Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 8 (2017) pp. 2565-2585

http://www.ripublication.com

Color Based Vehicle Detection and Tracking using Kalman Filter with a Fractional Feedback Loop

Prasanna Kumar Reddy Katuru[1], Vivek Josyula[1], Praveen Samudrala[1]
M. Lokanath[1]* and P. Ashish[1]

[1]School of Electronics Engineering, VIT University, 632014, India.

Abstract

Detecting and tracking of vehicles has become an important application of computer vision these days, tracking and predicting the vehicle is essential when it goes out of visibility for example police chase or futuristic car applications. The basic essence of this paper is to develop a mechanism using Kalman Filter to detect and track a moving vehicle. To increase the stability and tracking capabilities, this paper implements a Fractional Feedback Mechanism. Taking into account the choice of the user, we have provided a mechanism that allows the user to choose the color of the vehicle that he/she would like to track. In this paper, first vehicle tracking is implemented, it is further extended to color based vehicle tracking. Results are obtained by implementing out designs over two platforms, namely MATLAB and C++ using OpenCV libraries. The proposed work is implemented in real time, with challenging scenarios and the results showcases the lustiness of the proposed work.

Index Terms: Kalman Filter, Vehicle Tracking, Vehicle Detection, Fractional Feedback.

I. INTRODUCTION

As the number of vehicles on the roads have increased substantially lately, it is also getting tougher and tougher to keep a watch on such vehicles that cause misdemeanors on a consistent basis. As the statistics suggest, there has been a significant jump in the number of car chases reported. Often for high speed chases, cameras equipped on helicopters are used to keep a bird's eye watch of the vehicle

trying to escape. The skill of a driver to steer into high speed lanes and the ability to hide using buildings and tunnels may help them occasionally to shake loose of the helicopter tracking them.

One of the most widely used tracking algorithms is the Kalman Filter [3] algorithm which is an optimal recursive state estimator with minimum error covariance. Since 1968 constant gain Kalman Filters have been analyzed [2]. They provide satisfactory results in stand-alone and data fusion mode for tracking [4] [5] [6]. Constant gain Kalman Filter has been modified to adaptive gain Kalman filter to further improve it's performance [7]. An Extended Kalman is utilized to realize high dynamic tracking of GPS signal [8] which has improved tracking accuracy and dynamic acceleration. An object tracking system which combines Support Vector Machines (SVM) and Kalman filter is proposed where Kalman filter is used to predict the dynamics of the target object [8]. An Intelligent Transport System (ITS) [15] is developed to make an efficient road transport system, where Guassian Mixture Model (GMM) was applied for vehicle detection and Kalman filter is applied for object tracking [9]. In case of moving vehicles in both lanes, environmental variations, vision based traffic monitoring systems are robust and are accurate [10] [11]. A fractional order Kalman Filter is proposed in [12] [16] which is used for tracking purposes which predicts the new state utilizing fractional calculus of previous states. Fractional calculus gives steady-going results for analyzing few systems [12] [17]. Further, the divergence of Kalman filter will increase when the rate of convergence is slow [1]. Minimum variance state estimator defines the practical stability of Kalman filter [13].

The rest of the paper is organized as follows: Section II describes few tracking algorithms and presented our proposed algorithm. Section III represents a testing of our algorithm in various real time scenarios. Section IV gives the analysis and discussion. Section V concludes the paper and section VI gives our references.

This paper proposes a mechanism to predict the movements of a vehicle even when it breaks the Line-Of-Sight. We implement a three-step procedure for transition, develop and test our proposed method.

Firstly, we start with a simple design and develop a program to detect and track a single and multiple object. Secondly, we shift platform from MATLAB to C++ platform, to facilitate real time tracking more efficient. Therefore, initially we implemented multiple object detection over C++ platform and OpenCV libraries. Finally, we implement single vehicle target tracking using C++ platform using OpenCV libraries.

