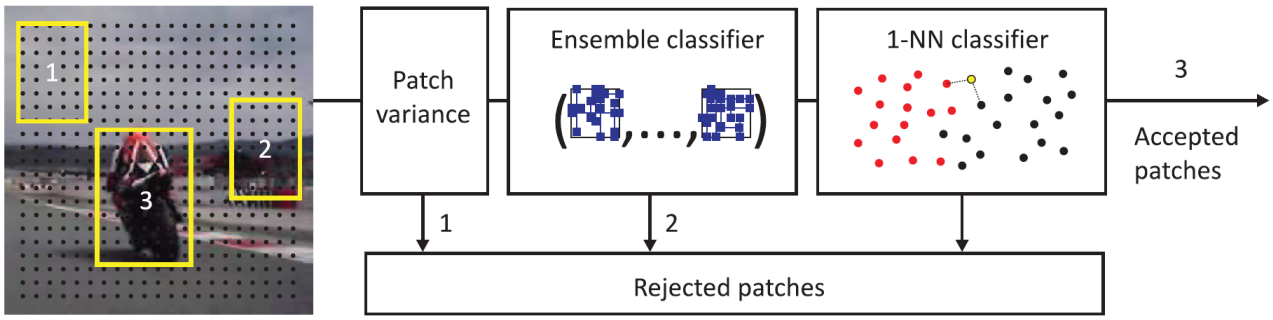


Figure 2.3: Block diagram of detector: Patch variance filter removes the background patches (bbox1), ensemble classifier removes more complicated patches (bbox2) and NN-classifier outputs the object patch (bbox3), rejecting other patches[1]

2.0.2 Detection

The task of detector is localization of the object in the current frame. Detector uses a discriminative classifier to distinguish the positive (object patches) examples from negative (background) ones. It uses sliding window technique to scan through the image partitioned as grid (called scanning grid) consisting of windows of different sizes and shift. As in [8] feature extraction is done using pixel comparisons generated randomly at the starting of the process. For each patch under the candidate window an ensemble of classifiers apply these pixel comparisons, to produce 13-bit binary code. Weights are given based on number of patches having same code which further help in classifying the patch.

Generation of scanning grid

A grid of windows is generated by using different scaling and shifting parameter, applied on the initial specified object bbox, Values of parameters are: scales step=1.2, horizontal shift=10% of initial width, vertical step=10% of initial height, minimal bbox size=24 pixels. This calculation generates about 30,000 windows for a 240X320 frame.

Cascade Structure

The structure of the detector is cascaded consisting of three different stages. First stage consists of variance filter to remove background patches. Next stage is above mentioned ensemble classifier. And last stage is Nearest-Neighbour classifier (NN-classifier). This stage improves upon the work done by ensemble classifier by checking the resemblance of the patches outputted from second stage with the progressive object model.

2.0.2.1 Patch variance

Patch Variance is the first stage of the detector. This filter removes the patches from the scanning grid having gray-scale variance less than 50% of the variance of the initial bbox. Thus, it typically removes 50% of the non-object patches. But restriction on variance puts a restriction on maximal appearance change of the object. To avoid this, threshold should be adjusted according to the application. For this work it was kept constant at 0.5.

2.0.2.2 Ensemble Classifier

Ensemble classifier is the second stage of the detector and handles the non-rejected patches from first stage. Here 10 base classifiers form the ensemble which apply 13 random pixel comparisons on every incoming patch resulting in a 13 bit binary code x, $p_i(y|x)$ which indexes to an array of posteriors, where y = 0; 1 acts as label for the patch, x is the 13 bit code, i is the i^{th} classifier.
This posteriors are averaged over the whole ensemble and the patch is classified as positive if it is above the threshold, which is updated after every frame based on maximum of previous threshold and current array of posteriors.

Pixel comparisons

Every base classifier is based on a set of pixel comparisons. Similarly as in [8] the pixel comparisons are generated offline at random and stay fixed in run-time. First, the image is convolved with a Gaussian kernel with standard deviation of 3 pixels to increase the robustness to shift and image noise. Next, the predefined set of pixel comparison is stretched to the patch. Each comparison returns 0 or 1 and these measurements are concatenated into x to form a 13 bit binary code.

Posterior probability

Posterior probability is calculated as follows:

$$p_i(y|x) = \frac{\#p}{\#p + \#n}$$

where $\#p$ and $\#n$ are positive and negative patches respectively having same binary code x and y is the output label (1 or 0).

On online training data generated by learning component, the aforementioned pixel comparisons are applied and 13-bit binary codes are calculated for each patch by each base classifier. Based on these codes and number of patches having same code, each base classifier calculates weights (posterior) for that particular code. Since 8192 codes are possible for 13 comparisons($2^{13} = 8192$) for each classifier, an array of size 8192 elements is used to store the above calculated weights. When test data comes, again pixel comparisons are applied and codes are calculated by each classifier. If the code matches any of code observed in training data, that classifier gives the weight stored in the array for that code. This is done by all classifiers and finally average is taken across the weights given by each classifier, and if average is above threshold, that patch is classified positive

2.0.2.3 Nearest Neighbour classifier (NN-classifier)

The non-rejected patches from ensemble classifier come here. Patches which are still undecided (patches whose 13-bit code does not match to code observed during training) end up here. Therefore, we can use the online object model, M (explained below) and classify the patch using a NN-classifier. To understand the working of NN-classifier, some pre-requisites are:

1. Similarity S between two patches p_i p_j :

$$S(p_i, p_j) = 0.5(NCC(p_i, p_j) + 1) \tag{2.1}$$

2. Object model M is the data structure which represents the object and its surroundings observed so far. It is collection of positive and negative patches.$M = (p_1^+, p_2^+, ...p_m^+, p_1^-, p_2^- ....., p_n^-)$ where p^+ and p^- are positive and negative patches.

3. Given the patch p_j and object model M, different similarity measures are:

   - Similarity with positive nearest neighbour:

$$S^+(p, M) = max_{p_i^+ \in M} S(p, p_i^+) \tag{2.2}$$

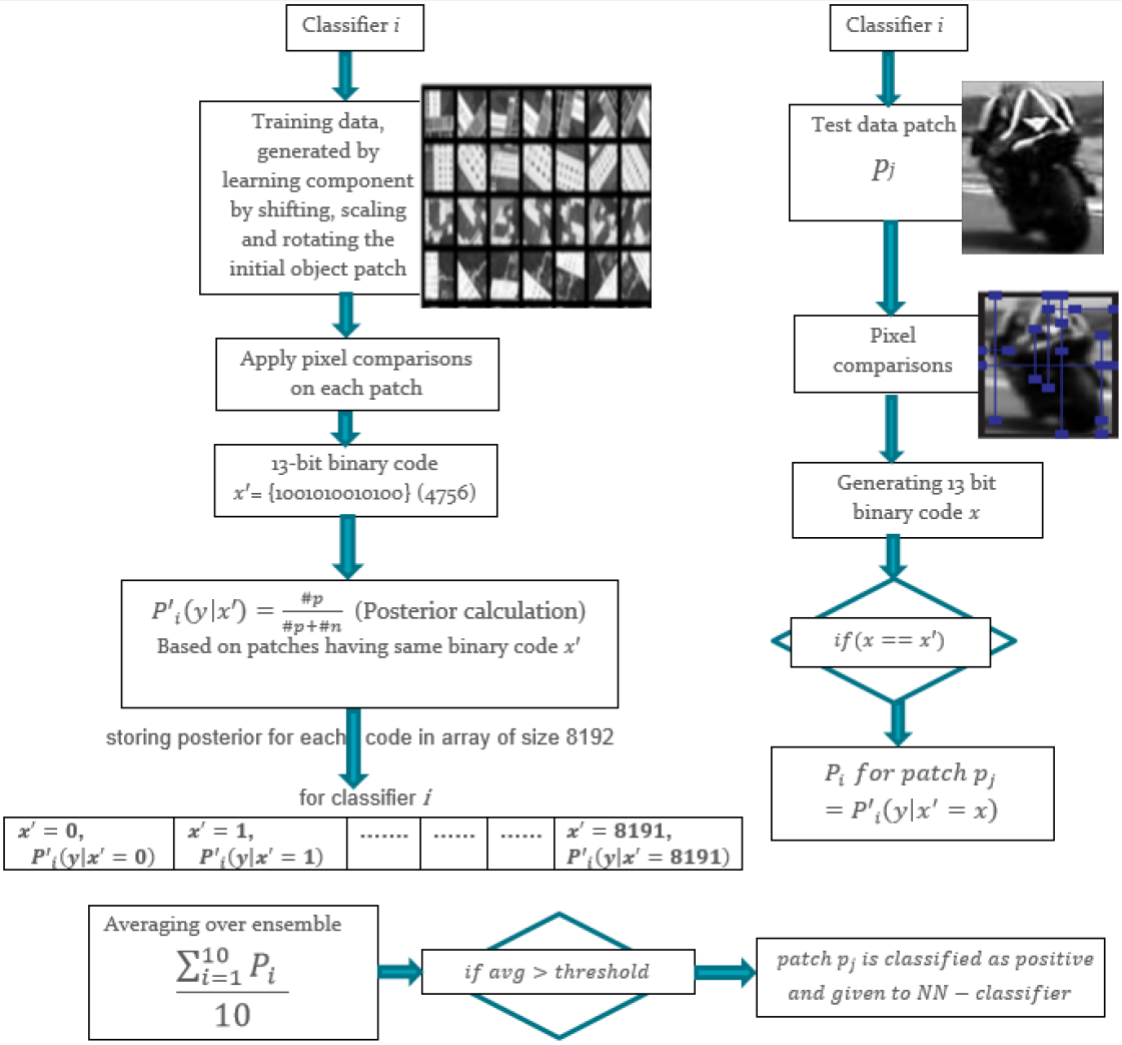


Figure 2.4: Ensemble classifier: Different steps performed by a classifier i of the ensemble: First vertical flow chart shows work on training data; Second(vertical) flow chart shows work during test data; Third horizontal flow chart shows final averaging over whole ensemble[1]

- Similarity with nearest negative neighbour:

$$S^{-}(p, M) = max_{(p_i^- \in M)} S(p, p_i^-) \tag{2.3}$$

- Similarity with the positive nearest neighbour considering 50% of earliest positive patches:

$$S^{+}_{50\%}(p, M) = max_{p_i^+ \in M \wedge i < \frac{m}{2}} S(p, p_i^+) \tag{2.4}$$

- **Relative similarity:** Measures the confidence of the patch p depicting the object:

$$S^r = \frac{S^+}{S^+ + S^-} \tag{2.5}$$

  S^r ranges from 0 to 1. Higher value means patch depicts object.

- **Conservative similarity:** Measures the confidence with which the patch p resembles appearance observed in the first 50% of the positive patches.

$$S^c = \frac{S^+_{50\%}}{S^+_{50\%} + S^-} \tag{2.6}$$

  S^c ranges from 0 to 1.

The Relative similarity is used to define the NN-classifier. When patch p has relative similarity greater than threshold i.e.$S^r(p, M) > \theta_{NN}$ it is classified as positive otherwise negative. When the number of templates in NN-classifier(the size of M)exceeds some threshold(given by memory), random forgetting of templates is done (i.e., randomly templates are removed from M). Number of templates stabilize around several hundred which can be easily stored in memory.

The flow chart of ensemble classifier is shown in Fig. 2.4

2.0.3 Learning

The learning component of TLD is responsible for updating detector. It brings in new object appearances and thus help detector in avoiding errors during object identification during run-time. The task of learning is handled by two experts: $P - expert$ and

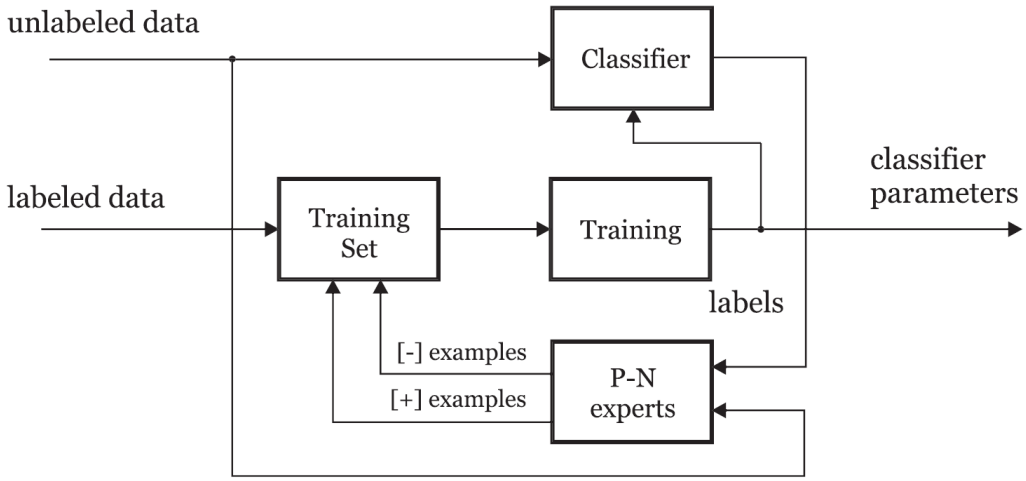


Figure 2.5: Block diagram of PN-learning [1]

$N - expert$ and thus the learning is called P-N Learning. $P - expert$ works on examples classified as negative, estimates false negatives and adds them to training set with positive label. The $P - expert$ increases the classifier's generality. $N - expert$ works on examples classified as positive, estimates false positives and adds them with negative label to the training set. $TheN - expert$ increases the classifier's discrimination ability.

2.0.3.1 Initialization

In the first frame, the learning component trains the initial detector using labelled examples generated as follows. The positive training examples are synthesized from the initial bounding box. First 10 bounding boxes are selected on the scanning grid that are closest to the initial bounding box.
For each of the bounding box, 20 warped versions are generated by geometric transformations (shift 1%, scale change 1%, in-plane rotation 10^o and are added with Gaussian noise $(\sigma = 5)$ on pixels. The result is 200 synthetic positive patches. Negative patches are collected from the surrounding of the initializing bounding box, no synthetic negative examples are generated.

The labelled training patches are then used to update the ensemble classifier as discussed in subsection 2.2.2.2, After the initialization the object detector is ready for run-time and to be updated by a pair of P-N experts.

2.0.4 P-Expert

The goal of P-expert is to discover new appearances of the object and thus increase generalization of the object detector. P-expert exploits the fact that the object moves on a trajectory and add positive examples extracted from such a trajectory. However, in the TLD system, the object trajectory is generated by a combination of a tracker, detector and the integrator. This combined process traces a discontinuous trajectory, which is not correct all the time. The challenge of the P-expert is to identify reliable parts of the trajectory and use it to generate positive training examples. To identify the reliable parts of the trajectory, the P-expert relies on an object model M. An illustration is shown in Fig 2.6.

