नेताजी सुभाष प्रौद्योगिकी संस्थान
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
(An Institution of Govt. of NCT of Delhi-Formerly, Delhi Institute of Technology)
Azad Hind Fauj Marg, Sector -3, Dwarka, New Delhi-110078
Tel. : 25099050, Fax : 25099025, Website : http://www.nsit.ac.in

B.E. PROJECT ON

Multiple Object Tracking

Submitted By:

Aditya Bakshi 2016UEC2178

Under the Guidance of
Prof. Jyotsna Singh

Project-II in partial fulfilment of requirement for the award of
B.E. in
Electronics & Communication Engineering

Division of Electronics & Communication Engineering
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
(UNIVERSITY OF DELHI)
NEW DELHI-110078
YEAR 2016-2020

CERTIFICATE

This is to certify that the report entitled **"Multiple Object Tracking"** being submitted by Aditya Bakshi to the Division of Electronics and CommunicationEngineering, NSIT, for the award of bachelor's degree of engineering, is the record ofthe bonafide work carried out by them under our supervision and guidance. The resultscontained in this report have not been submitted either in part or in full to any otheruniversity or institute for the award of any degree or diploma.

SUPERVISOR
Prof. Jyotsna Singh
ECE Division
NSIT, New Delhi, India

ACKNOWLEGEMENTS

We are highly grateful to the division of Electronics and Communication Engineering, Netaji Subhash Institute Of Technology (NSIT) for providing this opportunity to carry out the project work.

The constant guidance and encouragement received from our supervisor Prof. Jyotsna Singh, has been of great help in carrying our present work, literature and experimentation is greatly acknowledged with reverential thanks.

Finally, We would like to express gratitude to our friends and all the faculty members of Division of Electronics and communication Engineering, NSIT for their support throughout the course of this work.

Division of Electronics and communication Engineering

Netaji Subhash Institute of Technology (NSIT)

Azad Hind Fauj Marg

Sector-3, Dwarka, New Delhi

PIN- 110078

Plagiarism Report

Multiple Object Tracking

ORIGINALITY REPORT

%14	%5	%3	%11
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to SASTRA University Student Paper	%3
2	ml-cheatsheet.readthedocs.io Internet Source	%3
3	insis.vse.cz Internet Source	%1
4	Allebach, J, N Bose, E Delp, S Rajala, C Bouman, L Sibul, Wayne Wolf, and Ya-Qin Zhang. "Multidimensional Signal Processing", Electrical Engineering Handbook, 1997. Publication	%1
5	Submitted to Kaplan College Student Paper	%1
6	Submitted to UNITEC Institute of Technology Student Paper	%1
7	Submitted to Mentone Grammar Student Paper	%1
8	Submitted to Karunya University Student Paper	%1

ABSTRACT

We have implemented Multiple Object Tracking using Open CV.

We have used 7 different object tracking algorithms and compared their results. We can determine the best use cases for a particular We have implemented and compared the performance of 7 object tracking algorithms. Every algorithm has some issue when it comes to object tracking, so we can say that one algorithm fits all situations does not exist, but we can chose the most suitable algorithm for our work.

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Object Tracking

Video tracking is the process of locating a moving object (or multiple objects) over time using a camera. It has a variety of uses, some of which are: human-computer interaction, security and surveillance, video communication and compression, augmented reality, traffic control, medical imaging and video editing. Video tracking can be a time-consuming process due to the amount of data that is contained in video. Adding further to the complexity is the possible need to use object recognition techniques for tracking, a challenging problem in its own right.

An ideal object tracking algorithm will:

- Only require the object detection phase once (i.e., when the object is initially detected)
- Will be extremely fast — *much* faster than running the actual object detector itself
- Be able to handle when the tracked object "disappears" or moves outside the boundaries of the video frame
- Be robust to occlusion
- Be able to pick up objects it has "lost" in between frames

This is a tall order for any computer vision or image processing algorithm and there are a variety of tricks we can play to help improve our object trackers.

There are different kinds of object tracking algorithms :

1. **Dense Optical flow**: These algorithms help estimate the motion vector of every pixel in a video frame.
2. **Sparse optical flow**: These algorithms, like the Kanade-Lucas-Tomashi (KLT) feature tracker, track the location of a few feature points in an image.
3. **Kalman Filtering**: A very popular signal processing algorithm used to predict the location of a moving object based on prior motion information. One of the early applications of this algorithm was missile guidance! Also as mentioned here, "the on-board computer that guided the descent of the Apollo 11 lunar module to the moon had a Kalman filter".
4. **Meanshift and Camshift**: These are algorithms for locating the maxima of a density function. They are also used for tracking.
5. **Single object trackers**: In this class of trackers, the first frame is marked using a rectangle to indicate the location of the object we want to track. The object is then tracked in subsequent frames using the tracking algorithm. In most real life applications, these trackers are used in conjunction with an object detector.

6. **Multiple object track finding algorithms**: In cases when we have a fast object detector, it makes sense to detect multiple objects in each frame and then run a track finding algorithm that identifies which rectangle in one frame corresponds to a rectangle in the next frame.

1.2 Tracking vs Detection

1. **Tracking is faster than Detection**: Usually tracking algorithms are faster than detection algorithms. The reason is simple. When you are tracking an object that was detected in the previous frame, you know a lot about the appearance of the object. You also know the location in the previous frame and the direction and speed of its motion. So in the next frame, you can use all this information to predict the location of the object in the next frame and do a small search around the expected location of the object to accurately locate the object. A good tracking algorithm will use all information it has about the object up to that point while a detection algorithm always starts from scratch. Therefore, while designing an efficient system usually an object detection is run on every n^{th} frame while the tracking algorithm is employed in the n-1 frames in between. Why don't we simply detect the object in the first frame and track subsequently? It is true that tracking benefits from the extra information it has, but you can also lose track of an object when they go behind an obstacle for an extended period of time or if they move so fast that the tracking algorithm cannot catch up. It is also common for tracking algorithms to accumulate errors and the bounding box tracking the object slowly drifts away from the object it is tracking. To fix these problems with tracking algorithms, a detection algorithm is run every so often. Detection algorithms are trained on a large number of examples of the object. They, therefore, have more knowledge about the general class of the object. On the other hand, tracking algorithms know more about the specific instance of the class they are tracking.

2. **Tracking can help when detection fails**: If you are running a face detector on a video and the person's face gets occluded by an object, the face detector will most likely fail. A good tracking algorithm, on the other hand, will handle some level of occlusion.

3. **Tracking preserves identity**: The output of object detection is an array of rectangles that contain the object. However, there is no identity attached to the object. For example, in the video below, a detector that detects red dots will output rectangles corresponding to all the dots it has detected in a frame. In the next frame, it will output another array of rectangles. In the first frame, a particular dot might be represented by the rectangle at location 10 in the array and in the second frame, it could be at location 17. While using detection on a frame we have no idea which rectangle corresponds to which object. On the other hand, tracking provides a way to literally connect the dots.

1.3Work

Multiple object tracking, or **MOT**, is a versatile experimental paradigm developed by Zenon Pylyshyn for studying sustained visual attention in a dynamic environment in 1988. It was first developed in order to support visual indexing theory (FINST theory). MOT was then commonly used as an experimental technique in order to study how our visual system tracks multiple moving objects. Dozens or perhaps hundreds of modified MOT experiments have been conducted as a continuous attention-demanding task to further understanding human's visual and cognitive function.

Application:

- Traffic management system
- Crowd Surveillance
- Object tracking in sports

Work before midsem

We have implemented multiple object tracking using Open CV. It is giving good results and is able to track any number of objects we want. The only down side of it being is that it will be somewhat computationally intensive.

Work after midsem

We have implemented 7 different object tracking algorithms and compared their results. We can determine the best use cases for a particular We have implemented and compared the performance of 7 object tracking algorithms. Every algorithm has some issue when it comes to object tracking, so we can say that one algorithm fits all situations does not exist, but we can chose the most suitable algorithm for our work.

We have summarised the comparison of all the 7 algorithms in the conclusion section of this report later on.

1.4 Future Improvement

We have summarised the comparison of all the 7 algorithms in the conclusion section of this report later on.