II. ALGORITHMS AND IMPLEMENTATIONS

Kalman Filter's success is accredited to majorly to its ability to converge faster to the manually measured value even if fluctuations occur due course of the simulation. Essentially, the filter assumes an arbitrary value and tries to converge to the measured value. Kalman Filter uses Attributes from previous states to predict the future state and compares the measured values to provide any adjustments, if necessary, to increase the accuracy and precision of the System.The functionality of a Kalman filter is explained in Fig-1.

The system performs its operations mainly based on 4 inputs for each iteration, of which 3 are constants and do not change throughout the process which are Measured Value, Error in Data, Original Error Estimate and one variable which is an error throughout the process which are Measured Value, Error in Data, Original Error Estimate and one variable which is an error in the estimate.

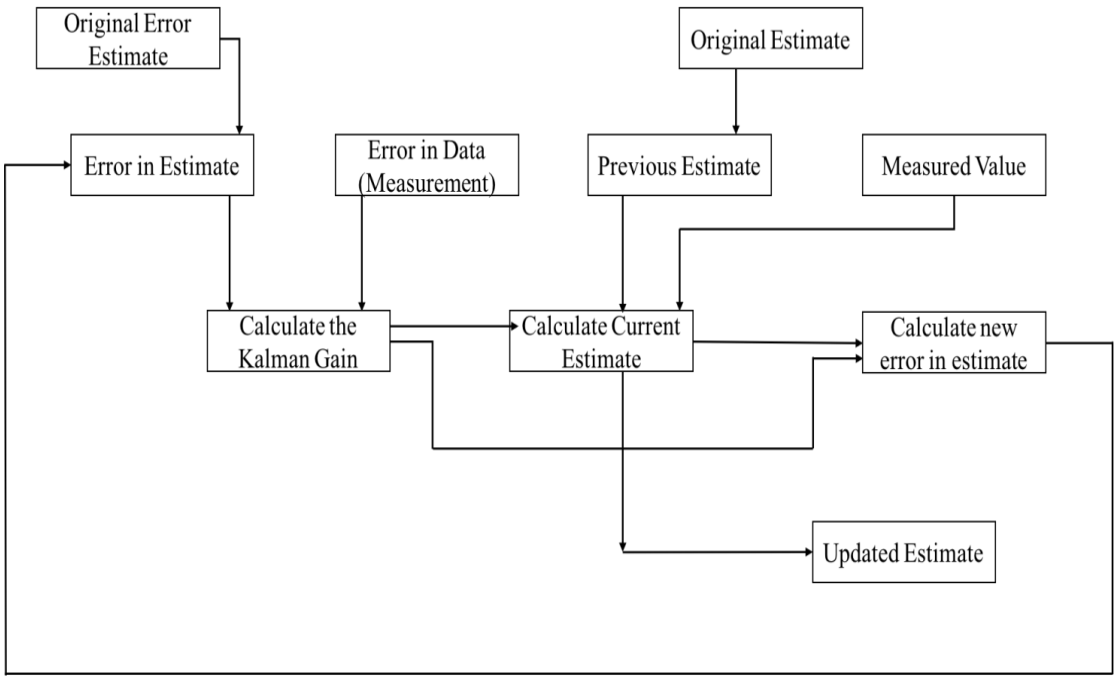

Fig-1. Block Diagram of the Kalman Filter.

Kalman Gain(KG) is defined as the ratio of Error in Estimate(Eest) to the sum of Error in estimate(Eest) and Error in Measurement(Emea). This can be mathematically represented as:

$$KG = \frac{E_{est}}{E_{est}+E_{mea}} \quad (1)$$

The Current Estimate (Estt) value for the present iteration is calculated from previous estimate (Estt-1), Measurement (Mea), and Kalman Gain (KG) as follows:

$$Est_t = Est_{t-1} + KG * [Mea - Est_{t-1}] \quad (2)$$

The current Error in Estimate (E_{est_t}) is calculated as follows:

$$E_{est_t} = [1 - KG] * E_{est_{t-1}} \quad (3)$$

Here, the function of Kalman Gain is highlighted. It acts as a ratio that gives a weight to the deviation of the predicted value from the actual value. In other words, the Kalman Gain takes up a value between 0 and 1, which takes a fraction of the difference between the estimate and measured value and adds it to the existing estimate to provide an updated version of the estimate that needs to be used for prediction of subsequent iterations.