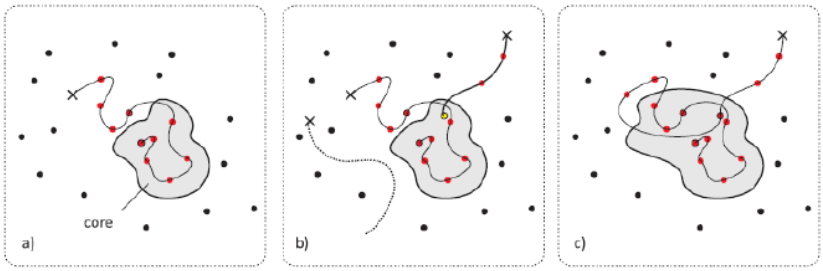


Figure 2.6: Illustration of P-expert. (a) Object model and the core in feature space (gray blob). (b) Unreliable (dotted) and reliable (thick) trajectory. (c) The object model and the core after the update. Red dots are positive examples, black dots are negative, and cross denotes end of a trajectory [1]

Consider an object model represented as coloured points in a feature space. Positive examples are represented by red dots connected by a directed curve suggesting their order, negative examples are black. Using the conservative similarity S^c, one can define a subset in the feature space, where S^c is larger than a threshold. We refer to this subset as the core of the object model. Note that the core is not a static structure, but it grows as new examples are coming to the model. However, the growth is slower than the entire model. P-expert identifies the reliable parts of the trajectory as follows: The trajectory becomes reliable as soon as it enters the core and remains reliable until is reinitialized or the tracker identifies its own failure. Fig. 3.3b illustrates the reliable and non reliable trajectory in feature space. And Fig. 3.3c shows how the core changes after accepting new positive examples from reliable trajectory. In every frame, the P-expert outputs a

decision about the reliability of the current location (P-expert is an online process). If the current location is reliable, the P-expert generates a set of positive examples that update the object model and the ensemble classifier. We select 10 bounding boxes on the scanning grid that are closest to the current bounding box. For each of the bounding box, we generate 10 warped versions by geometric transformations (shift $\pm 1\%$, scale change $\pm 1\%$, in-plane rotation $\pm 5^o$) and add them with Gaussian noise ($\sigma = 5$). The result is 100 synthetic positive examples for ensemble classifier.

Illustration of P-N learning

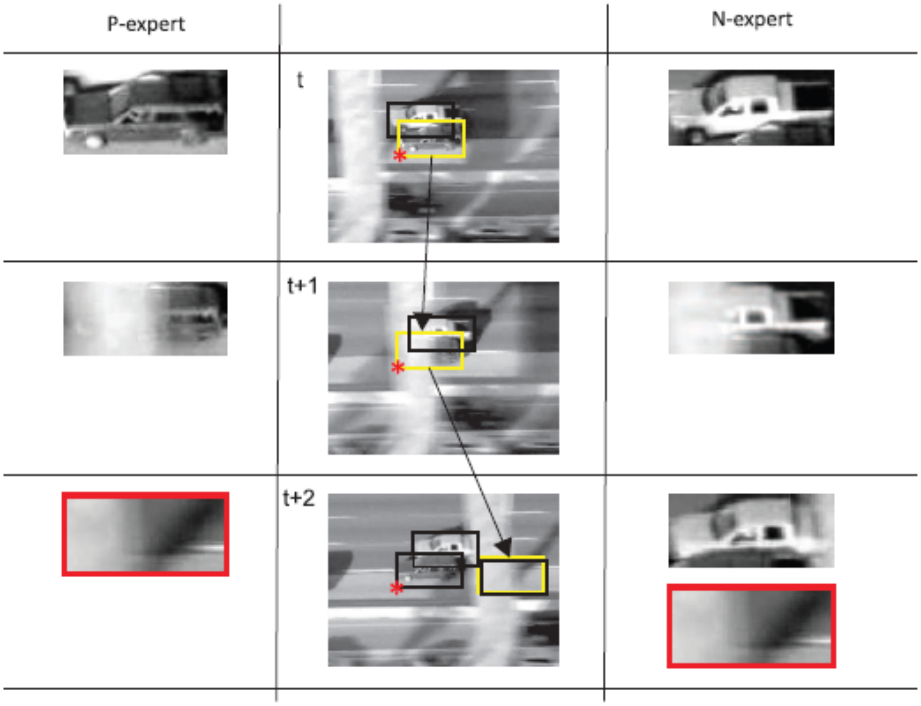


Figure 2.7: Illustration of the examples output by the P-N experts. The third row shows error compensation [1].

2.0.5 N-Expert

N-expert generates negative training examples. Its goal is to discover clutter in the background against which the detector should discriminate. The key assumption of the N-expert is that the object can occupy at most one location in the image. Therefore, if the object location is known, the surrounding of the location is labelled as negative. The N-expert is applied at the same time as P-expert, i.e., if the trajectory is reliable. In that

case, patches that are far from current bounding box (overlap < 0.2) are all labelled as negative. For the update of the object detector and the ensemble classifier, we consider only those patches that were not rejected either by the variance filter or the ensemble classifier.

Fig. 2.7. depicts a sequence of three frames, the object to be learned is a car within the yellow bounding box. The car is tracked from frame to frame by a tracker. The tracker represents the P-expert that outputs positive training examples. Notice that due to occlusion of the object, the output of P-expert in time t+2 outputs incorrect positive example. N-expert identifies a maximally confident patch (denoted by a red star) and labels all other detections as negative. Notice that the N-expert is discriminating against another car and, in addition, corrected the error made by the P-expert in time t+2.

2.0.5.1 TLD Architecture :

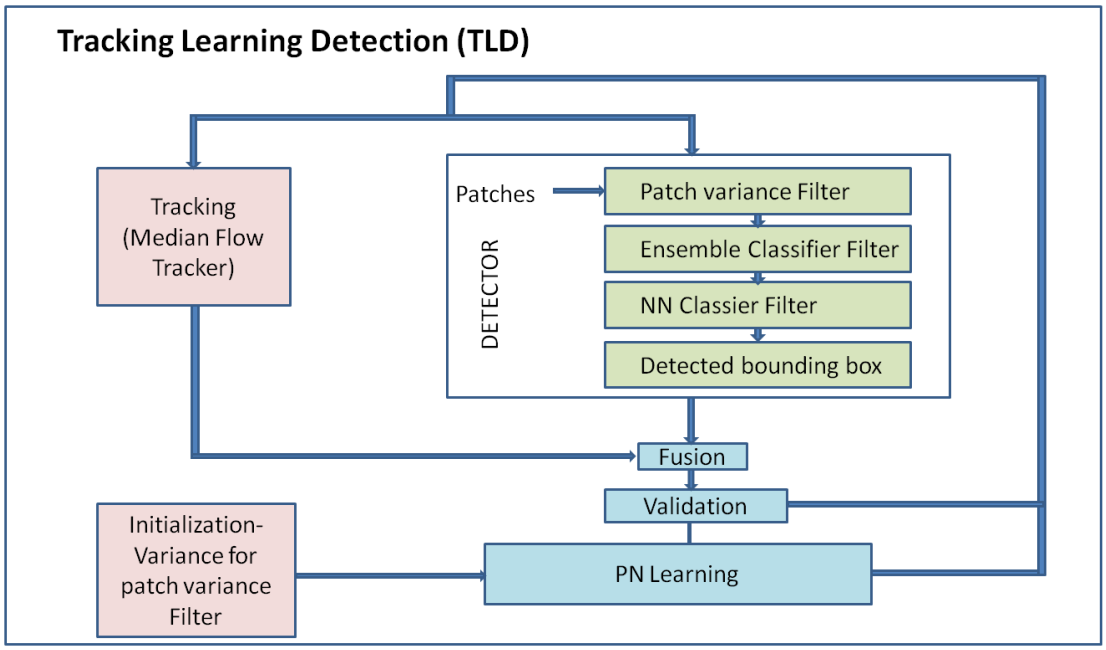


Figure 2.8: TLD Architecture

CHAPTER 3

Kernelized Correlation Filter (KCF) Tracker

3.1 Concepts behind KCF

3.1.1 Kernel trick

In brief, a kernel can be considered as a shortcut that helps us do certain calculation faster which otherwise would involve computations in higher dimensional space.

Intuitively, a kernel is just a transformation of your input data that allows us to process it more easily. Consider the scenario of separating the red circles from the blue crosses on a plane as shown in Fig 2.1

On the left figure, the separating surface would be the ellipse drawn. However, let us

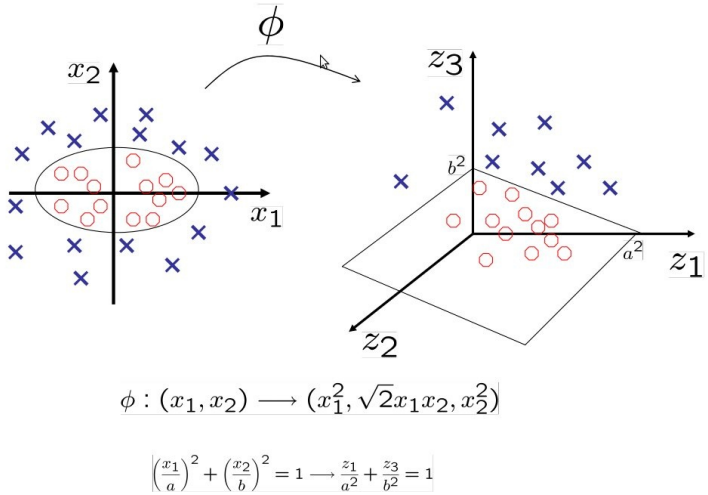


Figure 3.1: Mapping from low dimension to high dimension

assume a function ϕ has transformed our data into a 3 dimensional space through the mapping as shown in the figure. This would make the problem much easier since, now, our points are separated by a simple plane. This embedding on a higher dimension is called the kernel trick.

Mathematical definition:

Let the inputs x, y are n dimensional and assume that the function maps from n-

dimension to m-dimension space, then the kernel function K, is given by

$$K(\mathbf{x}, \mathbf{y}) =< f(\mathbf{x}), f(\mathbf{y}) >$$

where

$< \mathbf{x}, \mathbf{y} >$ denotes the dot product.

In general m is much larger than n.

In order to compute $< f(\mathbf{x}), f(\mathbf{y}) >$, it is essential for us to calculate $f(\mathbf{x})$, $f(\mathbf{y})$ first, and then do the dot product. Since they involve manipulations in m dimensional space, where m can be a large number, the above two computation steps can be quite expensive. The result of the dot product is a scalar. After all the huge number of computations in the high dimensional space, the result which we desire is a scalar. Hence it is nothing but coming back to one-dimensional space. If we can find an appropriate kernel we don't really need to go through all the trouble i.e. to go to the m-dimensional space to get this one number.

Example

Let the inputs x,y be in 3-dimensions.i.e...

$\mathbf{x} = (x_1, x_2, x_3)$

$\mathbf{y} = (y_1, y_2, y_3)$

Consider the mapping function f from 3-dimensions to 9-dimensions be

$f(\mathbf{x}) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)$, and the kernel is

$K(\mathbf{x}, \mathbf{y}) = (< \mathbf{x}, \mathbf{y} >)^2$.

To show how computations complexities can be reduced, let us plug in some numbers

Take

$\mathbf{x} = (1, 2, 3)$

$\mathbf{y} = (4, 5, 6)$

Then

$f(\mathbf{x}) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$

$f(\mathbf{y}) = (16, 20, 24, 20, 25, 36, 24, 30, 36)$

$< f(\mathbf{x}), f(\mathbf{y}) >= 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$

Here a lot of algebra has been done because f is a mapping from 3-dimensional to 9 dimensional space.

Now let us use the kernel instead:

$K(\mathbf{x}, \mathbf{y}) = (4 + 10 + 18)^2 = 32^2 = 1024$

The same result has been obtained with a fewer number of calculations.

The advantage of kernels is that they allow us to perform operations in infinite dimensions. At times it is not only just computationally expensive, but also impossible to go to higher dimension i.e. $f(\mathbf{x})$ can be a mapping from n dimension to infinite dimension which we may have little idea of how to deal with. Thus kernel becomes handy in such situations.

Applications of kernels

If we consider the definition of kernel above, $< f(\mathbf{x}), f(\mathbf{y}) >$, in the context of f being a feature vectors, the inner product means the projection of $f(\mathbf{x})$ onto $f(\mathbf{y})$ or colloquially, how much overlap do $\mathbf{x}$ and $\mathbf{y}$ have in their feature space. In other words, how similar they are. So kernel can also be understood as a measure of similarity.

Relation to SVM:

The idea of SVM is that for $y = \mathbf{w}\phi(\mathbf{x}) + b$,

where,

$\mathbf{w}$ is the weight,

ϕ is the feature vector, and

b is the bias.

If y > 0, then we classify data to class 1, else to class 0. The objective is to find a set of weight and bias such that the margin is maximized. For the data which is linearly separable $\phi(\mathbf{x}) = \mathbf{x}$. For data which is not linearly separable the feature vector $\phi(\mathbf{x})$ which transforms $\mathbf{x}$ to higher dimensions makes the data linearly separable. Kernels helps to make the calculation process faster and easier.

Types of Kernels

Dot-product kernels have the form $k(\mathbf{x}, \mathbf{x}') = g(\mathbf{x}^T\mathbf{x}')$,for some function g.

polynomial kernel is a peculiar form of dot-product kernels which has the form $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + a)^b$.

Radial Basis Function(RBF) kernels have the form $k(\mathbf{x}, \mathbf{x}') = h(\|\mathbf{x} - \mathbf{x}'\|^2)$, for some function h.

Gaussian kernel is a special case of RBF kernel which has the form $exp\Big(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{x}'\|^2\Big)$.

3.1.2 Histogram of Oriented Gradients (HOG)

Histogram of oriented gradients (HOG)[9] is a feature descriptor that is used to detect objects in computer vision and image processing.

The essential thought behind the histogram of oriented gradients descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions.

Implementation of the HOG descriptor algorithm is as follows:

Divide the image into small connected regions called cells. Groups of adjacent cells are considered as spatial regions called blocks. The adjacent figure shows the division of an image into blocks and cells. For each cell compute a histogram of gradient directions or edge orientations for the pixels within the cell. Discretize each cell into angular bins according to the gradient orientation. Each cell's pixel contributes weighted gradient to its corresponding angular bin. Normalized group of histograms represents the block histogram. The set of these block histograms represents the descriptor.