In our project we have used single object tracking algorithms and modified them to tracking multiple objects. How this is achieved is via making algorithm object for each specimen to be tracked. This means if we are to track N objects than N separate instances of the algorithm will run making the whole program slow.

We can say that the performance of the program is inversely propational to number of objects to be tracked.

As a future improvement to our project we wish to make this process more efficient. This improvement will enable the program to run on less powerful hardware and also in cases of small microprocessors as in when integrated with drones or anything else.

There are many proposed ides and that we can follow to achieve our aforementioned goal. Some of them are:

- We can use GPU of the system to accelerate the processing of algorithm. Currently the program only uses the system CPU to process the information. With use of GPU we can greatly improve the efficiency of the code and make it run a lot faster.

- Make use of multithreading in the code. Each instance of the algorithm can be allocated a separate thread to process. A different operating system can be developed to particularly run this program of multiple object tracking which unlike the batch operating system will only dedicate its resources to manage smooth running of this program. This will reduce all overhead caused due to context switching and at the same time manage memory specifically for this program. It can take advantage multi core system hardware to run all the threads most efficiently as possible. The operating system will provide kernel level permissions to the program to fully take advantage of the hardware.

- We can improve the object tracking algorithm itself to cater to our goal of reducing processing power and making it efficient. Instead of running separate instance of algo.

CHAPTER 2

The Centroid Tracking Algorithm

It is one of the most basic object tracking algorithm. The centroid tracking algorithm is a multi-step process. We will review each of the tracking steps in this section.

Step #1: Accept bounding box coordinates and compute centroids

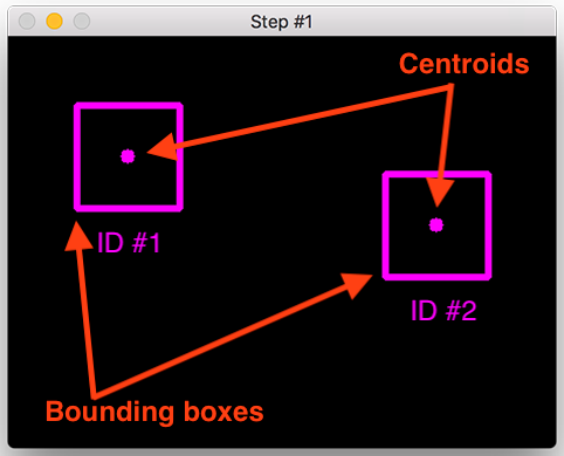


Figure 2.1: *To build a simple object tracking algorithm using centroid tracking, the first step is to accept bounding box coordinates from an object detector and use them to compute centroids.*

The centroid tracking algorithm assumes that we are passing in a set of bounding box*(x, y)*-coordinates for each detected object in ***every single frame***.

These bounding boxes can be produced by any type of object detector you would like (color thresholding + contour extraction, Haar cascades, HOG + Linear SVM, SSDs, Faster R-CNNs, etc.), provided that they are computed for every frame in the video.

Once we have the bounding box coordinates we must compute the "centroid", or more simply, *the center (x, y)-coordinates* of the bounding box. **Figure 1** above demonstrates accepting a set of bounding box coordinates and computing the centroid.
Since these are the first initial set of bounding boxes presented to our algorithm we will assign them unique IDs.

Step #2: Compute Euclidean distance between new bounding boxes and existing objects

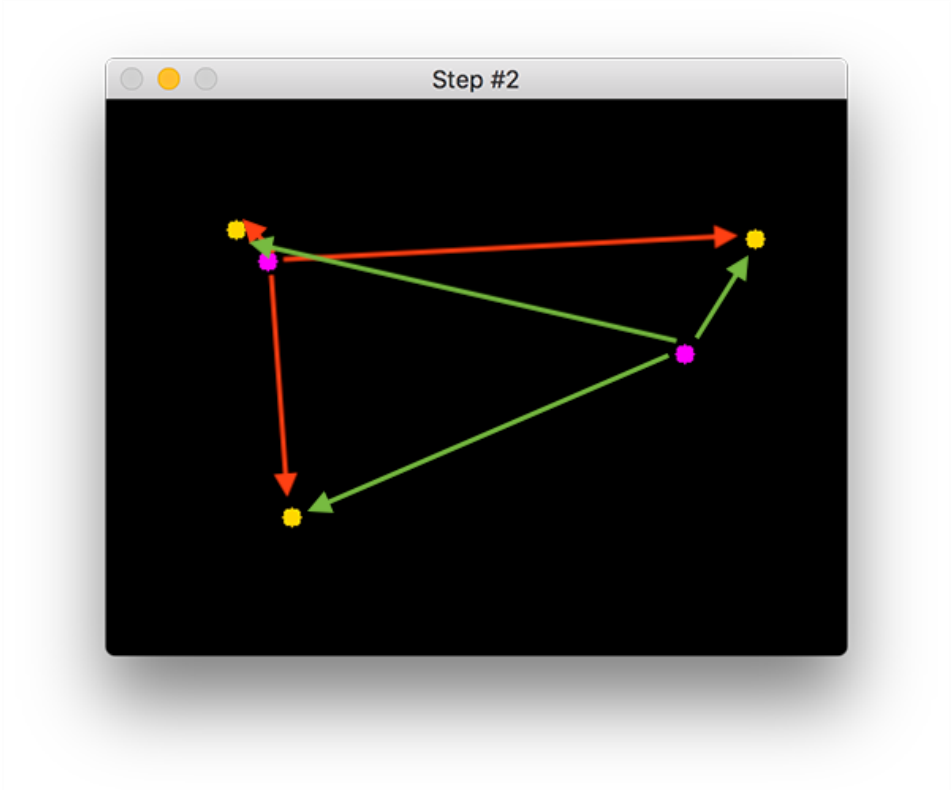


Figure 2.2: *Three objects are present in this image for simple object tracking with Python and OpenCV. We need to compute the Euclidean distances between each pair of original centroids (red) and new centroids (green).*

For every subsequent frame in our video stream we apply **Step #1** of computing object centroids; however, instead of assigning a new unique ID to each detected object (which would defeat the purpose of object tracking), we first need to determine if we can *associate* the *new* object centroids (yellow) with the *old* object centroids (purple). To accomplish this process, we compute the Euclidean distance (highlighted with green arrows) between each pair of existing object centroids and input object centroids.

From **Figure 2** you can see that we have this time detected three objects in our image. The two pairs that are close together are two existing objects.

We then compute the Euclidean distances between each pair of original centroids (yellow) and new centroids (purple). But how do we use the Euclidean distances between these points to actually match them and associate them?

The answer is in **Step #3**.

Step #3: Update *(x, y)*-coordinates of existing objects

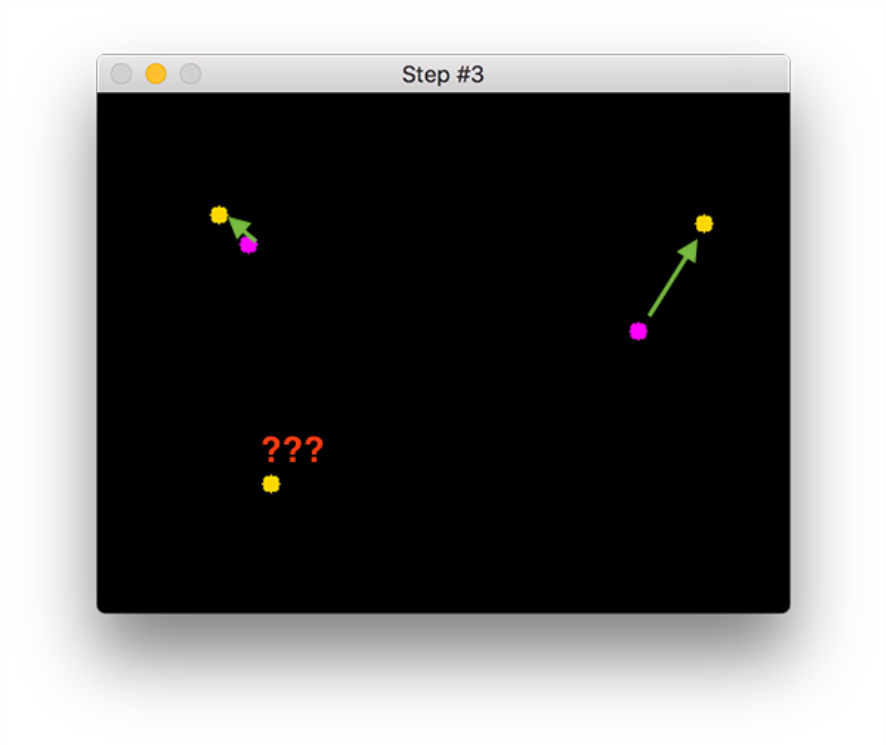


Figure 2.3: *Our simple centroid object tracking method has associated objects with minimized object distances.*

The primary assumption of the centroid tracking algorithm is that a given object will potentially move in between subsequent frames, but the *distance* between the centroids for frames F_t and F_{t+1} will be *smaller* than all other distances between objects.