If KG is approximately equal to 1, then a very quick convergence can be observed. Eventually, KG approximating to 0 implies a longer convergence time period. If Emea is approximately equal to 1, then a very slow convergence occurs. And, if Emea tends to 0, a fast convergence is observed.

Mathematically, we can express the algorithm using Matrix blocks as shown in Fig-2. In Fig-2, the present and previous state contain State Matrix (X) and Process Covariance Matrix (P), using the matrixes we will predict the new state matrix and Process covariance matrix with their respective equations where Adaption Matrices (A, B, C), Control Variable Matrix (U), Predicted State Noise Matrix (W) and Process Noise Covariance Matrix (Q).

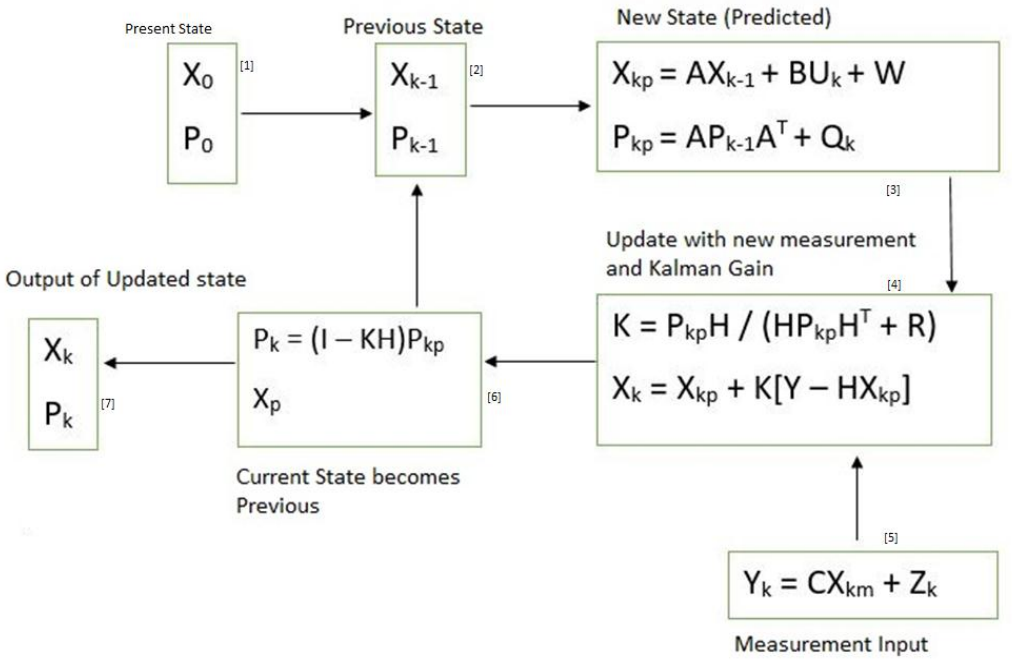

Fig-2. State Matrix diagram of Kalman Filter

We shall now see how each block works and what it adds to the process of Tracking:

The process starts with a predefined matrix that contains information about the present state of the variables. In many cases, it is observed to be zero as the process of tracking has not been started yet. But, in the first iteration the information from the first frame is updated onto this Matrix. The properties of the previous matrix are passed on to the Present Matrix, which will further be updated later on in the process. Therefore, in our second iteration, i.e. as the second frame is read by the system, we see that information from first frame is updated onto this matrix, as first frame becomes the previous state of the present frame. Consequently, we use complex calculations and matrix operations to calculate the predicted values and predicted attributes.

The main purpose of adaptation matrices is to transform the matrix dimensions of X_{k-1} and U_k to match the dimension of X_{kp}. In the next state stage, Kalman Gain and X_p are calculated with inputs from Predicted State matrix and Measurement Matrix. These values shall be used in further stages to update our predicted states that need to be ready for the next iteration. The next stage is used to calculate P_k and X_p that will be used to update attributes/parameters for the next iteration. In this 'I' stands for Identity Matrix. Finally, in the last stage the updated state variables are stored.