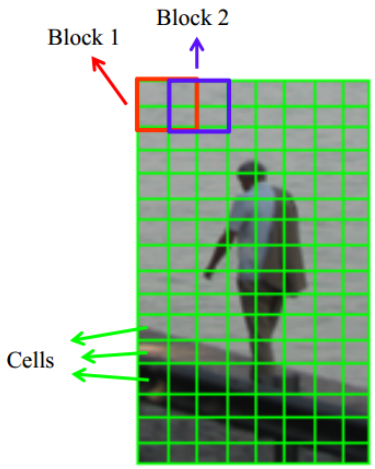


Figure 3.2: Blocks and cells in HOG

Example

For a 64×128 image,

• Divide the image into 16×16 blocks of 50 % overlap.Hence 7×15=105 blocks in total.

• Each block should consist of 2×2 cells with size 8×8.

• Compute centered horizontal and vertical gradients with no smoothing. Compute gradient orientation and magnitudes. For color image, pick the color channel with the highest gradient magnitude for each pixel.

• Quantize the gradient orientation into bins

• The vote is the gradient magnitude. Interpolate votes bi-linearly between neigh-

bouring bin center. The vote can also be weighted with Gaussian to downweight the pixels near the edges of the block.

• Concatenate histograms. Hence the Feature dimension is 105×4×no.of bins.

3.1.3 Circulant Matrices

Consider an n×1 vector (signal) denoted by $\mathbf{x} = [a_1, a_2, \ldots, a_n]^T$. Let us refer to it as the base sample. The one-dimensional translations of this vector can be modelled by a cyclic shift operator, which is the permutation matrix given by,

$$P = \begin{bmatrix} 0 & 0 & 0 & \ldots & 1 \\ 1 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \end{bmatrix}$$

The product $P\mathbf{x} = [a_n, a_1, \ldots, a_{n-1}]$ shifts $\mathbf{x}$ by one element, modelling a small translation. To achieve a larger translation, we can chain 'u' shifts by using the matrix power $P^u\mathbf{x}$. To shift $\mathbf{x}$ in the reverse direction a negative u should be chosen.
Due to the cyclic property, we get the same signal $\mathbf{x}$ periodically after every n shifts. This means that the full set of shifted signals is obtained with

$$\{P^u\mathbf{x} | u = 0, 1, \ldots, n-1\}$$

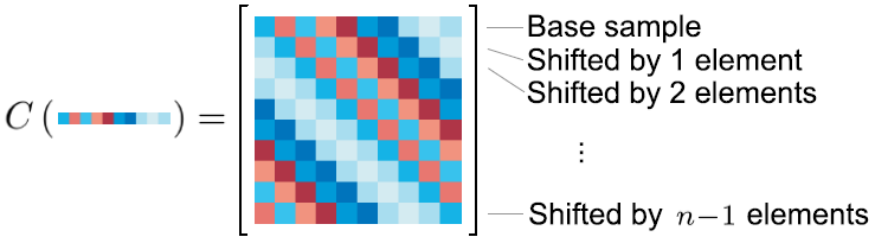


Figure 3.3: Illustration of a circulant matrix

Again due to the cyclic property, we can equivalently view the first half of this set as shifts in the positive direction, and the second half as shifts in the negative direction. A 1D signal translated horizontally with this model is illustrated in the Fig. 3.3.

The resulting pattern which we have just arrived at is a circulant matrix, which has several intriguing properties. The circulant matrix $C(\mathbf{x})$ where $\mathbf{x}$ being the base sample can be represented as

$$X = C(\mathbf{x}) = \begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ a_n & a_1 & a_2 & \dots & a_{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_2 & a_3 & a_4 & \dots & a_1 \end{bmatrix} \tag{3.1}$$

It can be noticed that the pattern is deterministic, and fully specified by the generating vector $\mathbf{x}$, which is the first row. The most useful and amazing property of the circulant matrices is that all circulant matrices can be made diagonal by the Discrete Fourier Transform (DFT), regardless of the generating vector $\mathbf{x}$[10]. This can be expressed as

$$X = Fdiag(\hat{\mathbf{x}})F^H \tag{3.2}$$

where,

F is a constant matrix that does not depend on $\mathbf{x}$, and

$\hat{\mathbf{x}}$ denotes the DFT of the generating vector, $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{x})$. Hat is used as shorthand notation for the DFT of a vector.

The constant matrix F is known as the DFT matrix, and is the unique matrix that computes the DFT of any input vector$\mathbf{z}$, as $\mathcal{F}(\mathbf{z}) = \sqrt{n}F\mathbf{z}$. This is possible because the DFT is a linear operation. Eq. 3.2 expresses the eigen decomposition of a general circulant matrix. The shared, deterministic eigen vectors F lie at the root of many uncommon features, such as commutativity or closed-form inversion.

3.2 Algorithm

KCF tracker [5] assumes that the frame to frame movement of the target is small and that the target doesn't go out of the frame. So in order to find the target in the subsequent frame we need to search over a region around the initial position of the target. Except the target patch all other patches in the search region are considered as negative samples. All these negative samples are approximated as the translated versions of the

target patch. An analytical model for image patches extracted at different translations is proposed. Regression helps in differentiating the target sample from the negative samples. The following figure shows the vertical cyclic shifts of base(target) sample.

Figure 3.4: Examples of vertical cyclic shifts of a base sample

3.2.1 Linear Regression

Regression

Modelling the relationship between a scalar dependent variables Y and one or more explanatory variables (or independent variables) denoted X can be termed as regression. The relationship can be obtained by having a training data (x_i, y_i). It is used to fit a predictive model to an observed data set of y and x values. After developing such a model, if an additional value of x is then given without its accompanying value y, the fitted model can be used to make a prediction of the value of y.

Assume the training data be points (x,y) and the regression is linear then we will get a function of the form y = mx + c. Regression helps in finding m,c. Using this we can find the value of y for any given x.

Now let us assume the independent variables be 1D signals and the regression targets are scalar values. The goal of training is to find a function $\mathbf{f(z)} = \mathbf{w^T z}$ that minimizes the squared error over samples $\mathbf{x_i}$ and their regression targets y_i [11]

$$min_w \sum_{i=1}^{n} (f(\mathbf{x_i}) - y_i)^2 + \lambda \left\|\mathbf{w}\right\|^2 \tag{3.3}$$

λ is a regularization parameter that controls overfitting, as in the SVM. The minimizer has a closed-form solution, which is given by

$$\mathbf{w} = (X^H X + \lambda I)^{-1} X^H Y \tag{3.4}$$

where,

X is the data matrix has one sample per row $\mathbf{x_i}$,

Y is a column vector containing regression targets y_i

I is an identity matrix.

X^H is the Hermitian transpose,i.e., $X^H = (X^*)^T$, and X^* is the complex-conjugate of X. For real numbers, the equation becomes

$$\mathbf{w} = (X^T X + \lambda I)^{-1} X^T Y \tag{3.5}$$

A large system of linear equations must be solved to compute the solution, which is highly avoided in a real-time setting. But this heavy computation can be reduced if X is a circulant matrix.

When the training data is composed of cyclic shifts then X becomes a circulant matrix. From section 3.1.3 it is seen that it can be diagonalised. For diagonal matrices all operations can be done element-wise on their diagonal elements. Hence it is easy to work with them. Take the term $X^H X$, which can be seen as a non-centered covariance matrix. Using Eq. 3.2 we can rewrite as

$$X^H X = F diag(\hat{\mathbf{x}}^*) F^H F diag(\hat{\mathbf{x}}) F^H \tag{3.6}$$

Additionally, we can eliminate the factor $F^H F = I$. This property is the unitarity of F and can be cancelled out in many expressions. Hence we are left with

$$X^H X = F diag(\hat{\mathbf{x}}^*) diag(\hat{\mathbf{x}}) F^H \tag{3.7}$$

Because operations on diagonal matrices are element wise, we can define the element-wise product as $\odot$ and obtain

$$X^H X = F diag(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}) F^H \tag{3.8}$$

Keeping Eq. 3.8 in the formula for Ridge Regression, Eq. 3.4, we get

$$\mathbf{w} = \left(F diag(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}) F^H + \lambda I\right)^{-1} X^H Y \tag{3.9}$$

By simple algebra, and the unitarity of F, we have

$$\mathbf{w} = \left(F diag(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}) F^H + F\lambda F^H\right)^{-1} X^H Y \tag{3.10}$$

$$= \left(F diag(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda) F^H\right)^{-1} X^H Y \tag{3.11}$$

$$= F diag(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda)^{-1} F^H F diag(\hat{\mathbf{x}}) F^H Y \tag{3.12}$$

$$= F diag\Big(\frac{\hat{\mathbf{x}}^*}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda}\Big) F^H Y \tag{3.13}$$

Then this is equivalent to

$$F\mathbf{w} = diag\Big(\frac{\hat{\mathbf{x}}^*}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda}\Big) FY \tag{3.14}$$

and since for any vector $\mathbf{z}$, $F\mathbf{z} = \hat{\mathbf{z}}$,

$$\hat{\mathbf{w}} = diag\Big(\frac{\hat{\mathbf{x}}^*}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda}\Big) \hat{\mathbf{y}} \tag{3.15}$$

We may go one step further, since the product of a diagonal matrix and a vector is just their element-wise product.

$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{x}}^* \odot \hat{\mathbf{y}}}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda} \tag{3.16}$$

For simplicity, the above derivations are done for single-channel, one-dimensional signals. These results can be generalized to multichannel, two-dimensional images in a straightforward way.

Since the negative samples are the translated versions of the base sample, the training data (base sample + negative samples) form a circulant data matrix.

Since we have $\mathbf{w}$, we can find the value of $\mathbf{f}$ for each sample in the next frame. In general the regression targets are Gaussian. So the base sample is given a higher value as the regression target. So the sample for which we get a maximum value of $\mathbf{f}$ is the target patch. Thus the target's location from frame to frame is identified.

3.2.2 Non linear Regression

To improve accuracy, non linear regression approach is used. Here we are mapping the inputs of linear problem to a non linear feature space $\phi(\mathbf{x})$ with the help of kernel trick.

This process assumes that w takes the form,

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x_i}) \tag{3.17}$$

Here the variables under optimisation are α. Hence

$$f(\mathbf{z}) = \left(\sum_{i=1}^{n} \alpha_i \phi(\mathbf{x_i}) \right)^T \phi(\mathbf{z}) \tag{3.18}$$

$$= \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x_i})^T \phi(\mathbf{z}) \tag{3.19}$$

In non-linear regression the dot product between samples is replaced by kernel i.e. $\phi(\mathbf{x})^T \phi(\mathbf{x}') = k(\mathbf{x}, \mathbf{x}')$. Hence

$$f(\mathbf{z}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{z}, \mathbf{x_i}) \tag{3.20}$$

By substituting Eq 3.20 in the loss equation Eq 3.3 we get

$$L = \sum_{i=1}^{n} \Big(y_i - \sum_{j=1}^{n} \alpha_i k(\mathbf{x_i}, \mathbf{x_j}) \Big)^2 + \lambda \alpha^T K \alpha \tag{3.21}$$

$$= (Y - K\boldsymbol{\alpha})^T (Y - K\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha} \tag{3.22}$$

where

Y is a column vector containing the elements y_i

K is the kernel matrix with elements $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$

$\boldsymbol{\alpha}$ is a vector containing the elements α_i. To find the optimal value of α the loss function needs to be minimised with respect to $\boldsymbol{\alpha}$. Hence

$$\nabla L = 0 \tag{3.23}$$

$$\Rightarrow -K^T (Y - K\boldsymbol{\alpha}) + \lambda K \boldsymbol{\alpha} = 0 \tag{3.24}$$

Since K is symmetric

$$\lambda K \boldsymbol{\alpha} = KY - K^2 \boldsymbol{\alpha} \tag{3.25}$$

$$K(K + \lambda I)\boldsymbol{\alpha} = KY \tag{3.26}$$

Hence we get

$$\boldsymbol{\alpha} = (K + \lambda I)^{-1} Y \tag{3.27}$$

we know that the data matrix X is circulant, we have to check whether the kernel matrix is circulant.

Theorem 1

Given circulant data $C(\mathbf{x})$, the corresponding kernel matrix K is circulant if the kernel function satisfies $k(\mathbf{x}, \mathbf{x}') = k(M\mathbf{x}, M\mathbf{x}')$, for any permutation matrix **M**.
this means, for a kernel to preserve the circulant structure, it must treat all dimensions of the data equally. Fortunately, this includes most useful kernels.
The following kernels satisfy the above theorem :

Radial Basis Function kernels - e.g., Gaussian.

Dot-product kernels - e.g., linear, polynomial.

Additive kernels - e.g., intersection kernels.

Exponentiated additive kernels.

Proof to show that K is circulant

Take

$$K_{ij} = k(P^i \mathbf{x}, P^j \mathbf{x}) \tag{3.28}$$

$$= k(P^{-i} P^i \mathbf{x}, P^{-i} P^j \mathbf{x}) \tag{3.29}$$

Using known properties of permutation matrices, this reduces to

$$K_{ij} = k(\mathbf{x}, P^{j-i} \mathbf{x}) \tag{3.30}$$

By the cyclic nature of P, it repeats every nth power, i.e. $P^n = P^0$. As such, Eq.3.30 is equivalent to

$$K_{ij} = k(\mathbf{x}, P^{(j-i) \bmod n} \mathbf{x}) \tag{3.31}$$

where mod is the modulus operation (remainder of division by n).

We now use the fact the elements of a circulant matrix $X = C(\mathbf{x})$ satisfy

$$X_{ij} = x_{((j-i)\text{mod n})+1} \tag{3.32}$$

that is, a matrix is circulant if its elements only depend on $(j-i)\text{mod n}$. It is easy to check that this condition is satisfied by Eq.3.1, and in fact it is often used as the definition of a circulant matrix. Because K_{ij} also depends on $(j-i)\text{mod n}$, we must conclude that K is circulant as well.