Therefore, if we choose to associate centroids with minimum distances between subsequent frames we can build our object tracker.

In **Figure 3** you can see how our centroid tracker algorithm chooses to associate centroids that minimize their respective Euclidean distances.

But what about the lonely point in the bottom-left?
It didn't get associated with anything — what do we do with it?

Step #4: Register new objects

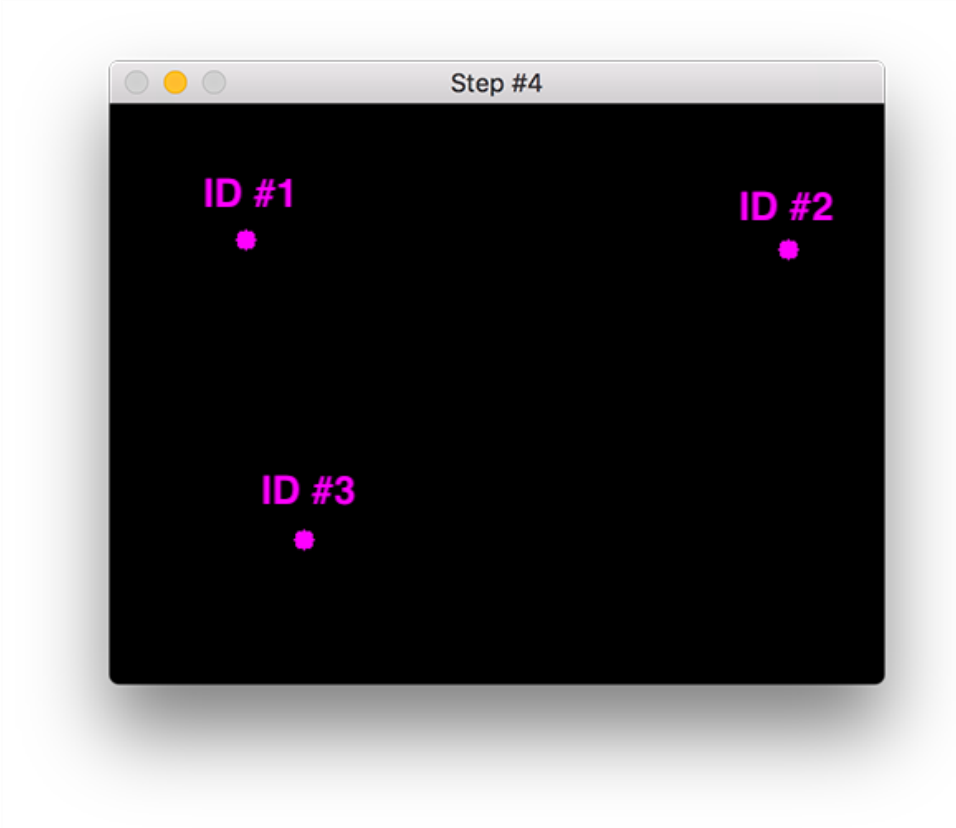


Figure 2.4: *In our object tracking with Python and OpenCV example, we have a new object that wasn't matched with an existing object, so it is registered as object ID #3.*

In the event that there are more input detections than existing objects being tracked, we need to register the new object. "Registering" simply means that we are adding the new object to our list of tracked objects by:

1. Assigning it a new object ID
2. Storing the centroid of the bounding box coordinates for that object

We can then go back to **Step #2** and repeat the pipeline of steps for every frame in our video stream.
Figure 4 demonstrates the process of using the minimum Euclidean distances to associate existing object IDs and then registering a new object.

Step #5: Deregister old objects

Any reasonable object tracking algorithm needs to be able to handle when an object has been lost, disappeared, or left the field of view.
Exactly how you handle these situations is really dependent on where your object tracker is meant to be deployed, but for this implementation, we will deregister old objects when they cannot be matched to any existing objects for a total of *N* subsequent frames.

CHAPTER 3

TRACKING ALGORITHMS

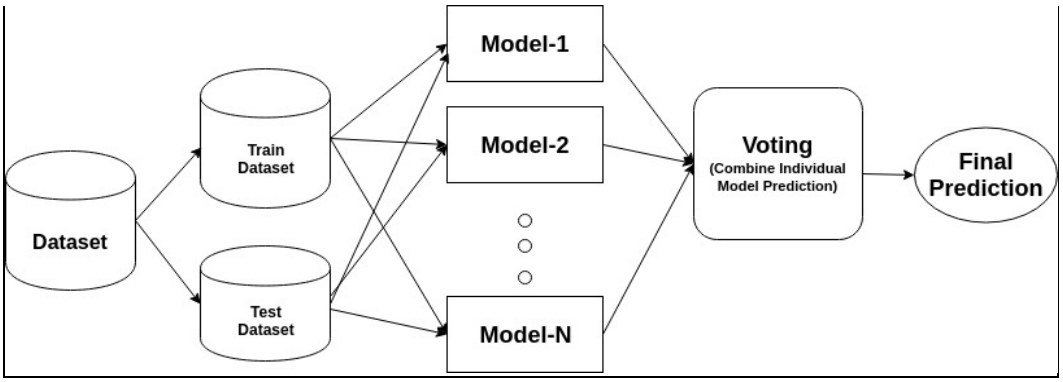
3.1 Boosting Tracker Algorithm

Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifier.

A weak learner is represented as a classifier, which is only slightly correlated with the true classification (it can label examples better than random guessing). In comparison to that a strong learner is elucidated as a classifier that is arbitrarily well-correlated with the true classification.

This tracker is based on an online version of AdaBoost — the algorithm that the HAAR cascade-based face detector uses internally. **AdaBoost**, short for *Adaptive Boosting*, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Gödel Prize for their work.

- It was the first really successful boosting algorithm developed for binary classification.
- It is the best starting point for understanding boosting.
- It can be used in conjunction with many other types of learning algorithms to improve performance.
- The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier.
- AdaBoost is best used to boost the performance of decision trees on binary classification problems.
- AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favour of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers.