Now, let's look into a mechanism for each algorithm proposed:

A. Multiple Object Detection and Tracking (On MATLAB):

In this section, we try to detect and track multiple objects using blob analysis and morphological processing. The algorithm used is as shown in Fig-3.

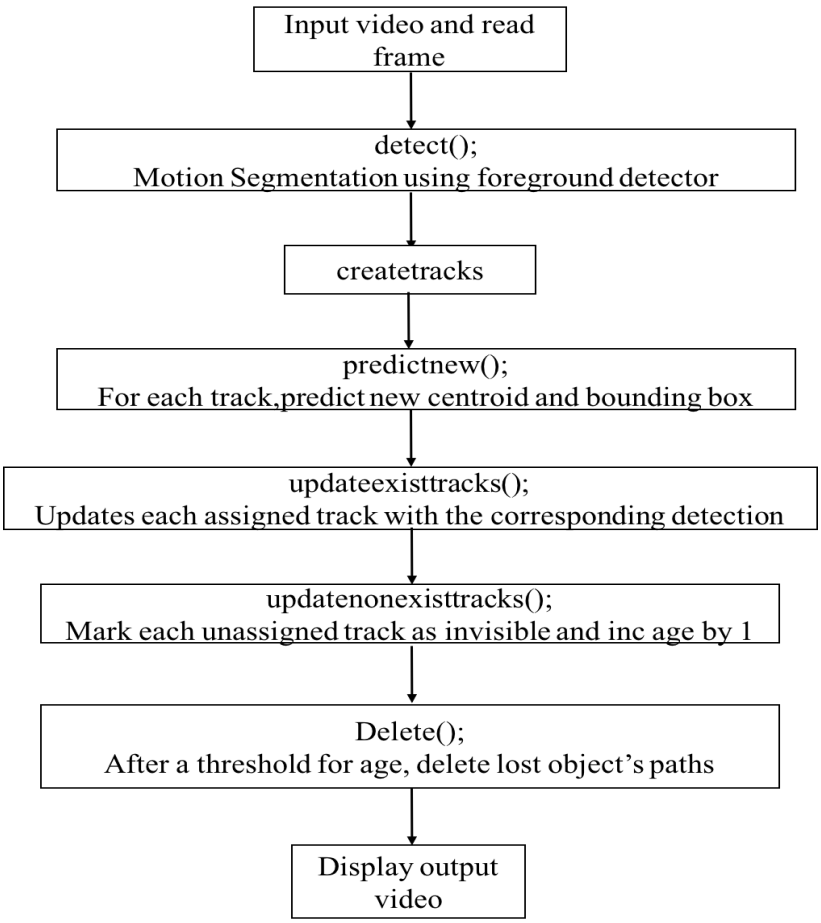

Fig-3. Multiple Object Tracking Algorithm.

To allow easy processing the Video used is broken down into frames to be processed individually. Then, each frame is read and analysis is performed to check for movement, eventually to enable detection. In the next section, we define a function called 'detect', whose task is to detect the foreground objects and the motion, if caused. Foreground detection used segregates the moving object from the background objects. The next section provides a function 'predictnew', that predicts the new location of the centroid of the object being tracked and updates the bounding box used to track it. The function 'detectionToTrack' is used to assign new tracks to new objects that prove motion in a frame. These tracks are then appended to the existing tracks that have been allocated memory space using an array. 'updateexisttracks' function provides a mechanism to update every element in the array that contains

tracks, as and when new detections and predictions of existing tracks are made. 'updatenonexisttracks' function provides a mechanism where objects which become invisible are given a variable called 'age'. If the age of the object crosses a threshold, then its corresponding element in the array is deleted. As mentioned earlier, after the object's age reaches a certain threshold, we delete the object's track information from the array. This is facilitated by the function 'delete'. For the new objects that might enter the frame of observation, we provide a function 'createTracks' to accommodate detection and tracking for those objects. Finally, the tracks developed are displayed using the function 'display output Video'.