Since K is circulant substitute $K = C(k^{xx})$, where k^{xx} is the first row of K in the formula for Kernel Ridge Regression, Eq.3.27, and diagonalizing it we get

$$\boldsymbol{\alpha} = \Big(C(\mathbf{k^{xx}}) + \lambda I\Big)^{-1} Y \tag{3.33}$$

$$= \Big(F diag(\hat{\mathbf{k}}^{\mathbf{xx}}) F^H + \lambda I\Big)^{-1} Y \tag{3.34}$$

By simple linear algebra, and the unitarity of F ($FF^H = I$),

$$\boldsymbol{\alpha} = \Big(F diag(\hat{\mathbf{k}}^{\mathbf{xx}}) F^H + \lambda F I F^H\Big)^{-1} Y \tag{3.35}$$

$$= F diag(\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda)^{-1} F^H Y \tag{3.36}$$

which is equivalent to

$$F^H \boldsymbol{\alpha} = diag(\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda)^{-1} F^H Y$$

Since for any vector $F\mathbf{z} = \hat{\mathbf{z}}$, we have

$$\hat{\boldsymbol{\alpha}}^* = diag\Big(\frac{1}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda}\Big)\hat{\mathbf{y}}^* \tag{3.37}$$

Finally, because the product of a diagonal matrix and a vector is simply their element-wise product,

$$\hat{\boldsymbol{\alpha}}^* = \frac{\hat{\mathbf{y}}^*}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda} \tag{3.38}$$

Detection

To detect the object of interest, we typically wish to evaluate $f(\mathbf{z})$ on several image locations $\mathbf{z_1}, \mathbf{z_2}, ...\mathbf{z_n}$, i.e., for several candidate patches. These patches can be modelled

by cyclic shifts. So

$$\begin{bmatrix} f(\mathbf{z_1}) \\ f(\mathbf{z_2}) \\ \vdots \\ f(\mathbf{z_n}) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \alpha_i k(\mathbf{z_1}, \mathbf{x_i}) \\ \sum_{i=1}^{n} \alpha_i k(\mathbf{z_2}, \mathbf{x_i}) \\ \vdots \\ \sum_{i=1}^{n} \alpha_i k(\mathbf{z_n}, \mathbf{x_i}) \end{bmatrix} \tag{3.39}$$

(3.40)

Upon expansion we get

$$\mathbf{f}(\mathbf{z}) = \begin{bmatrix} k(\mathbf{x_1}, \mathbf{z_1}) & k(\mathbf{x_2}, \mathbf{z_1}) & \dots & k(\mathbf{x_n}, \mathbf{z_1}) \\ k(\mathbf{x_1}, \mathbf{z_2}) & k(\mathbf{x_2}, \mathbf{z_2}) & \dots & k(\mathbf{x_n}, \mathbf{z_2}) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x_1}, \mathbf{z_n}) & k(\mathbf{x_2}, \mathbf{z_n}) & \dots & k(\mathbf{x_n}, \mathbf{z_n}) \end{bmatrix}^T \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \tag{3.41}$$

where, **f**(z) is a vector, containing the output for all cyclic shifts of z, i.e., the full detection response.

Since $\mathbf{z_1}, \mathbf{z_2}, ...\mathbf{z_n}$ can be modelled as cyclic shifts, from theorem 1 the above expression can also be written as

$$\mathbf{f}(\mathbf{z}) = (K^z)^T \alpha \tag{3.42}$$

where K^z is a circulant symmetric matrix with elements $K^z_{ij} = k(\mathbf{x_i}, \mathbf{z_j})$ Upon diagonilisation, we have

$$\begin{aligned} \mathbf{f}(\mathbf{z}) &= (C(\mathbf{k^{xz}}))^T \alpha && (3.43) \\ &= \left(F diag\left(\hat{\mathbf{k}}^{\mathbf{xz}}\right) F^H \right) \alpha && (3.44) \\ &= F^H diag\left(\hat{\mathbf{k}}^{\mathbf{xz}}\right) F \alpha && (3.45) \end{aligned}$$

which is equivalent to

$$F\mathbf{f}(\mathbf{z}) = diag\left(\hat{\mathbf{k}}^{\mathbf{xz}}\right) F \alpha \tag{3.46}$$

Replicating the same final steps from the previous section,

$$\hat{\mathbf{f}}(\mathbf{z}) = \hat{\mathbf{k}}^{\mathbf{xz}} \odot \hat{\alpha} \tag{3.47}$$

Compute the Fourier inverse of $\hat{\mathbf{f}}$ and find the position where its value is maximum. It gives the amount of displacement that the target patch has been moved from the previous frame. Hence the new location of target has been found.

3.2.3 Computation of Kernel

For single channel features

Dot-product and polynomial kernels

Dot-product kernels have the form $k(\mathbf{x}, \mathbf{x}') = g(\mathbf{x}^T\mathbf{x}')$, for some function g. Then, $k^{\mathbf{xx}'}$ has elements

$$k_i^{\mathbf{xx}'} = k(\mathbf{x}', P^{i-1}\mathbf{x}) = g(\mathbf{x}'^T P^{i-1}\mathbf{x}) \tag{3.48}$$

Let g also work element-wise on any input vector. This way we can write Eq.3.48 in vector form

$$k^{\mathbf{xx}'} = g(C(\mathbf{x})\mathbf{x}') \tag{3.49}$$

This makes it an easy target for diagonalization, yielding

$$k^{\mathbf{xx}'} = g(F^{-1}(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}')) \tag{3.50}$$

where F^{-1} denotes the Inverse DFT

In particular, for a polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + a)^b$

$$k^{\mathbf{xx}'} = (F^{-1}(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}') + a)^b \tag{3.51}$$

Radial Basis Function and Gaussian kernels

RBF kernels have the form $k(\mathbf{x}, \mathbf{x}') = h(\|\mathbf{x}, \mathbf{x}'\|^2)$, for some function h. The elements

of $k^{\mathbf{xx}'}$ are

$$k_i^{\mathbf{xx}'} = k(\mathbf{x}', P^{i-1}\mathbf{x}) = h(\left\|\mathbf{x}' - P^{i-1}\mathbf{x}\right\|^2) \tag{3.52}$$

We will show (Eq.3.52) that this is actually a special case of a dot-product kernel. We only have to expand the norm,

$$k_i^{\mathbf{xx}'} = h(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2\mathbf{x}'^T P^{i-1}\mathbf{x}) \tag{3.53}$$

The permutation P^i does not affect the norm of $\mathbf{x}$ due to Parseval's Theorem. Since $\|\mathbf{x}\|^2$ and $\|\mathbf{x}'\|^2$ are constant w.r.t. i, Eq.3.53 has the same form as a dot-product kernel (Eq. 3.48). Leveraging the result from the previous section,

$$k^{\mathbf{xx}'} = h(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2F^{-1}(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}'})) \tag{3.54}$$

As a particularly useful special case, for a Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = exp(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{x}'\|^2)$ we get

$$k^{\mathbf{xx}'} = exp\left(-\frac{1}{\sigma^2}\left(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2F^{-1}(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}')\right)\right) \tag{3.55}$$

For Multiple Channel features

To deal with multiple channels, assume that a vector $\mathbf{x}$ concatenates the individual vectors for C channels (e.g. 31 gradient orientation bins for a HOG variant), as $\mathbf{x} = [\mathbf{x_1}, ...\mathbf{x_c}]$. Notice that all kernels mentioned above are based on either dot-products or norms of the arguments. A dot product can be computed by simply summing the individual dot-products for each channel. By linearity of the DFT, this allows us to sum the result for each channel in the Fourier domain. As a concrete example, we can apply this reasoning to the Gaussian kernel, obtaining the multichannel analogue of Eq.3.55 as follows,

$$k^{\mathbf{xx}'} = exp\left(-\frac{1}{\sigma^2}\left(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2F^{-1}(\sum_c \hat{\mathbf{x}}_c^* \odot \hat{\mathbf{x}}_c')\right)\right) \tag{3.56}$$

The multichannel analogue of polynomial kernel is given by

$$k^{\mathbf{xx}'} = \left(F^{-1}(\sum_{c} \hat{\mathbf{x}}_c^{*} \odot \hat{\mathbf{x}}_c') + a\right)^b \tag{3.57}$$

Block diagram of KCF

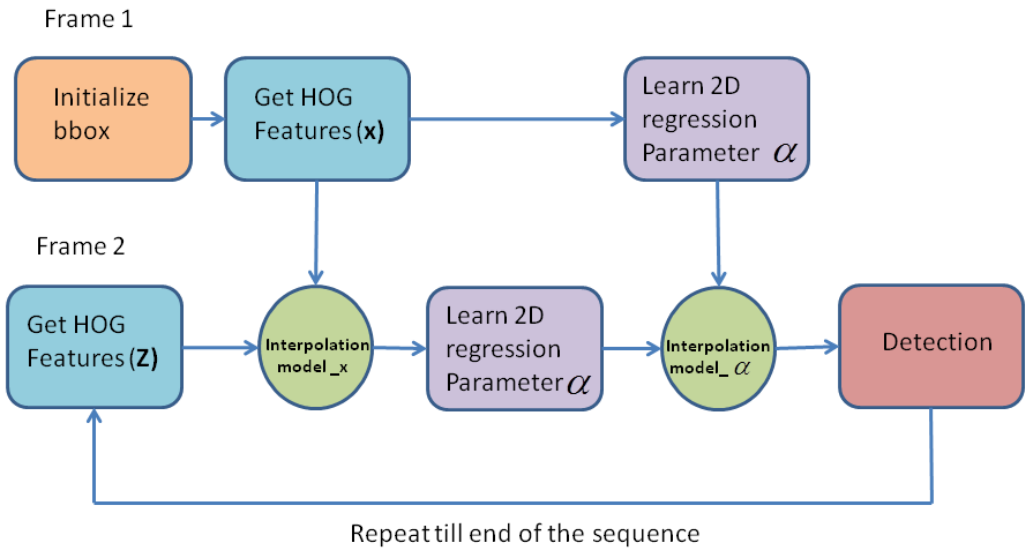


Figure 3.5: Block diagram of KCF

In each frame the features(x) are extracted and the training parameter(α) is computed. In order to give weightage to the features and training parameters obtained from previous frames interpolation is done. The interpolated features are represented as $modelx$ and the interpolated training parameter is represented as $model\alpha$. With the help of $model\alpha$ detection of the target is done.

3.2.4 Algorithm

- **Initialization :** Initialize object bounding box in the first frame and initialize 2D Gaussian data **(y)** as regression targets.

- **Feature extraction :** Consider 2.5 times the size of the initialized bounding box

and extract HOG features (**x**) from it. Initialize $model\_\mathbf{x}$ as **x**.

- **Training :** Using extracted feature data (**x**) train the non linear regression model and calculate the learning parameter α using the equation 3.38. Initialize $model\_\alpha$ as α.

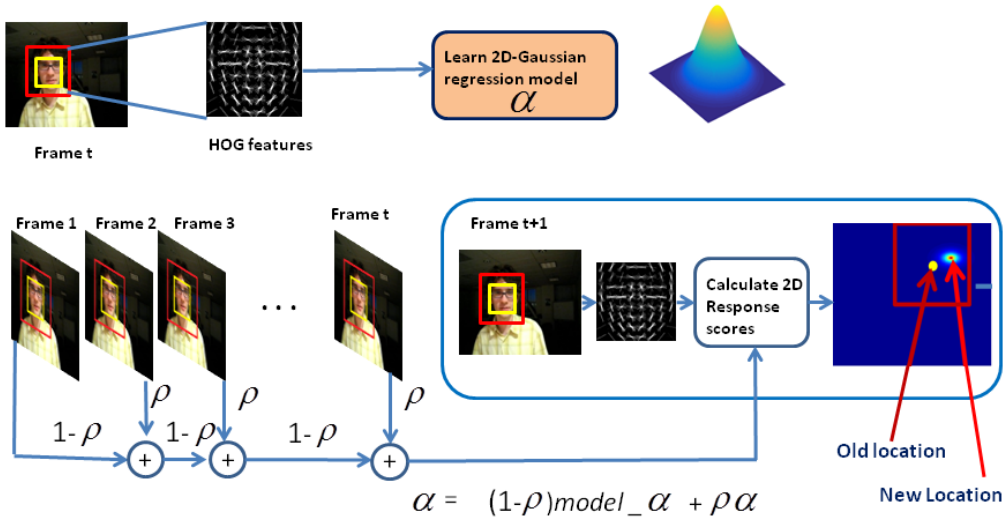


Figure 3.6: Flow chart of KCF

- **Detection :** In the second frame extract the features **z** and calculate the responses using the equation 3.47. Find the shift in peak's position and move the bounding box to new location.

- **Re-learning and Updation :** Update α and **x** values by linear interpolation i.e.

$$\alpha = (1-\rho)model\_\alpha + \rho\alpha$$
$$\mathbf{z} = (1-\rho)model\_\mathbf{x} + \rho\mathbf{z}$$

where $\rho = 0.02$ is the updation parameter.

Repeat the above two steps for all the frames in the sequence.

CHAPTER 4

Compressive Tracker (CT)

4.0.5 Introduction

Compressive Tracker(CT) is an effective and efficient tracking algorithm with an appearance model based on features extracted in the compressed domain. The main components of compressive tracking algorithm are shown by Figure [4.1] CT appearance model is generative as the object can be well represented based on the features extracted in the compressive domain. It is also discriminative because we use these features to separate the target from the surrounding background via a Naive Bayes classifier. In its appearance model, features are selected by an information-preserving and non-adaptive dimensionality reduction from the multi-scale image feature space based on compressive sensing theories [12, 13]. A very sparse measurement matrix that satisfies the restricted isometry property (RIP), thereby facilitating efficient projection from the image feature space to a low dimensional compressed subspace. For tracking, the positive and negative samples are projected (i.e., compressed) with the same sparse measurement matrix and discriminated by a simple naive Bayes classifier learned online.

4.0.6 Random projection and measurement matrix

A random matrix $R \in \Re^{nxm}$ whose rows have unit length projects data from the high dimensional image space $\mathbf{x} \in \Re^m$ to a lower-dimensional space $\mathbf{v} \in \Re^n$

$$\mathbf{v} = R\mathbf{x} \tag{4.1}$$

where $n \ll m$ The Johnson-Lindenstrauss lemma [14] states that with high probability the distances between the points in a vector space are preserved if they are projected onto a randomly selected subspace with suitably high dimensions.Also in [15] proved that the random matrix satisfying the Johnson-Lindenstrauss lemma also holds true for the restricted isometry property in compressive sensing. Therefore, if the random matrix R in equ.[4.1] satisfies the Johnson-Lindenstrauss lemma, then by extension it satisfies

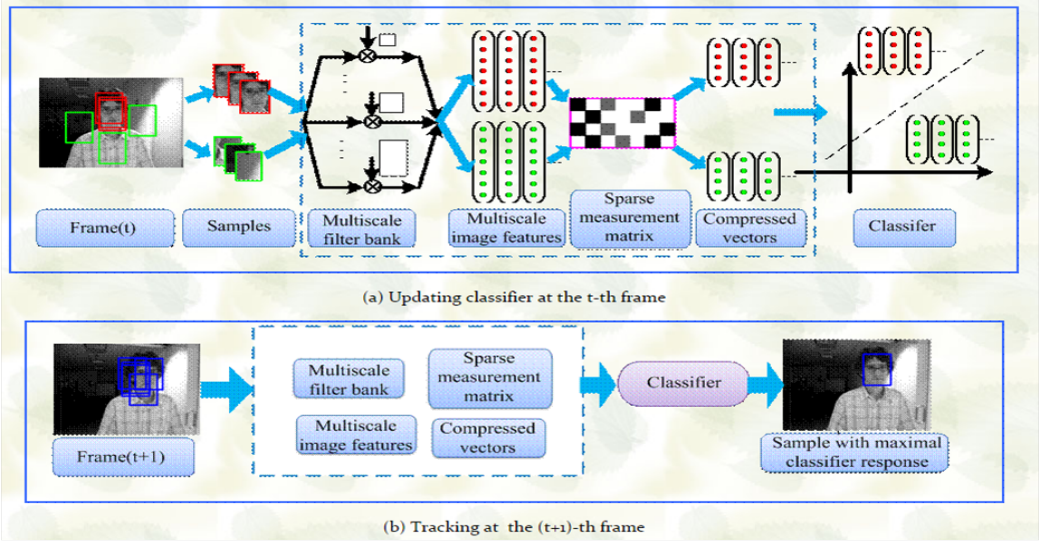


Figure 4.1: Main components of Compressive tracker algorithm[2]

RIP and hence, $\mathbf{x}$ can be reconstructed with minimum error from $\mathbf{v}$ with high probability if $\mathbf{x}$ is compressive such as audio or image. We can ensure that $\mathbf{v}$ preserves almost all the information in $\mathbf{x}$.