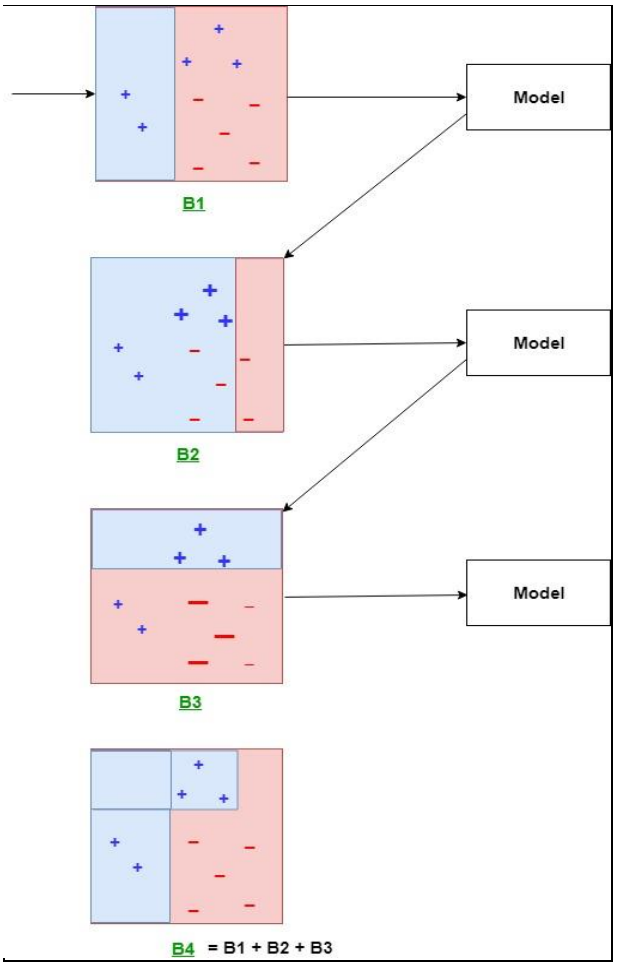


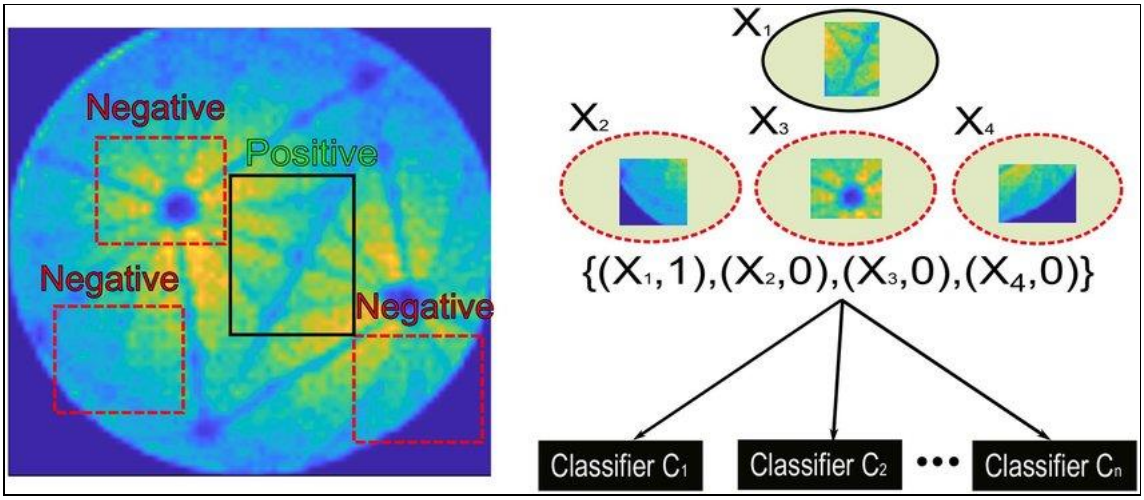
ADABOOST CLASSIFIER

Working of the tracker:

This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.

- This classifier needs to be trained at runtime with positive and negative examples of the object.
- The initial bounding box supplied by the user (or by another object detection algorithm) is taken as the positive example for the object, and many image patches outside the bounding box are treated as the background.
- Given a new frame, the classifier is run on every pixel in the neighbourhood of the previous location and the score of the classifier is recorded.
- The new location of the object is the one with maximum score.
- So now we have one more positive example for the classifier. As more frames come in, the classifier is updated with this additional data.





Advantages:

- A crucial property of using AdaBoost is that it almost never overfits the data no matter how many iterations it is run.
- This algorithm is a decade old and works ok.
- Interestingly, mostly used for legacy reasons and comparing other algorithms.

Disadvantages:

- Tracking performance is mediocre.
- It does not reliably know when tracking has failed.
- This tracker is slow and doesn't work very well.

3.2 MIL Tracker Algorithm

MIL stands for ***Multiple Instance Learning.***

It is a form of weakly supervised learning.

Training instances are arranged in sets, called bags. A label is provided for entire bags but not for instances.

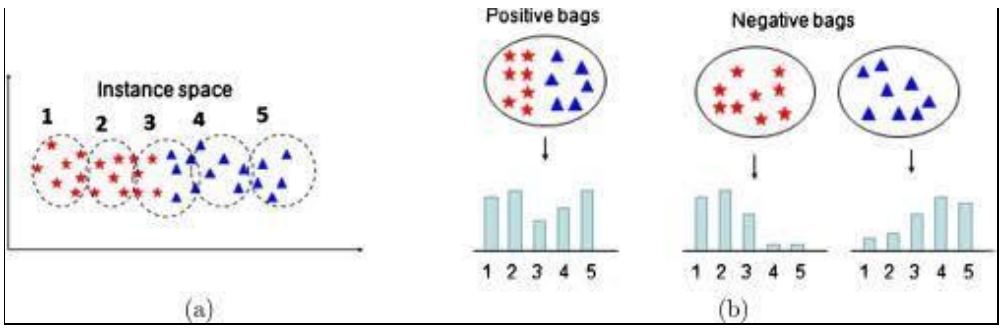
- Negative bags do not contain positive instances.
- Positive bags may contain negative and positive instances.
- Positive bags contain at least one positive instance.

It deals with **weakly annotated data**. This reduces the annotation cost.

Algorithms can now learn from a greater quantity of training data.

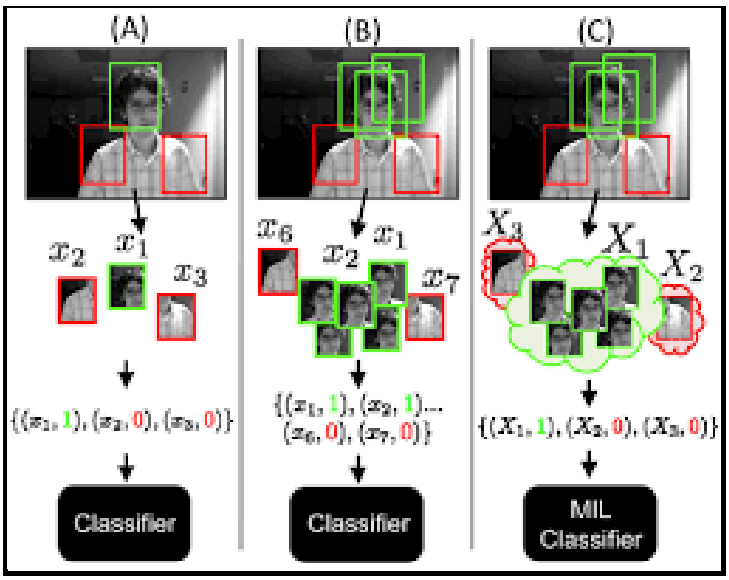
This tracker is similar in idea to the BOOSTING tracker described above.

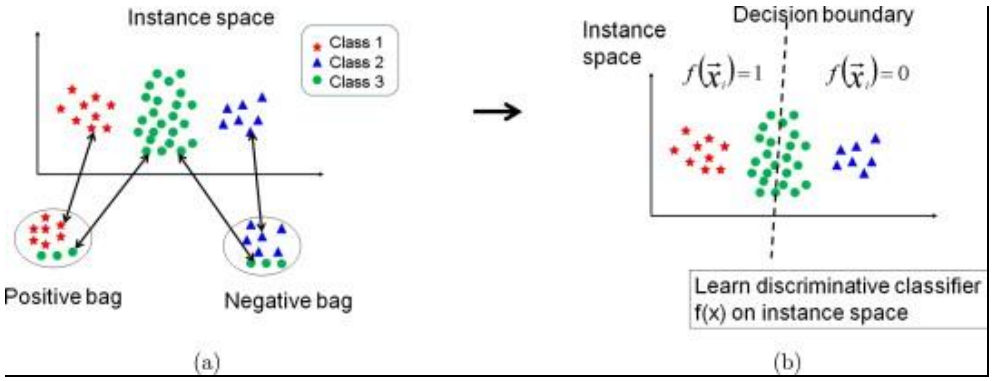
The big difference is that instead of considering only the current location of the object as a positive example, it looks in a small neighbourhood around the current location to generate several potential positive examples. It can be thought of as a bad idea because in most of these "positive" examples the object is not centered. This is where Multiple Instance Learning (MIL) comes to rescue.