B. *Multiple Car Detection* using *C++ (On OpenCV)*

For real time video processing, and to facilitate a working mechanism across different platforms, we have chosen OpenCV as a platform to extend our work. The programming language has though been C++. The algorithm is as shown in Fig-4.

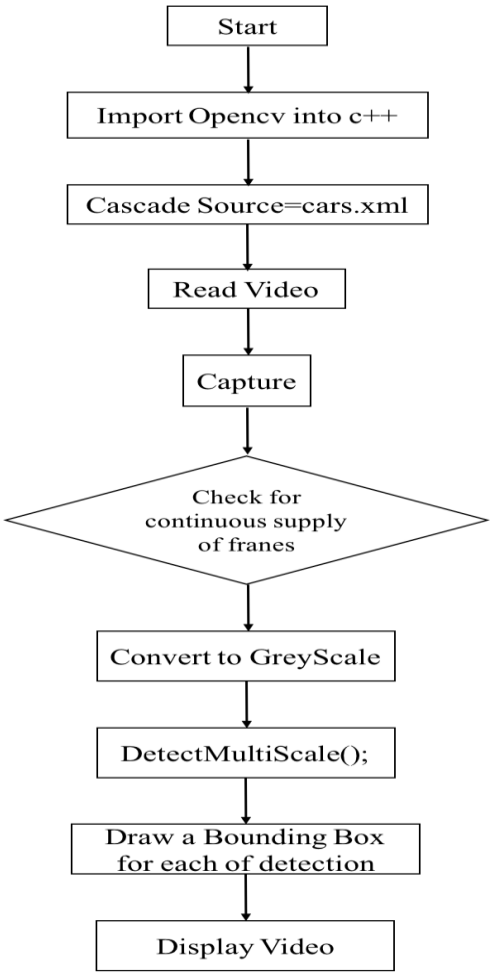

Fig-4. Multiple Object Tracking Algorithm (Using C++).

To enable programs written in C++ to run on OpenCV platform, we need to first import the required directories. A predefined xml file called "cars.xml" is used to provide it as a source file for a cascade data structure. This file trains the program to detect only cars, and not any other moving object. At this point we read the Video to which our detection mechanism has to be applied. The variable 'capture' is used to convert a continuous stream of video frames, which are then processed for detection. The 'cascade' declared earlier needs to be imported. This process combines the cars.xml file and the video frames to detect only the required objects (in our case, Cars). A condition to be met is that the stream of frames provided needs to be unbroken, i.e. if there is a break/pause in the stream of frames provided, the program terminates. This can be used to stop the detection process when the video played is complete. The set of frames being passed are converted to grayscale, as detection that occurs need not happen only on RGB images. In fact, detection is faster on Greyscale Images.

The function that is the heart of the process is 'detectMultiScale'. A Bounding Box is drawn for each of the detected car to denote detection and continues to draw the box around that car as long as it is in the frame of detection. The final step is to display the video along with the detections.

C. Tracking a Targeted Car using C++ (On OpenCV):

This is the ultimatum our project that we intend to achieve. We intend to track one and only one car that needs tracking. The selection of the car is user generated, i.e. the user needs to choose the color of the car and the tracking of that car alone starts. The platform used is OpenCV and programming language is C++. The algorithm used is as shown in Fig-5