4.0.6.1 Random projection and measurement matrix

A typical measurement matrix satisfying the restricted isometry property is the random Gaussian matrix $R \in \Re^{nxm}$ where $r_{ij} \sim N(0,1)$

$$r_{ij} = \sqrt{s} \times \begin{cases} 1 & with\ probability\ \frac{1}{2s} \\ 0 & with\ probability\ 1 - \frac{1}{s} \\ -1 & with\ probability\ \frac{1}{2s} \end{cases} \tag{4.2}$$

It has been proved that with s = 2 or 3, the matrix satisfies John-Lindenstrauss lemma.This matrix is very easy to compute which requires only a uniform random generator. More importantly, when s = 3, it is very sparse where two thirds of the computation can be avoided. Even when $s = O(m)$ the random projections are almost as accurate as the conventional random projections where $r_{ij} \sim N(0,1)$ Here $s = \frac{m}{4}$ is used, thus complexity is reduced to $O(cn)$ which is very less. Also, the memory required to store R is reduced since only non-zero entries need to be stored.

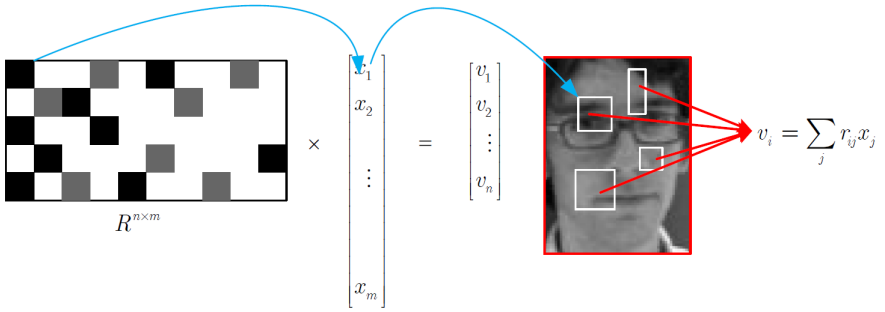


Figure 4.2: **CT algorithm**: Graphical representation of compressing a high dimensional vector x to a low dimensional vector v. In the matrix R, dark, gray and white rectangles represent negative, positive, and zero entries, respectively. The blue arrows illustrate that one of non-zero entries of one row of R sensing an element in x is equivalent to a rectangle filter convolving the intensity at a fixed position of an input image.[2]

4.0.7 Proposed Algorithm

In CT tracking problem is formulated as a detection task with the defined position of the object in first frame.After each frame, around 50 positive samples near the current target location and negative samples far away from the object center are sampled to update the classifier. To predict the object location in the next frame, some samples are drawn around the current target location and the one with the maximal classification score is chosen as output.

4.0.7.1 Dimensionality reduction and efficient feature extraction

The effective dimensionality reduction comes from implementing Haar-like features using a very sparse random matrix as in equation[4.2] The implementation is illustrated in Figure[4.2]

Each sample $z \in \Re^{wxh}$ is represented by convolving it with a set of rectangle filters at multiple scales $\{h_{1,1}.....h_{w,h}\}$ defined as

$$r_{ij}(x,y) = \begin{cases} 1 & 1 \leqslant x \leqslant i; \quad 1 \leqslant y \leqslant j \\ 0 & otherwise \end{cases} \tag{4.3}$$

where *i* and *j* are the width and height of a rectangle filter, respectively. Thus, convolving a sample patch with total *wh* filters gives *wh* responses which are formed into

a vector in R^{wh} These vectors are concatenated as a very high-dimensional multi-scale image feature vector **x** = $(x_1..., x_{wh}..., x_m) \in \Re^m$ where $m = (wh)^2$. The dimensionality of *m* is typically of the order of $10^6 to 10^8$

A sparse random matrix R in equation [4.3] with $s = \frac{m}{4}$ is adopted to project $\mathbf{x}$ onto a vector $\mathbf{v} \in \Re^n$ in a low-dimensional space. The random matrix R needs to be computed only once off-line and remains fixed throughout the tracking process.

4.0.7.2 Classifier construction and Update

For distinguishing the object patch from background, Naive Bayes classifier is used. For each feature v_i in $\mathbf{v}$, likelihood $p_i(v_i|y)$ is calculated, where y is a binary label (0 for background and 1 for object), multiplied along i and the patch having maximal likelihood is selected. Mathematically,

$$H(\mathbf{v}) = \log\left(\frac{\prod_{i=1}^{n} p(v_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(v_i|y=0)p(y=0)}\right) = \sum_{i=1}^{n} \log\left(\frac{p(v_i|y=1)}{p(v_i|y=0)}\right) \tag{4.4}$$

logarithm is introduced for compensating the effect of very extreme values (which may represent a misaligned sample). Here prior is taken as uniform i.e.,$p(y=0) = p(y=1)$and thus the second formula in eqn.[4.4] is used for classifier construction.

It has been shown that the random projections of high dimensional random vectors are almost always Gaussian[23 of [2]].Thus, the conditional distributions $p(v_i|y=1)$ and $p(v_i|y=0)$ in the classifier $H(\mathbf{v})$ are assumed to be Gaussian distributed i.e.,$p(v_i|y=1) \sim N(\mu_i^1, \sigma_i^1)$ and $p(v_i|y=0) \sim N(\mu_i^0, \sigma_i^0)$ So, for each element in $\mathbf{v}$,$p(v_i|y)$ is calculated as aforementioned and than multiplied as in eqn.[4.4] to get the feature value of $\mathbf{v}$. Such $\mathbf{v}$ is calculated for every sample, and the sample having maximum $H(\mathbf{v})$ is given as output.

Classifier update is done after every frame by updating the value of parameters μ and σ measured from features extracted from positive and negative samples.

$$\begin{aligned} \mu_i^1 &\leftarrow \lambda\mu_i^1 + (1-\lambda)\mu_i^1 \\ \sigma_i^1 &\leftarrow \sqrt{\lambda(\sigma_i^1)^2 + (1-\lambda)(\sigma^1)^2 + \lambda(1-\lambda)((\mu_i^1)^2 - (\mu^1)^2)} \end{aligned} \tag{4.5}$$

and similarly μ_i^0 and σi^0 is updated using μ^0 and σ^0. Here, $\lambda > 0$ is learning parameter (typically $0.7 \sim 0.95$, smaller value weighs more on new frames) and μ^1 and σ^1 calculated following way

$$\mu^1 = \frac{1}{n} \sum_{k=0|y=1}^{n-1} v_i(k)$$
$$\sigma^1 = \sqrt{\frac{1}{n} \sum_{k=0|y=1}^{n-1} (v_i(k) - \mu^1)^2} \tag{4.6}$$

where v_i is the element of the feature vector $\mathbf{v}$ extracted from positive samples. Similarly μ^0 and σ^0 are calculated from negative samples

4.0.7.3 Algorithm

The following steps illustrates how compressive tracker works :

1. **Initialization:** Initially, matrix R is generated and the positions and scaling of rectangular filters corresponding to the non-zero entries in R are stored in separate matrix. These matrices are kept constant throughout the process.

2. **Input:** t^{th} video frame

3. **Detection:** A set of patches are sampled in surrounding area of radius γ round bbox location from previous frame for object detection i.e.
   $D^\gamma = \{\mathbf{z} \mid\mid\mid \mathbf{I}(\mathbf{z}) - \mathbf{I}_{t-1} \mid\mid < \gamma\}$ where $\mathbf{I}_{t-1}$ is the tracking location at the $(t-1)^{th}$ frame

4. **Feature extraction** For each $\mathbf{z}$ in D^γ efficient feature extraction with low dimension is done using the convolution shown in Fig.[4.2] which gives features similar to generalized Haar-like features. Thus, now we have a matrix corresponding to each sample and its reduced feature set $\mathbf{v}(\mathbf{z})$.

5. **Classification:** Using equation[4.4] $H(v)$ is calculated for each feature vector $\mathbf{v}(\mathbf{z})$.

6. **Output:** The feature vector $\mathbf{v}(\mathbf{z})$ having maximal classifier response is selected as output and represent current bbox location $\mathbf{I}_t$

7. **Sampling positive and negative patches:** Positive patches in surrounding area of radius α around current bbox $\mathbf{I}_t$ are sampled i.e.
   $D^{\alpha} = \{\mathbf{z} \mid\mid\mid \mathbf{I}(\mathbf{z}) - \mathbf{I}_{t-1} \mid\mid < \alpha\}$ and negative samples in annular region with inner radius τ and outer radius β,$D^{\tau,\beta} = \{\mathbf{z} \mid \tau < \mid\mid \mathbf{I}(\mathbf{z}) - \mathbf{I}_{t-1} \mid\mid < \beta\}$
   with $\alpha < \tau < \beta$

8. **Classifier update:** Feature extraction is done in same way as in step(4) for above sampled positive and negative patches using same matrix R and rectangular filters. From these features, parameters in eqn.[4.6] are calculated and used for classifier update using eqn.[4.5] for next frame.

CHAPTER 5

Critical analysis of Trackers and Proposed method

This chapter includes critical analysis of three algorithms mentioned in the earlier chapters. A deep insight about how each tracker handles the tracking challenges mentioned in Chapter 1 has been provided. This analysis has helped design a novel algorithm. In KCF, tracking by detection is done using regression instead of classification, on which all discriminative tracking methods relies. Kernels are used for non linear regression which resulted in kernelized correlation filtering (KCF), which is a generative method for tracking. where as TLD and CT are discriminative tracking methods. So, the proposed algorithm is a combination of both generative and discriminative methods. A description about the proposed algorithm that combines KCF and TLD has been provided in Section 3.4.

5.1 Critical analysis of TLD

5.1.1 TLD handling challenges :

Scale changes : TLD owes its success to the scanning grid which samples the image into various patches of different size and online learning. Scanning grid allows patches of different sizes to be tested by detector and hence, it does not miss any scale change the object undergoes. The synthetic examples generated by learning component, which includes shifting, scaling and in-plane-rotation helps detector to act robustly for scale changes.

Illumination changes : It can handle illumination changes well as it uses random ferns, which have been shown to successfully overcome illumination variation as they use comparison rather than absolute pixel values for feature generation.

Occlusion : If the video sequence is such that, it provides enough instances for learning the object before its occlusion i.e., after some 10-15 frames, TLD can hang onto object otherwise it never recovers.

Tracking resumption : While considering tracking resumption after object's reappear-

ance, TLD can easily relocate the object (provided there are enough frames for it to learn the object) due to its learning component.

5.1.2 TLD Failure cases :

Full out-of plane rotation : TLD does not perform well in case of full out-of- plane rotation. In that case, the Median-Flow tracker drifts away from the target and can be re-initialized only if the object reappears with appearance seen/learned before.

Fast motion : As TLD uses KLT, even after its iterative pyramidal implementation, it is not able to provide large enough flow vectors for fast tracking without compromising the processing head.

Background clutter : For TLD, P-expert and N-expert provide the required background clutter rejection. N-expert labels all the patches away from the current bbox as negative and hence effectively discriminates background from object.

Non-rigid deformation : : It does not perform well for articulated objects such as pedestrians due to dependency of detector on the training data. If the training data does not have enough information about each individual component and their alignment (different poses) of the non-rigid object, TLD gives random output or shows absence of object.

The average frame rate is approximately 25 FPS and is good enough for real time applications but not as high as KCF. Its mean precision value computed over visual tracker benchmark dataset has been observed as 60.8 % which is lesser than KCF.

5.2 Critical analysis of KCF

The main idea of KCF is based on the fact that for efficient tracking, number of training samples (especially negative) are required more. But if the no. of samples are more for training then the computation complexity also would be more, at the same time there is a large amount of information redundancy. KCF method have exploited this redundancy in data by formulating both the training and testing data as circulant and made computations of $\mathcal{O}(n \log n)$ instead $\mathcal{O}(n^3)$. This has resulted in high speed which approximately is 150 fps in tracking.

5.2.1 KCF handling challenges

Illumination changes : KCF uses correlation technique, its detection mechanism outputs that patch which is highly similar to the initial target patch.Because of illumination change though the correlation factor reduces for all patches the maximum one among them would be the target patch.

Background clutter : The implicit inclusion of thousands of negative patches around the tracked object would results in effectively handling background clutters.

Non-rigid deformation : The robustness of HOG features used in KCF have lead to handle non rigid deformation.

In-plane and out-plane rotations : KCF can handle in-plane rotation, out-of-plane rotation since these rotations will not happen suddenly, by choosing the interpolation factor we can give more priority for recent frames.

5.2.2 KCF Failure cases

Scale changes : Since it is assumed that the object size is fixed and the samples that are taken for identifying the target have the same size as that of the initial target size, KCF cannot handle scale changes.

No tracking resumption : The tracker assumes that the target makes small motion around the initial position. The search region is considered to be only around the initial bounding box. It doesn't search for the target all over the frame. So when the object moves out of the frame and comes back to a position which is far away from the initial position, KCF fails to detect and it can never recover from the failure.

Occlusions : In general KCF cannot handle occlusions. Since the training parameter learns that the object that causes the occlusion as target. Hence the track is lost. But in a special case where the target object is not moving it can handle the occlusion. In this case the occlusion comes and goes away and the target remains at the place as it was before occlusion. KCF can maintain the track since the object that caused the occlusion is now out of the frame.