Working of MIL tracker:

- In MIL, you do not specify positive and negative examples, but positive and negative "bags".
- The collection of images in the positive bag is not all positive examples.
- Instead, only one image in the positive bag needs to be a positive example. In an example, a positive bag contains the patch centered on the current location of the object and also patches in a small neighbourhood around it.
- Even if the current location of the tracked object is not accurate, when samples from the neighbourhood of the current location are put in the positive bag, there is a good chance that this bag contains at least one image in which the object is nicely centred.
- Then the learner is allowed some flexibility in finding a decision boundary.
- Further, it allows us to update the appearance model with a set of image patches, even though it is not known which image patch precisely captures the object of interest.
- This leads to more robust tracking results with fewer parameter tweaks.





Advantages:

- The performance is pretty good.
- It does not drift as much as the BOOSTING tracker and it does a reasonable job under partial occlusion.
- Algorithm is simple to implement, and can run at real-time speeds.
- If using OpenCV 3.0, this might be the best tracker available.
- Better accuracy than BOOSTING tracker.

Disadvantages:

- Tracking failure is not reported reliably.
- Does not recover from full occlusion.
- Faces difficulty when data points are grouped in sets.

3.3 TLD Tracker Algorithm

TLD Tracker Algorithm TLD stands for **Tracking, learning and detection.**

As the name suggests, this tracker decomposes the long term tracking task into three components — (short term) tracking, learning, and detection.

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.

Working of TLD Tracker –

● 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. This output of this tracker tends to jump around a bit.

● For example, if you are tracking a pedestrian and there are other pedestrians in the scene, this tracker can sometimes temporarily track a different pedestrian than the one you intended to track.

● On the positive side, this track appears to track an object over a larger scale, motion, and occlusion. If you have a video sequence where the object is hidden behind another object, this tracker may be a good choice.

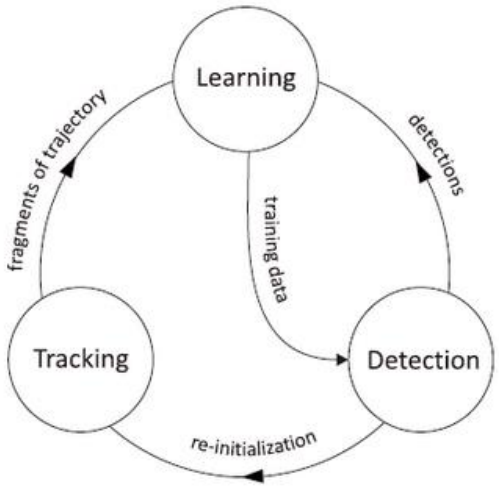


Figure 3.1: *Flowchart of TLD framework*

3.3.1 Tracking-Learning-Detection framework:

In TLD , tracking and detection are integrated along witha new learning component, called P-N learning.These 3components forms a strong feedback loop as shown in Figure. Tracker follows the object from one frame to the nextframe in the image sequence whereas the detector considers each frame individually and runs a sliding window on theframe to find the location of the object. Optical flow is usedfor feature correspondence during tracking and new appearances of the object found in the trajectory of the tracker arelearned by the detector. The task of learning is to estimateand update errors in the detector, to avoid these errors infuture. Error is estimated by a pair of experts, using the P-N learning method.Occlusion occurs in object tracking if the object beingtracked is hidden by another object and tracking fails .TLDovercomes loss of tracking due to occlusion, by online training of classifier. Tracker and detector will provide results inparallel, which will be provided to an integrator.

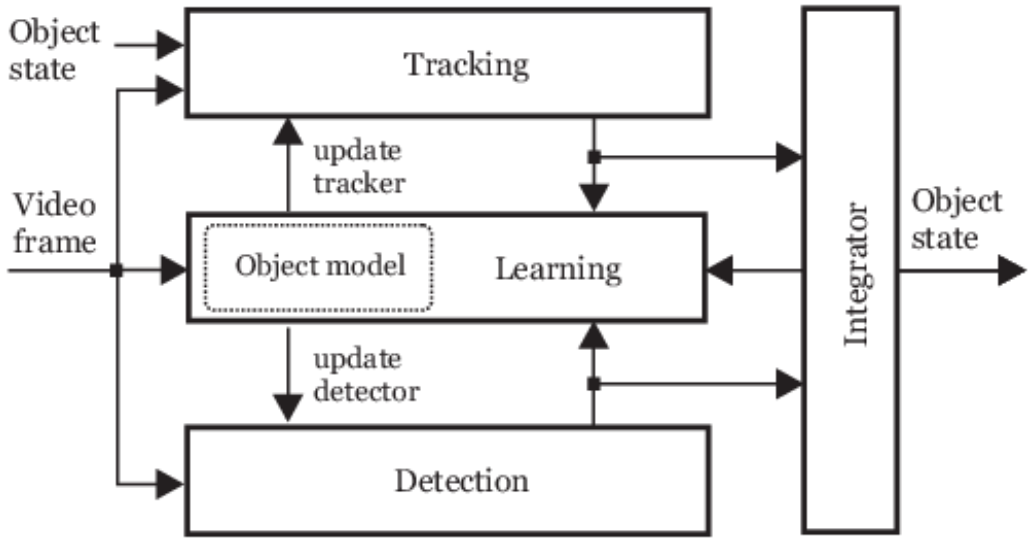


Figure 3.2: *Detailed Block Diagram of TLD framework*

3.3.2 Object tracking:

In Fragment based tracking suggested, the target is broken down in patches (image fragments).Fragmented patches are represented by intensity histogram. For tracking,it selects candidate window around theposition obtained in previous frame ,taking scale change of10 percent into consideration. Each patch votes for possibleposition of object in the current frame by comparing it withcorresponding image patch histogram, to create a vote map.Possible position is determined with the help of vote map.In another technique called Multi Instance learning , positive and negativebag of samples are created using

target examples and passedto a learner. A discriminative classifier is used for findingthe decision boundary.Next window is sampled uniformly ina circular area around previous location. The discriminativeclassifier is updated by input from new data points. Table shows the comparison between various techniques in termsof method used.In survey of object tracking,various tracking algorithms are compared .On criteria of occlusion TLD outperforms other algorithms. Thestrong performance of TLD arises from the fact that, it includes a strong feedback loop for tracking and detectionand use of, a new learning method called P-N learning ,which helps in dynamically adapting to new patterns.

3.3.3 Tracking:

The task of tracker is to estimate the motion betweenconsecutive frames, without scanning the entire frame again.In TLD algorithm, a uniform grid of 10X10
points is created within the bounding box and their corresponding points in the next frames are estimated using Lukas-kanade method. As a result, a set of optical flowvectors which describe movement from one frame to anotheris obtained. To increase the reliability , two measures areused - forward backward error and normalised cross correlation.Forward backward error is based on the assumption thatcorrect tracking does not depend on direction. Here,first a tracker produces a trajectory by tracking point inforward direction, which is of a particular length .Then avalidation trajectory is obtained by backward tracking from
the point location in the last frame. The two trajectoriesare compared and forward trajectory is rejected if it is notsimilar to validation trajectory.Normalised cross correlation is obtained by subtractingmean and dividing the standard deviation. Bounding box is accepted if forward backward error is less than median forward backward error ($med_{F\ B}$) and NCC measure is larger than median normalised cross correlation ($med_{NC\ C}$).These2 measures filter out the points with errors and increase thereliability of tracking.Advantage of using trackers is that it makes object modelof the tracker generative rather than static .New appearances on tracker trajectory are added at each step ,with the help of P-N learning described in next section. The problem with optical flow trackers is that they accumulate errorsand drift from the location of objects in case of long termtracking and loose tracking completely in case of occlusion.Therefore,in TLD algorithm detection is also used alongwith tracking, to address this problem.

3.3.4 Learning:

As tracking progresses, the object can change its appearance. P-N learning is method to update the appearance model suggested by Kalal , who provided proof to P-N learning theory. Learning algorithm works iteratively. Recursive tracker and object detector are two sourceof information for learning stage. Each iteration k finishesby retraining and correcting the classifier, i.e., estimation ofθ, where θ represents parameter of the classifier.