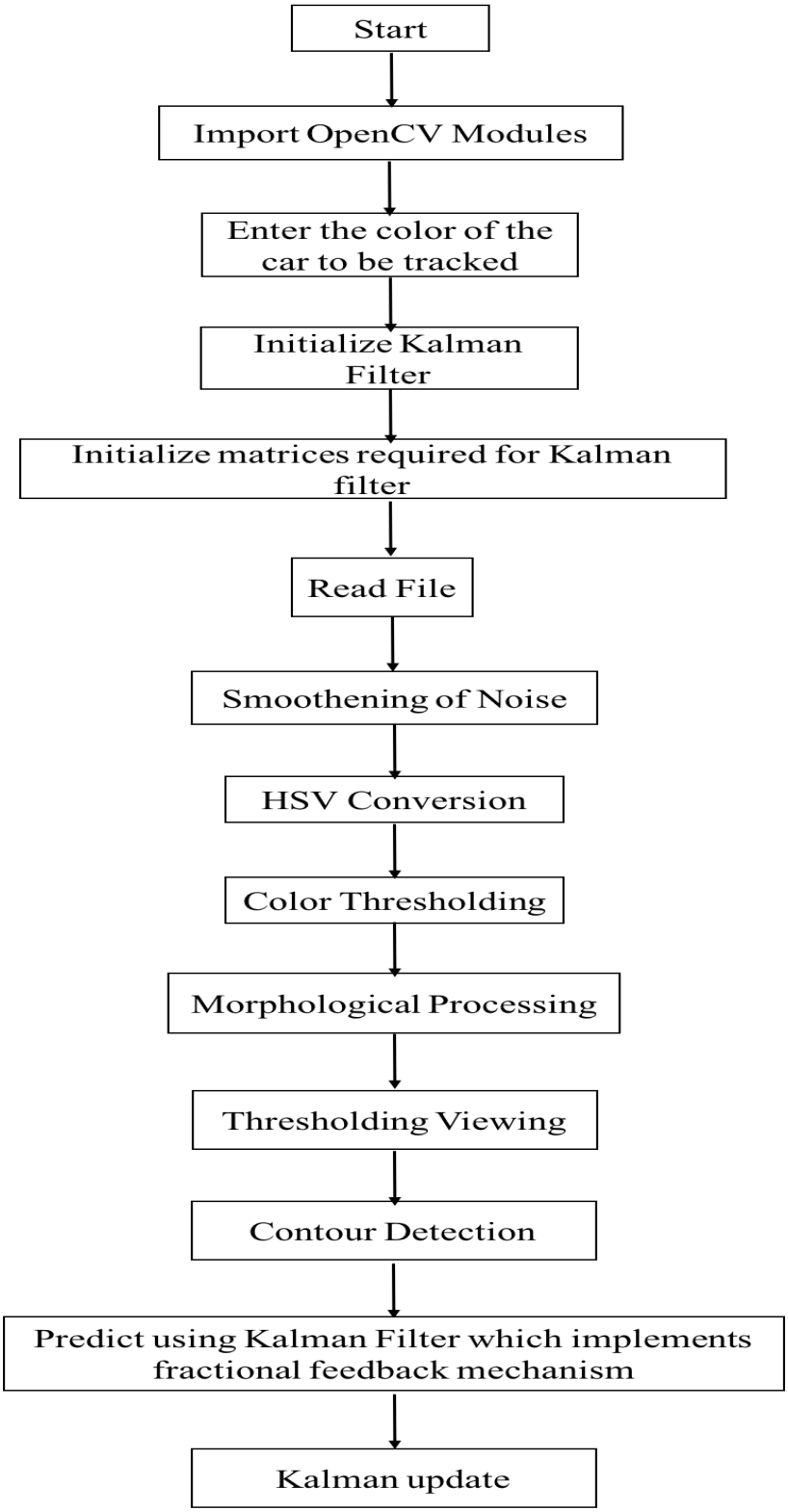

Fig-5. Algorithm for tracking a targeted car (using C++).

The program starts with importing OpenCV modules that enable us to start and initialize tracking mechanisms. In our program, the user is given a choice to select the color of the car he/she would like to track. This is done via setting thresholds for hue color values. Now, we initialize Kalman Filter that needs to be used for Detection purposes. As discussed in the Mathematics behind Kalman Filter, the state Matrices such as A, H, Q are initialized. We read the video as frames. Then, we predict the positions and state variables using Kalman Filter. Next, we use filters to smoothen out noises as excessive noise can cause the Kalman Filter to misread many of the inputs, resulting in irregular and inaccurate results. The term 'HSV Conversion' refers to converting the car's color into hue values and applying the threshold specified by the user. When the color threshold is used, all the objects that fall into the range of allowed hue levels are detected, but only the necessary car is tracked due to the effect of locking mechanism. Morphological Processing is used to erode and dilate the edges of the car, so as to allow the program to detect the moving object more efficiently. Threshold Viewing is the process where after all the discussed filters are applied and only selected threshold levels of hue values are considered for tracking. Then, locking mechanism is used by the program, so that it does not lose track of the target being tracked. General Formula to enable locking mechanism is by defining the threshold level as the ratio of width of the bounding box to a height of the bounding box. If the ratio drops down, then the tracking mechanism fails. It is at this stage we implement our fractional feedback mechanism. What it essentially does is that the Kalman Gain is averaged over a period of time and added to the New Kalman Gain calculated. Doing this will increase the stability of the tracking mechanism. So, there will be no abrupt changes or fluctuations in our tracking mechanism (We will see later in the results section as to how it is stabilized). Penultimately, we display the results. The final step ensures that the Kalman Updates are sent accordingly for the tracking to happen smoothly for the next iteration.