5.3 Critical analysis of CT

5.3.1 CT handling challenges

Illumination changes : The normalization of features extracted in CT has resulted in handling illumination changes.

Fast motion and Non rigid deformation : The learning parameter λ can be adjusted to give more preference to recent observations which in turn handle the non rigid deformations and fast motions.

5.3.2 CT Failure cases

No tracking resumption : If the object goes out frame and comes back there is no scheme for searching entire frame. The detector in CT searches only 30 pixels radius around the object in the next frame.

Scale changes : The CT assumes target size to be fixed, the detection considers same size as target it cannot cop up with scale changes.

Back ground clutter : Since the number of training samples are less it gets confused in background clutter scenario.

Table 5.1: **Critical analysis of KCF, TLD and CT**

Criteria	KCF	TLD	CT	Remarks
Illumination Variation	Yes	Yes	Yes	Efficient feature extraction in all the cases, instead of using absolute pixel values make them illumination insensitive
Scale Variation	No	Yes	No	KCF and CT assumes that the object size is fixed, hence they cannot handle scale changes
Occlusion	No(mostly)	Yes	No	TLD can recover from short term partial occlusion.
Tracking Resumption	No	Yes	No	TLD searches through entire image and hence relocates the object while CT has limited detection window
Background Clutters	Yes	No	No	No. of training samples are less in both TLD and CT as compared to KCF
Non rigid deformation	Yes	No	Yes	KCF and CT depends on the immediate previous samples where as TLD relies on object model.
Fast motion	Yes	No	Yes	The tunable parameters in KCF and CT handles fast motion whereas in TLD tracker fails if motion is beyond ten pixels
Out of plane rotation	Yes	No	Yes	Object model in TLD highly depends on pose of the object
Precision	very high	moderate	Less	KCF outperforms TLD in terms of precision values
Processing Speed	very high	moderate	moderate	Instead of treating training and testing data as individual patches KCF formulates the whole data as circulant. This idea made it computationally efficient

5.4 Proposed Method

Our proposed method considers the outputs of the two state of the art tracking algorithms and selects the best one by calculating the conservative similarity of the patches. The two algorithms functions independently and provides their respective outputs, since both are complementary to each other, the inabilities or drifts in one can be compensated by other. The combining algorithm is shown in Algorithm 1.

The key to robust working of the proposed algorithm is efficient integration of output of cascaded detector of TLD and KCF to provide final output. If the target bbox is at nearby location of previous bbox then the preference will be given to tracker by comparing the similarity measures of TLD tracker and KCF tracker outputted patches with the object model patches, otherwise preference will be given to Detector. Overlap between two bboxes bbox1 and bbox2, represented as O($bbox_1$; $bbox_2$) is measured as same as in TLD, ratio of intersection area to union area of two bboxes.

1. **Initialization**: A scanning grid is generated for TLD by scaling and shifting input bbox with minimal bbox size = 20 pixels. Pixel comparisons are also generated randomly. Positive examples are generated by warping 10 of the closest patches to input bbox using geometric transformations. Negative patches are collected from background, no synthetic negative examples are generated. KCF considers the target patch as a positive sample. All the other patches that are in the search area are considered as negative samples. These are modelled as cyclic shifts of the target patch.For training regression approach is used. The training parameter $model\alpha$, the feature vector $modelx$ are initialised to zero.

2. **Input**: Current video frame

3. **Tracking and Detection**: Tracking by KCF is done by calculating the regression function response for several candidate patches and outputs the patch (kbb) whose response is closer to the response of the initial target patch. TLD uses its cascaded detector to classify the patches from scanning grid and isolates atmost 100 positive patches which are then clustered. This clustered bboxes mark the TLD detector response(dbb_i). The median flow tracker of TLD also outputs a bbox which is marked as TLD tracker response tbb.

Algorithm 1 Integration of output of KCF and TLD

1: **if** tracker is valid **then**

$Compare \quad S^c_{tbb} \quad and \quad S^c_{kbb}$

2: **if** $S^c_{tbb} > S^c_{kbb}$ **then**

$cbb = tbb$

3: **else**

$cbb = kbb$

4: **end if**

5: **if** $O(cbb, dbb_i) > 0.7$ && $(S^r_{dbb_i} > 0.6)$ **then**

$\begin{bmatrix} dbb.x & dbb.y & dbb.w & dbb.h \end{bmatrix}$ += $\begin{bmatrix} dbb_i.x & dbb_i.y & dbb_i.w & dbb_i.h \end{bmatrix}$

$fbb = mean(10 * cbb, dbb)$

6: **end if**

7: **else if** $max(S^c_{dbb_i}) > 0.65$ && $O(tbb, dbb_i) < 0.4$ **then**

$fbb = dbb_i$

8: **else**

Print Object is lost

9: **end if**

4. **Integration**: Referring to Algorithm 1, if tracker in TLD is defined i.e. FB error [16] is less than 10 pixels then tracker is valid. Then compare S^c_{tbb} and S^c_{kbb} where S^c is (Conservative Similarity: measure of confidence that the patch resembles appearance observed in the first 50 % of the positive patches which are stored in object model.) and tbb, kbb are output bbox of TLD and KCF trackers. The bbox which has higher S^c is considered as cbb. dbb_i, output of TLD satisfying overlap $O(cbb; dbbi) > 0.7$ and S^r (relative similarity: measure of confidence that the patch depicts the object) greater than a particular threshold are added coordinate wise and the final output fbb is weighted mean over each spatial dimension of summed dbb and cbb (line 5). Else if tracker is not valid we find out the most confident patch which is far from tracker (O(tbb; dbb) is less) and if its S^c is greater than threshold (line 7), it is selected as final output. Otherwise the object is considered lost (line 8).

5. **Learning and updation**: TLD generates 100 positive and negative examples (same as in step 1) and learns them by training the detector on them. Only those

KCF patches whose S^c is higher have been updated into the object model.

Flow chart of the proposed algorithm

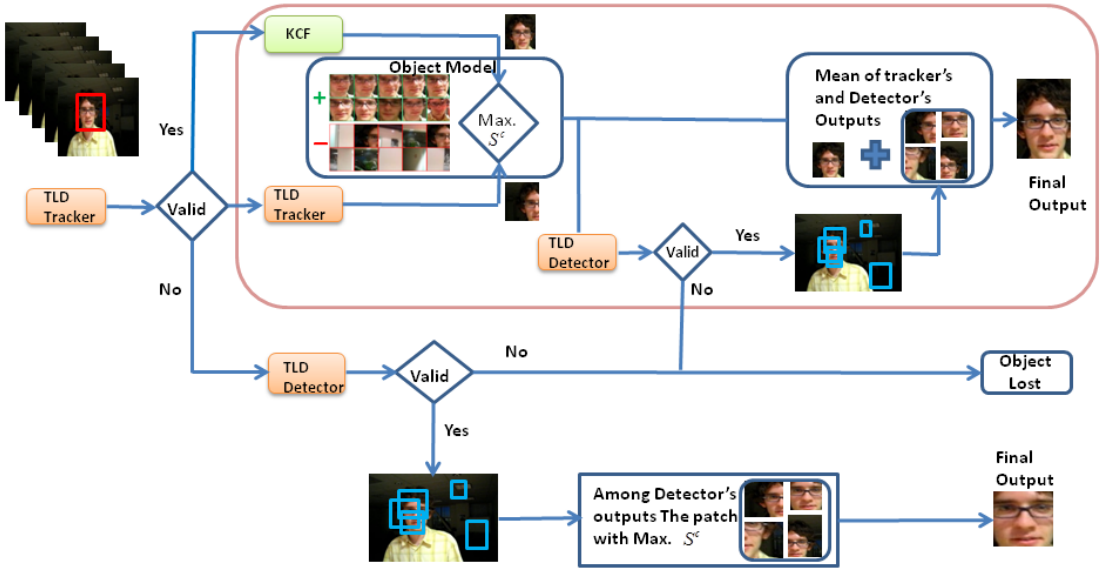


Figure 5.1: Flow chart of the proposed algorithm

CHAPTER 6

Experiments, Results and Discussion

In general, the trackers in literature [1, 2, 5, 4] are being evaluated using any one of the benchmark data sets available by considering only few parameters like in the case of KCF, It is evaluated on the bench mark data set by considering only precision plots , whereas success plots are more promising than the earlier. Unlike conventional way of evaluation methods, we have carried out validation of our proposed method using two state of the art tracking bench mark datasets i.e. ALOV300++ [3] and Visual object tracking benchmark data set[17] which are completely diverse in terms of data and evaluation methodology.

The ALOV300++ data set uses F-score as evaluation metric whereas the Benchmark dataset uses both precision and Success plots for Spatial Robustness Evaluation (SRE), Temporal Robustness Evaluation (TRE), One Pass Evaluation (OPE). From the results it is clearly evident that the proposed method outperforming the state of the art trackers both in terms of robustness and success rate.

6.0.1 Evaluation on ALOV300++ dataset

Description of the dataset

Amsterdam Library of Ordinary Videos for tracking, ALOV300++ [3] dataset has covered 14 circumstances in tracking such as illuminations, transparency, specularity, confusion with similar objects, clutter, occlusion, zoom, severe shape changes, different motion patterns, low contrast, and so on. Each group has minimum of 10 sequences and the total number of frames are 89364. The evaluation metric used for this dataset is F-score.

Evaluation metric

F-Score is used as the evaluation metric. It can be interpreted as a harmonic mean of the precision and recall. It reaches its best value at 1 and worst at 0. The mathematical

formula for F-Score is given by

$$F - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.1}$$

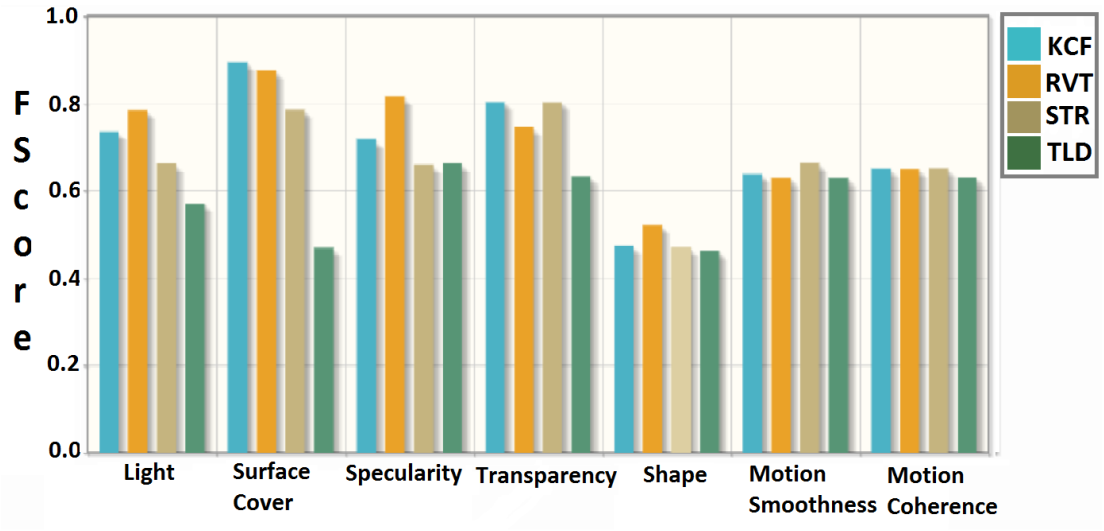


Figure 6.1: F-Score comparison chart

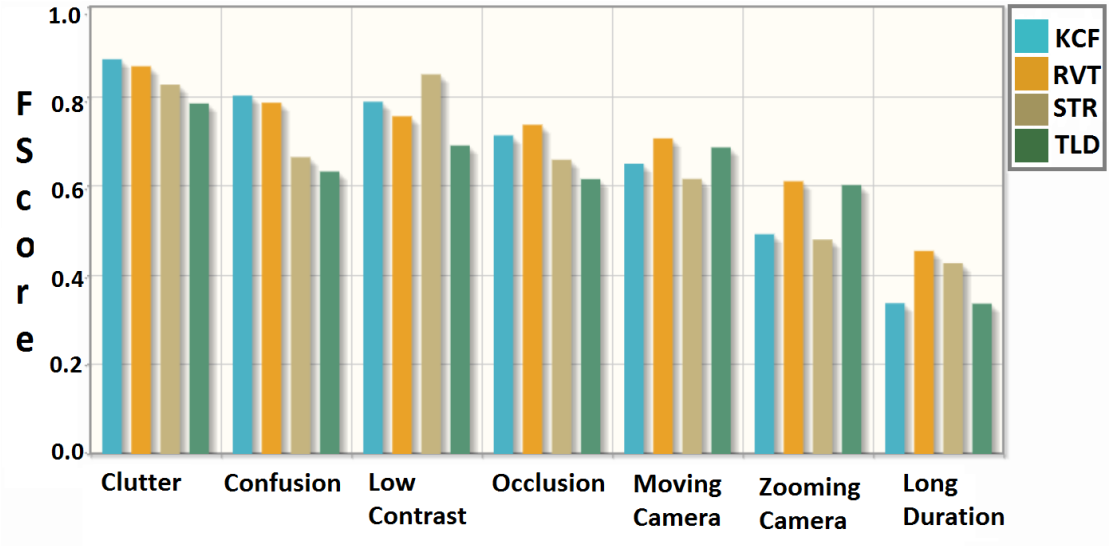


Figure 6.2: F-Score comparison chart

Let the short hand notation for precision, recall be P and R respectively. Precision can be defined as the ratio of the True Positive tracks (TP) and the total number of detected tracks (OD). Hence $P = \frac{TP}{OD}$. Recall can be defined as the ratio of true positive tracks(TP) and the number of ground truth data GT (i.e.True Positive + False Negative). Hence $R = \frac{TP}{GT}$. F-score for each video in the dataset has been calculated. Average value of F-score for sequences of each attribute is calculated and bar graphs are gener-

ated as shown in Fig. 6.1 & 6.2.

The following figures show the barcharts for all the 14 type sequences. The trackers under this evaluation are our proposed method Robust visual tracker (RVT), KCF, TLD, and STR [4].