The task of the learning is to initialize the object detector in the first frame and update the detector inrun-time using two types of experts called, the P-expert and the N-expert. P-expert makes use of temporal structure ofinput frames and works under assumption that the objectfollows a certain trajectory, and if a detector labelled currentposition as negative, while the tracker considers this to bethe current position of the object, a false negative exampleis added to detector's set with a positive label. N-expert, onthe other hand, makes use of the fact that there may onlybe one object appearance in any given frame. It analyzes allbounding boxes provided by the detector and the one produced by a tracker. The most confident detector response ischosen as a reference patch. All the other remaining patchesthat do not overlap with the referenced one are labelled asnegative.

3.3.5 Detection:

By using sliding window approach , detector scans the entire image and identifies whether the object under the window is the desired target or not.This requires comparison ofa large number of sub -patches .One method is to directlymatch each sub-patch with object using NN classifier but itrequires huge amount of computation .So instead of directlyusing NN classifier ,cascade approach is used in whichpatches are passed through 2 stages-variance filter and ensemble classifier which reject most of the non object patches.

Advantages of TLD -

- Works the best under occlusion over multiple frames.
- Also, tracks best over scale changes.

Disadvantages of TLD -

- Lots of false positives making it almost unusable.

CHAPTER 4

4.1 GOTURN Tracker Algorithm

GOTURN,shortfor*GenericObjectTrackingUsingRegressionNetworks*,isaDeep Learning based trackingalgorithm.

Most tracking algorithms are trained in an *online* manner. In other words, the tracking algorithm learns the appearance of the object it is tracking at runtime. Therefore, many real- time trackers rely on online learning algorithms that are typically much faster than a Deep Learning based solution.

GOTURN changed the way we apply Deep Learning to the problem of tracking by learning the motion of an object in an *offline* manner. The GOTURN model is trained on thousands of video sequences and does not need to perform any learning at runtime.

Working of GOTURN Tracker -

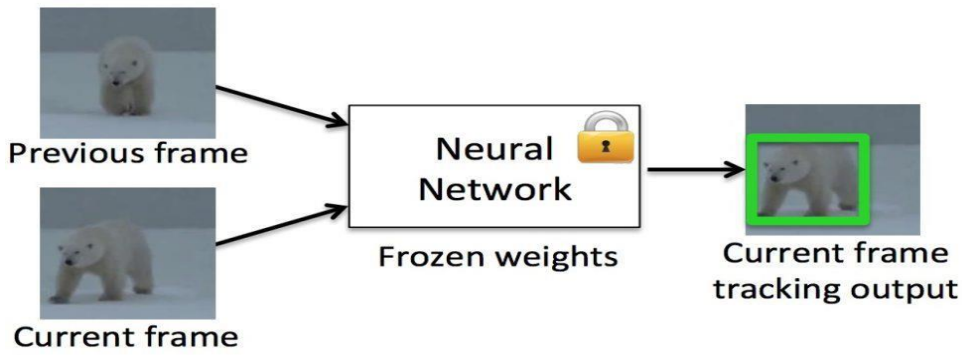


Figure4.1: *GOTURN takes two cropped frames as input and outputs the bounding box around the object in the second frame.*

- As shown in the above Figure, GOTURN is trained using a pair of *cropped* frames from thousands ofvideos.

- In the first frame (also referred to as the previous frame), the location of the object is known, and the frame is cropped to two times the size of the bounding box around the object. The object in the first cropped frame is alwayscentered.

- The location of the object in the second frame (also referred to as the current frame) needstobepredicted.Theboundingboxusedtocropthefirstframeisalsousedtocro p the second frame. Because the object might have moved, the object is not

centred in the secondframe

- A Convolutional Neural Network (CNN) is trained to predict the location of the bounding box in the secondframe.

4.1.1 CNN (Convolutional Neural Network) -

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.
An image is nothing but a matrix of pixel values, So, just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification.
In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

CNNs represent a huge breakthrough in image recognition. They're most commonly used to analyze visual imagery and are frequently working behind the scenes in image classification. Image classification is the process of taking an **input** (like a picture) and outputting a **class** (like "cat") or a **probability** that the input is a particular class ("there's a 90% probability that this input is a cat").

A CNN has

- Convolutional layers
- ReLU layers
- Pooling layers
- a Fully connected layer

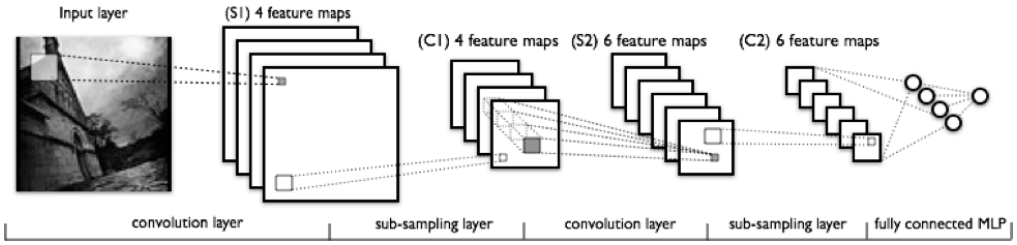


Figure4.2: *How CNN accomplish image.*

A CNN **convolves** (not convolute) learned features with input data and uses 2D convolutional layers. This means that this type of network is ideal for processing 2D images. Compared to other image classification algorithms, CNNs actually use very little preprocessing. This means that they can **learn** the filters that have to be hand-made in other algorithms. CNNs can be used in tons of applications from image and video recognition, image classification, and recommender systems to natural language processing and medical image analysis.

CNNs have an input layer, and output layer, and hidden layers. The hidden layers usually consist of convolutional layers, ReLU layers, pooling layers, and fully connected layers.

Convolutional layers apply a convolution operation to the input. This passes the information on to the next layer.

Pooling combines the outputs of clusters of neurons into a single neuron in the next layer.

Fully connected layers connect every neuron in one layer to every neuron in the next layer.

In a convolutional layer, neurons only receive input from a subarea of the previous layer. In a fully connected layer, each neuron receives input from *every* element of the previous layer.

A CNN works by extracting features from images. This eliminates the need for manual feature extraction. The features are not trained! They're learned while the network trains on a set of images. This makes deep learning models extremely accurate for computer vision tasks. CNNs learn feature detection through tens or hundreds of hidden layers. Each layer increases the complexity of the learned features.

4.1.2 GOTURN Architecture-

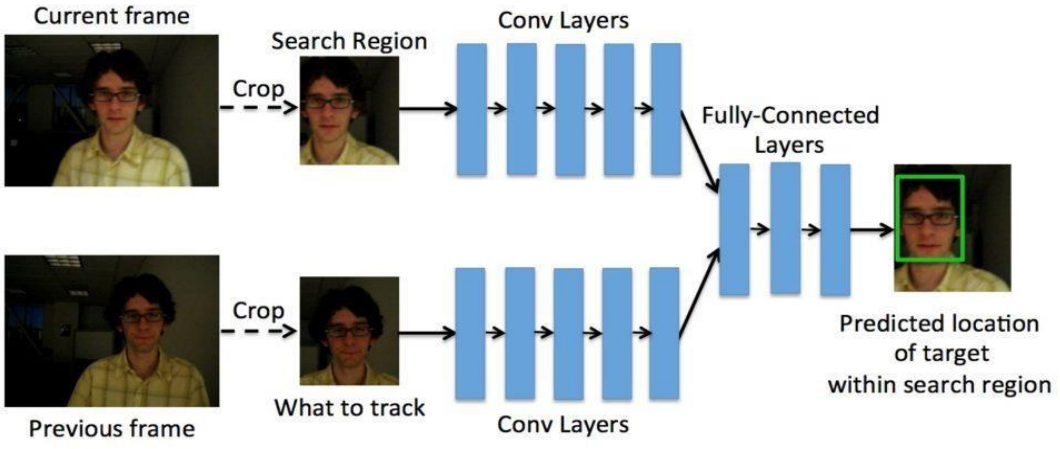


Figure4.3 :*GOTURN Architecture*

- Figure shows the architecture of GOTURN. As mentioned before, it takes two cropped frame as input.

- Notice,thepreviousframe,shownatthebottom,iscenteredandourgoalisthefindthe bounding box for the current frame shown on thetop.