III. RESULTS

The results provided are screenshots of execution of detection and tracking for the video input provided. For each method, the inputs are different.

A. Multiple Object Detection and Tracking (On MATLAB)

Fig-6 and Fig-7 are the outputs acquired when multiple moving objects are detected using the designed algorithm. Notice that only the moving objects are detected. Even though there are many other objects in Fig-6, we detect only the moving objects.

Fig-6 Detection of a single object on the road.

Fig-7 Detection of Multiple Objects on Road.

B. Multiple Object Detection using C++ (On OpenCV):

In our first output (Fig-8), we can see that bounding boxes are drawn to all the cars in the frame. The Fig-9 shows that the Truck is not detected. The reason being that 'cars.xml' file was used to train the program to track cars alone.

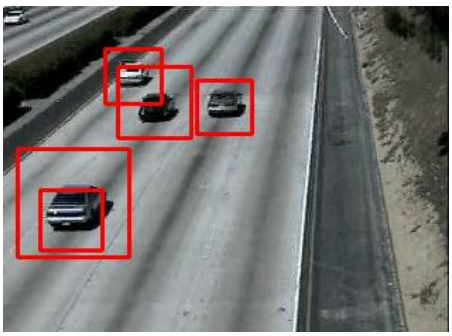

Fig-8 Detection of Multiple cars, as predicted.

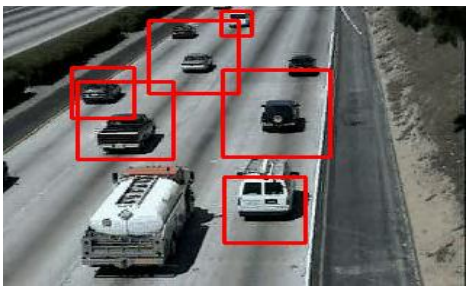

Fig-9 Detection of Cars only, even in presence of Truck

C. Tracking a Targeted Car using C++ (On OpenCV):

For testing purposes, we consider a high-speed police chase in the city of Phoenix, Arizona, USA obtained from YouTube. The Green box denotes detection and Red box denotes Tracking and Estimation.

The Fig-10 shows how all the yellow objects (User defined color), are detected and given a bounding box. Fig-11 shows the algorithm used locking mechanism to lock onto car's motion. In Fig-12, even though the car breaks the line of sight, the tracking does not stop. Additionally, the algorithm predicts the movement, as shown by the red box. In Fig-13, recapturing occurs as the car is visible again. The Fig-14 shows that the algorithm does not track a yellow bus nearby as the algorithm has already locked on to the yellow car.

Fig-10 Initializing the tracking algorithm for Yellow Car.

Fig-11 Tracking of Car targeted, before breaking visibility.

Fig-12 Tracking of Car targeted, when it is invisible to the source

Fig-13 Recapturing the desired car after it is visible again.

Fig-14 Only car is tracked as locking mechanism is still in place.

D. ***Additional testing of proposed algorithm using a camera recorded video***:

For the purposes of real-time application of our algorithm, we gave our own recorded video as input (A toy car). The toy car is detected in Fig-1 before disappearing behind the opaque object. When the toy car breaks LOS, we observe that the algorithm continues to track the motion (by prediction). Finally, when the toy car emerges on the other side, the object is recaptured. We see that deviation increases with increase in time of invisibility.