Analysis of F score results :

Table 6.1: **F-score analysis of RVT, KCF, STR and TLD** ; scores in **bold** font shows the best one and *Italic* font shows the second best

Attribute	RVT	KCF	STR	TLD	Remarks about RVT
01-Light	**0.79**	*0.71*	0.67	0.57	The Best in handling illumination changes
02-Surface cover	*0.87*	**0.89**	0.75	0.51	Second best in handling drastic changes in background
03-Specularity	**0.83**	*0.71*	0.65	0.67	The best in handling reflections and specularities.
04-Transparency	0.77	**0.81**	**0.81**	0.65	Performs inferior to KCF and STR
05-Shape	**0.55**	*0.5*	0.49	0.47	The best in handling Non rigid deformations
06-Motion Smoothness	0.67	*0.68*	**0.69**	0.67	Performs inferior to KCF and STR
07-Motion Coherence	**0.67**	**0.67**	**0.67**	0.65	Stands among the top
08-Clutter	*0.85*	**0.87**	0.81	0.79	Second best in handling background clutter because of confusion in detection
09-Confusion	*0.78*	**0.79**	0.65	0.62	Second best in handling background clutter because of confusion in detection
10-Low contrast	0.75	*0.79*	**0.85**	0.68	Because of structure discriminative ability STR outperforms
11-Occlusion	**0.75**	*0.73*	0.65	0.62	The best in handling Occlusions
12-Moving Camera	**0.72**	0.65	0.61	*0.69*	The best in handling abrupt and global motions
13-Zooming Camera	**0.61**	0.45	0.43	*0.60*	The best in handling scale changes
14-Long Duration	**0.51**	0.35	*0.49*	0.35	The best tracker for Long term tracking
Overall Score	**0.72**	*0.68*	0.65	0.61	Overall the best tracker

6.0.2 Analysis of trackers performance :

The data shown in Table 4.1 shows that our proposed method is outperforming the other state of the art trackers in 8 out of 14 challenges and stands second best in 4 challenges.

In the video sequences of light, specularity, shape, moving camera, zooming camera, occulusions and long duration. RVT is better than KCF, TLD, STR. It means that proposed tracker can handle scale changes, illumination variations, occulusions, motion

blur, tracking resumption and changes in shape and sepecularity better than any other tracker in the literature.

Survivor Curve

As shown in Fig.?? from Visual object tracking :An experimental survey [3], STR [4] is the best tracker in overall performance though it not the best one in each challenge. The upper right shaded portion denotes the too-hard-track-yet videos and the lower shaded area gives the videos for which all trackers can perform well. This portion of the ALOV++ dataset is less than 7%, whereas the number of videos too hard to track by any tracker is 10%.

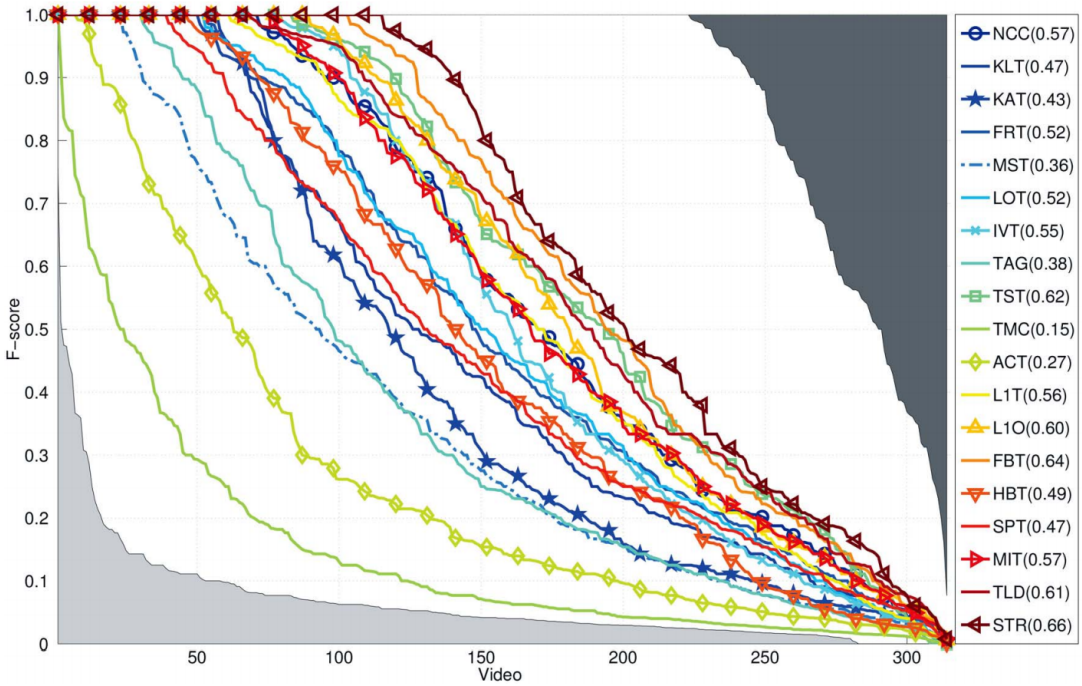


Figure 6.3: Survivor Curves : Comparison of state of the art trackers [3]

We have compared F-scores for all 314 videos and plotted survival curves which are plots of F-score verses number of videos. (x, y) on a survivor curve represents x number of videos in the dataset having a F-score value greater than y. The Fig. ?? shows the survivor curves of the proposed algorithm (RVT) and other prominent trackers KCF, TLD, STR, FBT, NCC and L1O

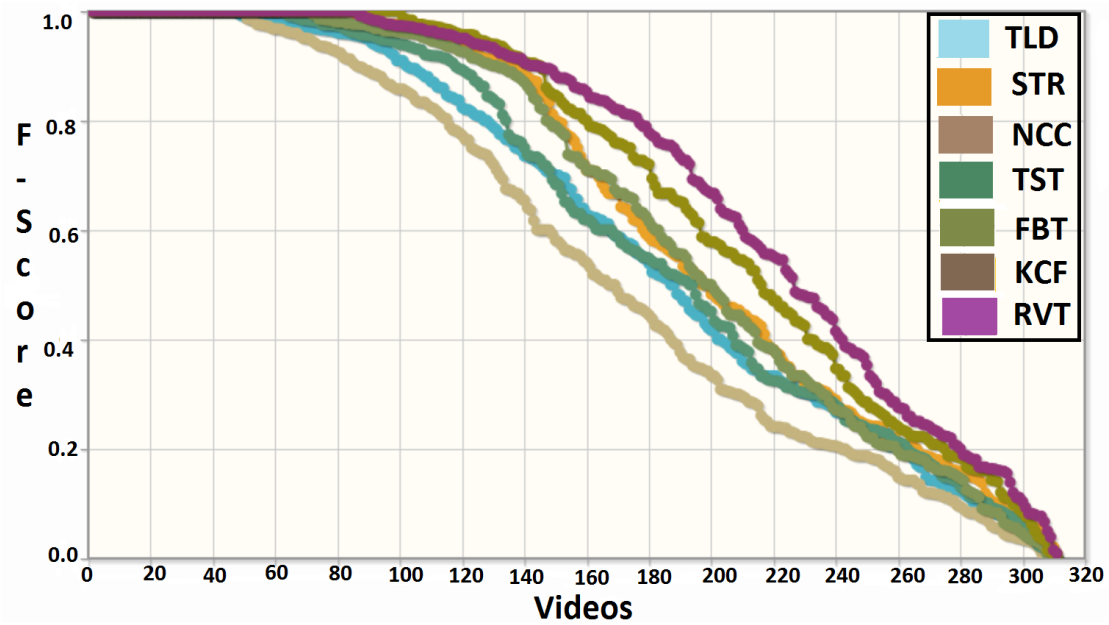


Figure 6.4: Survivor Curves : F-score Comparison of RVT with state of the art trackers

From the above plot it is evident that the integrated algorithm is clearly outperforming not only KCF and TLD trackers but also STR tracker which is mentioned the best tracker in the recent surveys [17, 3].

6.1 Evaluation on Benchmark Dataset

Description of dataset

The benchmark dataset contains 50 sequences with ground truth annotation. Each row in the ground-truth files represents the bounding box of the target in that frame, (x, y, box-width, box-height). These sequences can be categorized by annotating them with the 11 challenging attributes i.e, Illumination Variation(25), Scale Variation(28), Occlusion(29), Deformation(19), Motion Blur(12), Fast Motion(17), In-Plane Rotation(31), Out-of-Plane Rotation(39), Out-of-View(6), Background Clutters(21), Low Resolution(4). The value in the brackets besides each attribute shows the number of sequences annotated with that attribute. One sequence is often annotated with several attributes but no sequence has all the attributes. Some attributes occur more frequently, e.g., out-of-plane rotation and in-plane rotation, than others.

Evaluation Methodology : Quantitative analysis

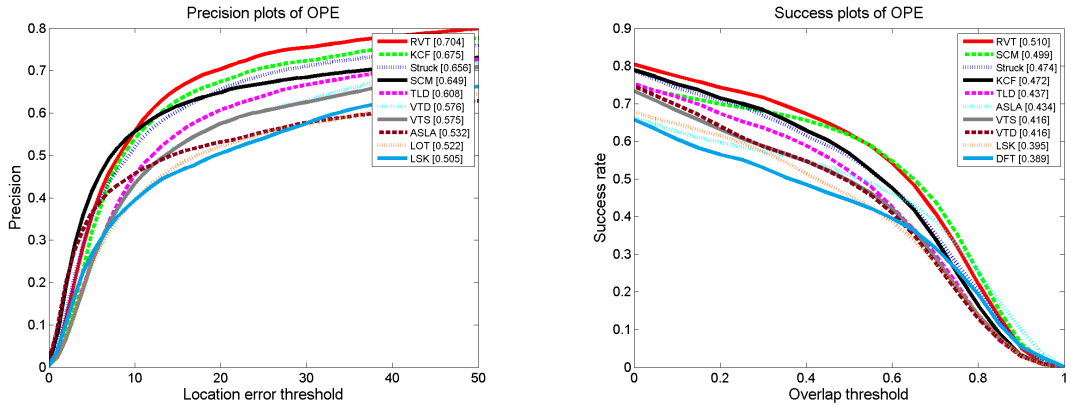
Precision and success rates are used as tools for evaluation of all the trackers in One pass evaluation (OPE). To validate robustness of the trackers in addition to OPE, Spatial Robustness Evaluation (SRE) where the initial bounding box is shifted in all directions and Temporal Robustness Evaluation (TRE) where initialization of bounding box can happen at any frame in the sequence but not in the first frame, are also used. Our proposed method RVT, KCF, TLD and STR trackers are evaluated.

Precision plots

Precision is the center location error, which is defined as the average Euclidean distance between the center locations of the tracked targets and the manually labelled ground truths. Then the average center location error over all the frames of one sequence is used to summarize the overall performance for that sequence. Lets denote this P_seq i.e., precision value for that sequence. precision values shows the percentage of frames whose estimated location is within the given threshold (20 pixels) distance of the ground truth. Final precision value for each attribute can be obtained by taking the P_seq values of all the sequences which are annotated with that attribute and then average it.

Success plots

Another evaluation metric is the bounding box overlap. Given the tracked bounding box



(a) Precision plot of OPE (b) Success plot of OPE

Figure 6.5: Analysis of OPE

r_t and the ground truth bounding box r_a, the overlap score is defined as $S = \frac{|\, r_t \cap r_a \,|}{|\, r_t \cup r_a \,|}$ where,$\cap$, $\cup$ represent the intersection and union of two regions, respectively, and $|\ .\ |$ denotes the number of pixels in the region. To measure the performance on a sequence of frames, we count the number of successful frames whose overlap S is larger than the given threshold. The success plot shows the ratios of successful frames at the thresholds varied from 0 to 1. Using one success rate value at a specific threshold (e.g. t=0.5) for tracker evaluation may not be fair or representative. Instead we use the area under curve (AUC) of each success plot to rank the tracking algorithms. The overall success rate and success rate for each attribute is found in a similar way as we find precision values.

6.1.1 One Pass Evaluation (OPE)

The conventional way of evaluating trackers is to run them throughout a test sequence with initialization from the ground truth position in the first frame and find the average success rate or precision values. This method is referred as one-pass evaluation (OPE). The figure [6.5] shows the overall precision and success plots of OPE and second column in Table 6.2 & 6.3 shows the quantitative analysis of precision values for each attribute.From the plots and tables data it is evident that the overall performance of RVT is better in both precision and success rates.

Table 6.2: Precision Values : The highest values are shown in bold

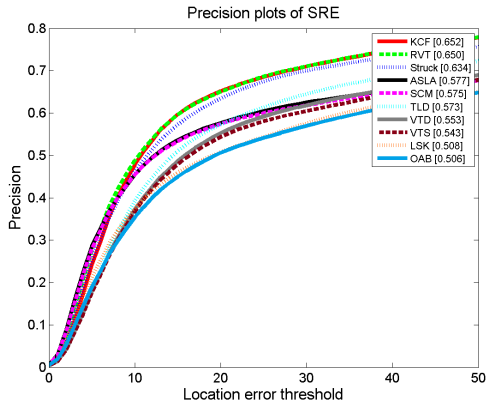
	OPE				SRE				TRE			
Attribute	RVT	KCF	TLD	STR	RVT	KCF	TLD	STR	RVT	KCF	TLD	STR
Abrupt Motion	0.56	0.5	0.55	**0.6**	0.51	0.49	0.49	**0.57**	0.56	0.52	0.48	**0.58**
Background Clutter	0.67	**0.69**	0.4	0.58	**0.66**	0.65	0.4	0.55	**0.72**	0.71	0.4	0.62
MotionBlurr	**0.58**	0.51	0.51	0.55	0.46	0.51	0.52	**0.58**	0.6	0.54	0.49	**0.61**
Deformation	**0.71**	0.65	0.51	0.52	0.6	**0.61**	0.5	0.54	**0.75**	0.7	0.57	0.65
IlluminationVariations	**0.65**	0.63	0.53	0.55	0.54	0.58	0.48	**0.55**	**0.76**	0.74	0.4	0.7
Inplane Rotation	**0.68**	0.65	0.58	0.61	0.62	**0.65**	0.55	0.59	**0.72**	0.7	0.56	0.65
Low Resolution	0.48	0.37	0.34	**0.54**	**0.48**	0.39	0.36	0.5	0.54	0.5	0.37	**0.62**
Occlusion Overlap	**0.7**	0.64	0.58	0.6	0.6	**0.61**	0.51	0.56	**0.74**	0.71	0.57	0.63
OutofPlaneRotation	**0.69**	0.64	0.59	0.59	0.62	**0.63**	0.56	0.59	**0.73**	0.71	0.597	0.66
OutofView	0.55	0.54	**0.57**	0.53	**0.5**	0.49	0.46	0.45	**0.55**	**0.55**	0.49	0.48
ScaleVariations	0.63	0.63	0.6	**0.64**	0.6	**0.62**	0.4	0.6	**0.71**	0.7	0.6	0.65
Overall QualityPlot	**0.7**	0.67	0.6	0.65	**0.65**	0.65	0.57	0.63	**0.76**	0.74	0.62	0.7

6.1.2 Spatial Robustness Evaluation (SRE)

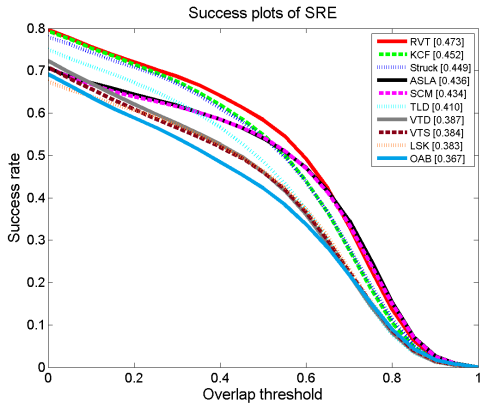
Since tracker is sensitive to initialisation of bounding box this type of evaluation is considered. In SRE the initial bounding box in the first frame is sampled by shifting or scaling the ground truth. Here, 8 spatial shifts including 4 corner shifts, 4 center shifts and 4 scale variations (supplement) are used. The amount for shift is 10 % of target size, and the scale ratio varies among 0.8, 0.9, 1.1 and 1.2 to the ground truth. Hence each tracker is evaluated 12 times for SRE.