- Bothframespassthroughabankofconvolutionallayers.Thelayersaresimplythefirst five convolutional layers of the CaffeNetarchitecture.

- Theoutputsoftheseconvolutionallayersareconcatenatedintoasinglevectoroflength 4096.

- This vector is input to 3 fully connectedlayers.

- Thelastfullyconnectedlayerisfinallyconnectedtotheoutputlayercontaining4nodes representing the top and bottom points of the bounding box.

Advantages of GOTURN-

- Compared to other Deep Learning based trackers, GOTURN is fast. It runs at 100FPS on a GPU in Caffe and at about 20FPS in OpenCV CPU.

- Even though the tracker is generic, one can, in theory, achieve superior results on specific objects (say pedestrians) by biasing the training set with the specific kind of object.

Disadvantages of GOTURN-

- **Trackingobjectsthatarenotinthetrainingsetinthepresenceofobjectsthatare in the training set**: For instance, while tracking the palm of a hand and as it moved over face, the tracker latched on to the face and never recovered. Even after covering the face with palm, the tracker could not get the tracker offface.
  Now, track face and occlude it with hands, but the tracker was able to track the face through the occlusion.
  There were many more faces in the training set than palms and so it has a problem tracking a hand when a face is in the neighbourhood.
  This may be a more general problem when there are multiple objects in the scene
  interacting.Thetrackermaylatchontoobjectsinthescenewhichareinthetrainingset when they come close to the tracked objects that are not in the trainingset.

- **Tracking part of an object**: It also appears that the tracker would have a hard time tracking a part of an object compared to the entire object. For example, when trying to use it to track the tip of finger, it ended up tracking the hand.

This is probably because it is not trained on parts of objects, but entireobjects.

- **LackofMotioninformation**:Sincemotioninformationisnotincorporatedinthetwo frame model, if we are tracking an object (say a face) moving in one direction, and it gets partially occluded by a similar object (say another face) moving in the other direction,thereisachancethetrackerwilllatchontothewrongface.Thisproblemcan be fixed by using the first frame as the previousframe.

4.2MOOSE TRACKING ALGORITHM

Visual tracking can be defined as the process of detecting the position of an object in each
frame of a video. Visual tracking has many real-life applications in security, surveillance,
traffic control, etc. Once an object can be tracked it can be labelled and more information
can be extracted. **Minimum Output Sum of Squared Error (MOSSE)**, a form of OptimizedCorrelation Output Filter, can be used for tracking.

The MOSSE filterwas developed by David Bolme, a PhD student in the Computer Science department andComputer Vision group of Colorado State University. MOSSE creates correlation filters,that significantly outperform simple templates and map input images to their ideal outputs.This reduces interference with the background and achieves better performance. MOSSEfilters are robust to changes in lighting, scale, pose and shape of an object.

Minimum Output Sum of Squared Error (MOSSE) uses adaptive correlation for object tracking which produces stable correlation filters when initialized using a single frame. MOSSE tracker is robust to variations in lighting, scale, pose, and non-rigid deformations. It also detects occlusion based upon the peak-to-sidelobe ratio, which enables the tracker to pause and resume where it left off when the object reappears. MOSSE tracker also operates at a higher fps (450 fps and even more). To add to the positives, it is also very easy to implement, is as accurate as other complex trackers and much faster. But, on a performance scale, it lags behind the deep learning-based trackers.Use **MOSSE when you need pure speed.**

(a) First Frame (b) Second Frame

Figure 4.4: The frames are taken from the Matrix sequence. Notice the change in target get illumination in subsequent frames. A Robust tracker should be able to handle such a change in appearance

WORKING: -

- There are two main components to the algorithm, initialization and tracking. For initialization purposes, an object is selected using either the first few frames for a simple version of the tracker or the first frame for a more complex version of the tracker.

- The object is cropped and centred to initialize the filter. The initialized filter is then correlated with a tracking window in the video to find the new location of the object.

- In a complex version, the tracker updates the filter as it tracks.

- An object is selected by a user by clicking on its centre for initializing the filter. A bounding box is drawn around the object after selecting an object for labelling. The bounding box represents the template during initialization and the tracking window during tracking.

- The template is cropped to initialize and update the filter during tracking.

- A pre-processing step is performed on each frame of the video before initializing the filterand tracking the object. The template obtained from the pre-processing step is convertedto the Fourier domain.

- A synthetic output is generated to initialize and update the filter.The synthetic output is also converted to the Fourier domain, and the filter is computed inthe Fourier domain.

- The output of the correlation is converted back to the spatial domainto find the new position of the object in the next frames of the video during tracking. Thecorrelation is performed in the Fourier domain to make the computations faster.

Figure 4.5: A frame from a video of children playing in the playground. The size of the frame is 1280 × 720 pixels.

Figure 4.6. A template of a soccer ball cropped from a frame of the video illustrated in Figure 4.5. The size of the template is 64 × 64.

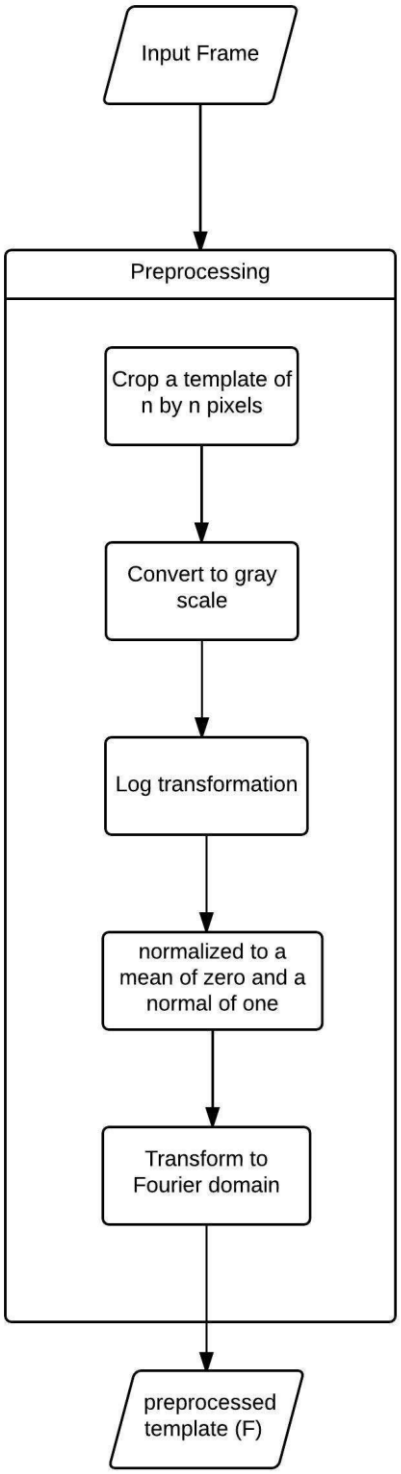


Figure 4.7. Flow chart for the pre-processing step.

4.2.1 Definition of terms: -

A series of terms and their definition in context with this tutorial are presented here.
These terms repeat throughout the session and are important to understand. It is not necessary to go through this section right now but one may return to understand the terms as required.

Object/Target: An object is a person or thing which has definable characteristics. An object in this tutorial is referred to a physical entity to be tracked. Sometimes, the object is also referred to as a target.

Template: A template is a sub-image cropped from a larger image. The template is used as an input image to initialize the filter. An image of an object is present in the template.
Also it may be referred to as correlation template.

Correlation: Correlation is a measure of similarity between two images. It is the sum of pairwise products of corresponding pixels of two images. Template matching uses correlation of a template and an image to find the location of an object in the image. The template is moved to different locations in the image and the position of the best match, highest correlation is found.

Filter: Typically, a filter is just a sub-image and the act of filtering is the process of placing the filter over the image at different locations and computing the pairwise-sum of the filter and corresponding image pixels. When the filter is applied across an entire image the result is itself a new image, i.e. a filtered image.

Tracking window: A tracking window is a sub-image in which the tracker looks for the new location of the object. The tracking window is retrieved from a frame of the video.
The tracking window correlates with the filter to give the new location of the object being tracked.

Synthetic target: A synthetic target is a synthetically generated image with a Gaussian

peak at the location of the object to be tracked. A synthetic target is used to map the input
image to its corresponding correlation output to generate a filter.
Occlusion: An occlusion is caused by an object which blocks the view of another object
in a video.

Tracking: Tracking is an action where the algorithm finds the new location of an object
in all the frames of a video over time.

Initialization: Here initialization is a process where the filter used for tracking an object
in a video is generated. In this tutorial, initialization is also referred to as training.

Updating: In this tutorial, updating is a process where a filter is updated with the new
information about the object, for example change in the pose of a person or change in the
scale of an object is a new information.

Advantages: -

- It is also very easy to implement.
- It is as accurate as other complex trackers and much faster.
- MOSSE tracker also operates at a higher fps (450 fps and even more)

Disadvantages: -

- On a performance scale, it lags behind the deep learning-based trackers.

4.3 Median Flow Tracker

Previous sections were discussing an approach for tracking of points which were considered as independent. However, in natural videos, the points are rarely independent but are parts of bigger units which move together. These units will be called objects (cars, pedes- trains,human face, etc.). This section exploits a pointtracking algorithm and the proposed FB error measure,and designs a novel robust object tracker with superior performance.

Internally, this tracker tracks the object in both forward and backward directions in time and measures the discrepancies between these two trajectories. Minimizing this ForwardBackward error enables them to reliably detect tracking failures and select reliable trajectories in video sequences.This tracker works best when the motion is predictable and small. Unlike, other trackers that keep going even when the tracking has clearly failed, this tracker knows when the tracking has failed.

WORKING: -

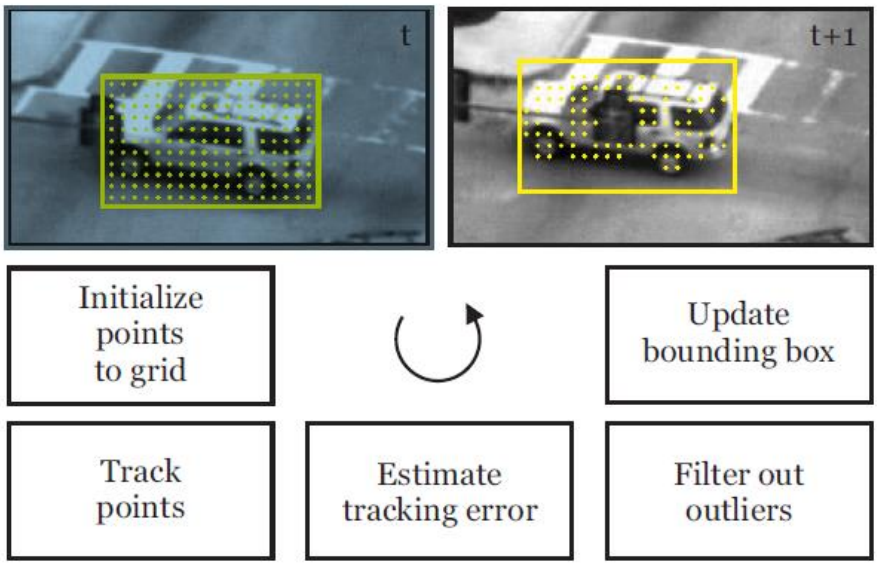


Figure 4.8 Showing an object with selected points

- The Median Flow tracker accepts a bounding box and a pair of images. A number of points within the bounding box are tracked, their error is estimated and the outliers are filtered out. The remaining estimate the bounding box motion.

- A block diagram of the proposed tracker is shown in Fig. The tracker accepts a pair of images $I_{t;}$ I_{t+1} and a bounding box $\beta_t$and outputs the bounding boxβ_{t+1}. A set of points is initialized on a rectangular grid within the bounding box $\beta_{t.}$

- Estimation of the bounding box displacement from the remaining points is performed using median over each spatial dimension.

- Scale change is computed as follows: for each pair of points, a ratio between current point distance and previous point distance is computed bounding box 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. 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.

- Based on the proposed Forward-Backward error we designed a novel tracker, called Median Flow.

EXAMPLE: -Given a point in one image (fig.), the Lucas-Kanade tracking algorithm will attempt to locate the same point in the following image. This is no trivial matter and so the points are likely to slide of target. Especially if they are not carefully selected. This means that we quickly get a cacophony of points, going of in many directions. Further we are typically interested in tracking an area/object and not a point.

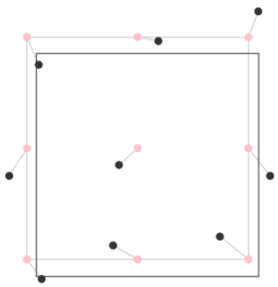


Figure4.9: Example image

Solution Of The Above Problem: - We select an area we are interested in. We then intiate a grid of points we can track using Lucas-Kanade point tracking. The question is how we update our area of interst based on the - often conflicting - motion estimates given by the LK point tracker. Median Flow suggest taking the median of the vectors. That is: list the movement of the points on the x-axis and take the median. Same for the y-axis. Scale is calculated by considering all pairs of points and meassuring the relative change in distance (distNew/distOld). Again the median is selected. This turns out to be a fairly robust estimate that compares with the best. Further it has the benefit of being simple to understand and implement.

Above example illustrates how values are calculated based on the point movement. To see changes in the medians you can drag the points (something the point tracker would normally do).

Advantages: -

- Excellent tracking failure reporting.
- Works very well when the motion is predictable and there is no occlusion.

Disadvantages: -

- Fails under large motion.
- This tracker works best when the motion is predictable and small

CHAPTER 5

CONCLUSION AND RESULTS

We have implemented and compared the performance of 7 object tracking algorithms. Every algorithm as some issue when it comes to object tracking, so we can say that one algorithm fits all situations does not exist, but we can chose the most suitable algorithm for our work.

We have summarised the comparison of all the 7 algorithms below:

ALGORITHM	PERFORMANCE	OCCLUSION	ADVANTAGES	DISADVANTAGES
Boosting	Mediocre		None. Works ok.	Slow. Poor failure reporting.
MIL	Good	Reasonable for partial, Does not recover from full occlusion	Reasonable job, Best tracker for openCV 3.0	Poor job of reporting failure
KCF	Better than MIL and Boosting	Cannot recover from full occlusion	Accuracy and speed are better.	Cannot be implemented in lower version, like openCV 3.0
TLD	Not so good	Works best under occlusion over multiple frames	Tracks best over scale changes	Prone to false positives making it unusable
MEDIAN FLOW	Great	Works well under predictable motion and no occlusion	Nice job reporting failures	Model will fail under fast moving objects
CSRT	Very good		High accuracy	Slower fps throughput

GOTURN	Average	Does not handle it well	Only tracker to work on CNN	Requires few additional files to run
MOSSE	Pretty good	Detects occlusion well	Very, very fast. Robust to variations.	Accuracy is low, lags behind the deep learning based trackers.

References

[1] S. Avidan. Ensemble tracking. *PAMI*, 29(2):261–271,2007.

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual track-ing with online multiple instance learning. *CVPR*, 2009.

[3] J. Bouguet. Pyramidal Implementation of the LucasKanade Feature Tracker Description of the algorithm.*Technical report, OpenCV Document, Intel Micropro-cessor Research Labs*, 1999.

[4] R. Collins, Y. Liu, and M. Leordeanu. Online selectionof discriminative tracking features. *PAMI*, 27(10):1631–1643, 2005.

[5] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors duringunstable tracking.*OLCV*, 2009.

[6] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. *CVPR*, 2010.

[7] J. Lim, D. Ross, R. Lin, and M. Yang. Incremental learning for visual tracking. *NIPS*, 2005.

[8] B. Lucas and T. Kanade. An iterative image registrationtechnique with an application to stereo vision. *IJCAI*,81:674–679, 1981.

[9] K. Nickels and S. Hutchinson. Estimating uncertainty inSSD-based feature tracking. *IVC*,20(1):47–58, 2002.

[10] E. Rosten and T. Drummond. Machine learning for highspeed corner detection. *ECCV*, May 2006.