The figure [6.6] shows the overall precision and success plots of OPE and third column in Table 6.2 & 6.3 shows the quantitative analysis of precision values for each attribute.

Precision values shows that RVT has no improved performance over KCF. But success rates show that RVT has the best performance. Since success rate is better performance measure we can say that RVT is spatially robust compared to the all other trackers.



(a) Precision plot of SRE



(b) Success plot of SRE

Figure 6.6: Analysis of SRE

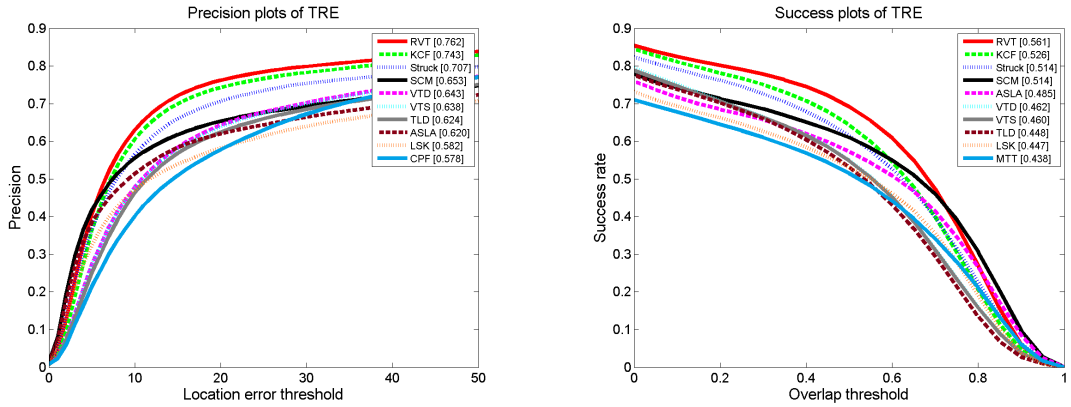
Table 6.3: Success Rate : The highest values are shown in bold

	OPE				SRE				TRE			
Attribute	RVT	KCF	TLD	STR	RVT	KCF	TLD	STR	RVT	KCF	TLD	STR
Abrupt Motion	0.44	0.385	0.417	**0.462**	0.411	0.382	0.394	**0.46**	0.455	0.411	0.392	**0.464**
Background Clutter	0.479	**0.498**	0.458	0.35	**0.473**	0.468	0.305	0.42	**0.532**	0.518	0.38	0.478
MotionBlurr	**0.452**	0.395	0.404	0.433	0.382	0.392	0.4	**0.495**	0.475	0.43	0.388	**0.485**
Deformation	**0.516**	0.47	0.378	0.393	**0.453**	0.445	0.365	0.407	**0.57**	0.532	0.404	0.51
IlluminationVariations	**0.467**	0.441	0.399	0.428	0.405	**0.412**	0.358	0.407	**0.514**	0.485	0.402	0.486
Inplane Rotation	**0.494**	0.449	0.416	0.444	**0.458**	0.449	0.387	0.418	**0.534**	0.498	0.407	0.473
Low Resolution	**0.374**	0.308	0.309	0.372	0.352	0.31	0.309	**0.362**	0.413	0.384	0.301	**0.456**
Occlusion Overlap	**0.515**	0.455	0.402	0.413	**0.447**	0.43	0.392	0.414	**0.547**	0.509	0.462	0.426
OutofPlaneRotation	**0.501**	0.445	0.42	0.432	**0.453**	0.429	0.395	0.418	**0.539**	0.501	0.425	0.477
OutofView	**0.478**	0.455	0.457	0.459	**0.44**	0.432	0.422	0.433	**0.483**	0.471	0.434	0.417
ScaleVariations	**0.446**	0.405	0.421	0.425	**0.423**	0.397	0.391	0.4	**0.501**	0.467	0.418	0.446
Overall QualityPlot	**0.51**	0.472	0.437	0.474	**0.473**	0.452	0.41	0.449	**0.561**	0.526	0.448	0.514

6.1.3 Temporal Robustness Evaluation(TRE))

Each tracker is initialized at some random frame with the ground-truth bounding box of target at that frame and runs to the end of the sequence, i.e., one segment of the complete sequence. The tracker is evaluated on each segment, and the overall measures are compared. 20 such segments are chosen for each sequence. By using this evaluation technique we can check for temporal robustness.

The figure [6.7] shows the overall precision and success plots of OPE and fourth column in Table 6.2 & 6.3 shows the quantitative analysis of precision values for each attribute. From the plots and table data, it is evident that the proposed algorithm is the best in terms of temporal robustness evaluation.



(a) Precision plot of OPE (b) Success plot of OPE

Figure 6.7: Analysis of TRE

6.2 Qualitative Results :

In this section we have presented qualitative evaluation of our proposed method on some of the bench mark videos. We have shown bounding boxes of three tracking methods RVT, KCF and TLD on some selected frames of the video sequence.
In the following pictures

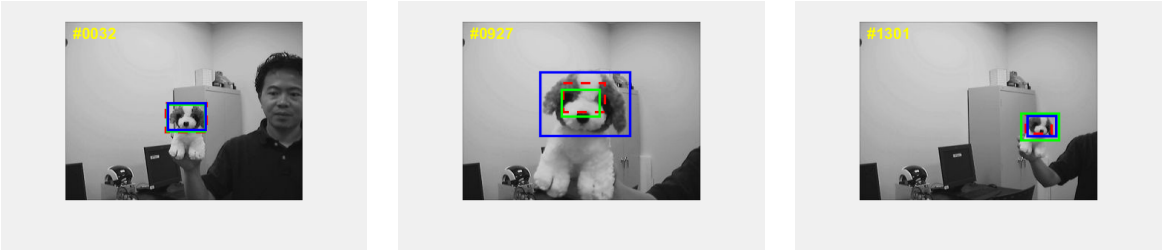
- **RVT** : Blue bbox
- **KCF** : Green bbox
- **TLD** : Red dashed bbox

Scale Changes: Dog Sequence

In the initial frames all the three trackers give approximately the accurate bounding box as shown in Fig.6.8. In later frames where the camera has zoomed-in only RVT is able to adapt for the scale variations better than TLD and KCF. This is because the efficient representation of the object model with the help of previously detected bboxes. have further in later frames when the camera is zoomed-out RVT is able to shrink the bounding box where as KCF and TLD cannot.

Handling Occlusions : Jogging Sequence

In the initial frames the three trackers are able to track the object accurately as shown in Fig.6.9. When the object gets fully occluded, KCF has lost the track. With the help of detector component which searches entire frame independent of tracker, RVT and TLD are able to track the object.



(a) Frame 32 (b) Frame 927 (c) Frame 1301

Figure 6.8: **Scale Variations** : (a) All the three trackers at starting (b) RVT adapted for Zoomed-in, where as TLD and KCF cannot) RVT adapted for Zoomed-out.

Handling Tracking Resumption

In the first frame all three are able to track In the next frame the object has gone out of the frame. In the later frames the object comes into the frame RVT and TLD are able to track the object again.

(a) Frame 11 (b) Frame 91 (c) Frame 240

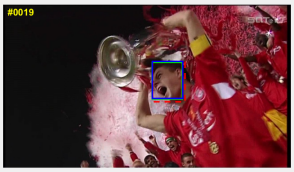
Figure 6.9: **Occlusion**: (a) All the three trackers at starting (b) object got occluded and KCF missed tracking (c) RVT, TLD continuing tracking.

Handling Background clutter and Motion blur : Soccer Sequence

As shown in Fig.6.10. during initial frames three trackers able to track accurately. Later in due to back ground clutter and motion blur TLD missed the track. Since more number of patches are taken into consideration while training the model in KCF and RVT, they have the ability to handle both background clutter and motion blur.

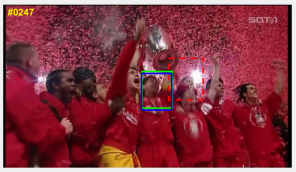
Handling Low resolution and Confusion : Walking Sequence

As shown in Fig.6.11. during initial frames three trackers able to track accurately. Later in due to low resolution and confusion the TLD missed the track KCF is not further tracking. Whereas RVT continued tracking by adapting the scale changes. Tracker component TLD failed due to occlusion and detector confused between similar objects,



(a) Frame 25

(b) Frame 246

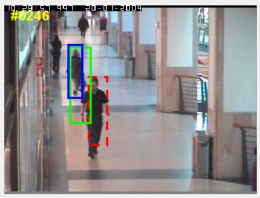


(c) Frame 291

Figure 6.10: **Clutter and Motion blur**: (a) All the three trackers at starting (b) TLD missed tracking (c) RVT and KCF continuous tracking.

But KCF can handle confusion with the help that RVT able to track and even adapted for scale changes also in low resolution.

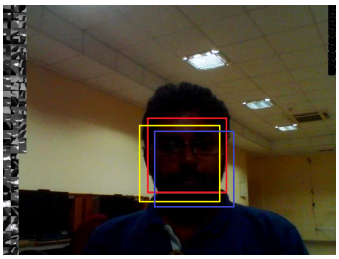
(a) Frame 25



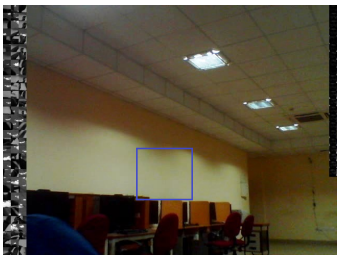
(b) Frame 246

(c) Frame 291

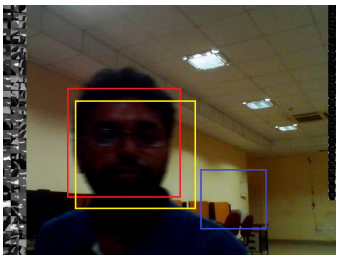
Figure 6.11: **Low resolution and Confusion**: (a) All the three trackers at starting (b) TLD confused the object and KCF not adapting to the object's scale (c) RVT continuous tracking.



(a) Frame 42



(b) Frame 47



(c) Frame 56

Figure 6.12: **Tracking resumption**: (a) All the three trackers at starting (b) object moves out of frame, KCF starts to track background (c) KCF loses object while other trackers resume tracking

CHAPTER 7

Conclusions and Future work

~~Conclusions~~

In this thesis, we proposed a simple yet robust tracking algorithm formed by intelligently integrating TLD and KCF. Both are state-of-the-art algorithms and perform well on most of the tracking challenges such as illumination change, object pose variation, short-term partial occlusion etc. But they still have areas in which work is left to do. We have come up with an algorithm which combines the two trackers efficiently to take advantages of both and complement each other to overcome their short-ends. Learning component of TLD helps the proposed algorithm in tracking resumption and detector helps in estimating the scale accurately while in KCF the negative samples are modelled as patches extracted at different translations of the target patch. It makes KCF more precise and accurate tracker. KCF is not adaptable to scale variations and it doesn't have the virtue of tracking resumption. Hence we provided an efficient algorithm to combine both. It has been observed that the proposed algorithm works better than KCF and TLD. It is able to handle scale variations, tracking resumption and its accuracy has been improved too.

A quantitative analysis of the proposed tracker has been done on the datasets Visual Tracker Benchmark and ALOV300++. Precision plots and success plots have been generated for all the videos in the dataset and they are averaged. Spatial robustness and temporal robustness of the proposed tracker has also been verified. It has been shown that the proposed algorithm out-performs most of the eminent trackers like KCF, TLD, STRuck.

~~Future work~~

The integrated algorithm can be made even more precise by improving the KCF performance. The features used in KCF are either raw pixels or Histogram of Oriented Gradients (HOG). The performance is high when HOG features are used. The more features we add, the more information about the appearance of the target is added. Hence KCF performance can be improved by using multiple features. For example, a

combination of gray scale feature, HOG feature, Local Binary pattern (LBP) feature can be used[18]. This combination could be of great use since gray scale feature is the basic description of the object, HOG has powerful detection capacity, LBP is good at describing the texture of the image.

The integrated algorithm can be extended for multiple object tracking. The idea is that perform tracking of each object in the frame parallely and device an algorithm to handle the interactions of the objects. It can also be extended to several practical applications like object follower, counting RBC in blood, counting bacteria cells etc.,. A real time surveillance system can also be designed with the help of this algorithm.

REFERENCES

[1] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1409–1422, 2012.

[2] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 864–877.

[3] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: an experimental survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 7, pp. 1442–1468, 2014.

[4] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 263–270.

[5] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, no. 3, pp. 583–596, 2015.

[6] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 2756–2759.

[7] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." in *IJCAI*, vol. 81, 1981, pp. 674–679.

[8] 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*. Ieee, 2007, pp. 1–8.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[10] R. M. Gray, "Toeplitz and circulant matrices: A review." DTIC Document, Tech. Rep., 1971.

[11] R. Rifkin, G. Yeo, and T. Poggio, "Regularized least-squares classification," *Nato Science Series Sub Series III Computer and Systems Sciences*, vol. 190, pp. 131–154, 2003.

[12] D. L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[13] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *Information Theory, IEEE Transactions on*, vol. 52, no. 12, pp. 5406–5425, 2006.

[14] D. Achlioptas, "Database-friendly random projections: Johnson-lindenstrauss with binary coins," *Journal of computer and System Sciences*, vol. 66, no. 4, pp. 671–687, 2003.

[15] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[16] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 2756–2759.

[17] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, no. 9, pp. 1834–1848, 2015.

[18] S. Zhang, X. Yu, Y. Sui, S. Zhao, and L. Zhang, "Object tracking with multi-view support vector machines," *Multimedia, IEEE Transactions on*, vol. 17, no. 3, pp. 265–278, 2015.

List of papers based on thesis