Fig-15

Fig-16

Fig-17

Fig-15 Tracking the targeted toy car, before breaking invisibility.

Fig-16Tracking the targeted toy car, when it becomes invisible.

Fig-17. Recapturing the targeted toy car when it reappears.

IV. GRAPHICAL ANALYSIS

We analyze the outputs using graphical tools to determine the efficiency and capabilities of our tracking mechanism. We confine our analysis to the Subsections C and D of Part III of this paper (results); As our final results are defined by subsections C and D. Analysis is done using two graphs. Our first graph is a plot of the positions of centroids of predicted and detected bounding boxes. The second graph is a plot representing the difference between the centroids of Detected and Predicted Centroids of Bounding Boxes.

A. Analysis of Targeted Car Tracking Result:

This section analyzes the results of our algorithm concerning with Fig-10, 11,12,13,14. The Table-1 is the tabulation of detected (XD) and predicted (XT) x axis position of the centroid pixel of the bounding box w.r.t to their frame numbers. From the legend of the graph, Fig-18, it is clearly understandable that the Blue plot refers to the parameters when the object is detected. Similarly, the Red plot refers to the parameters that represent the Predictions our algorithm makes.

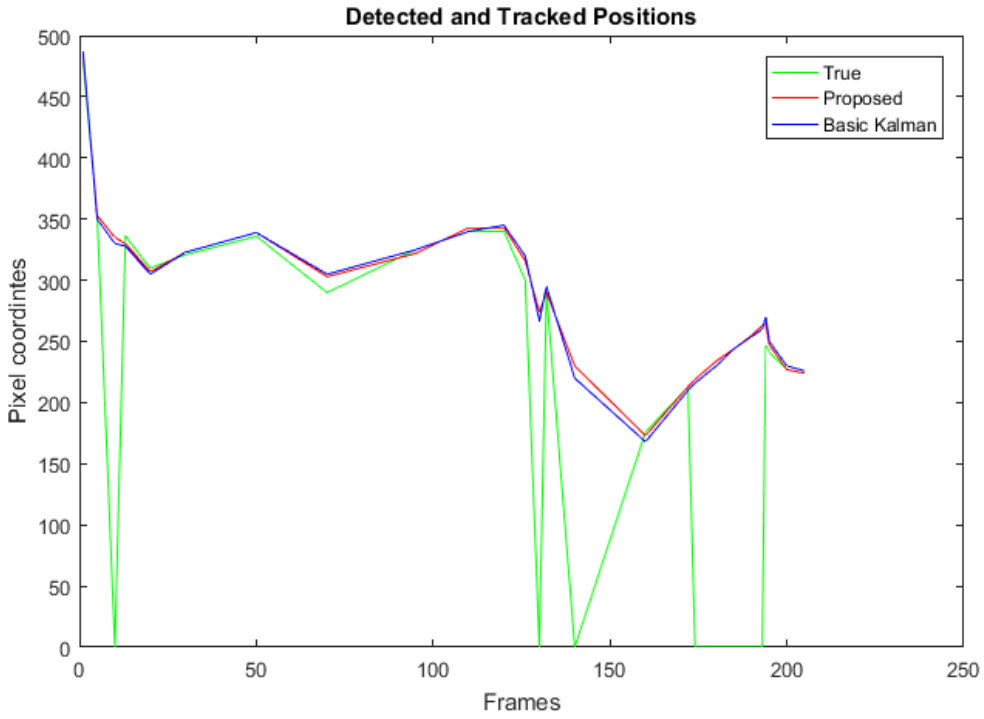

Fig-18 Graphical representation of Detected and Predicted Locations

Table-1. Tabular representation of Pixel values of centroids of Detected and Predicted Boundary Boxes

Time	X D	X P
1	480	485
5	353	353
10	NA	335
13	336	330
20	310	307
30	321	323
50	336	339
70	290	303
95	325	322
110	340	343
120	340	343
126	300	316
130	NA	274
132	290	290
140	NA	230
160	176	173
172	213	213
174	NA	219
180	NA	234
185	NA	244
190	NA	255
193	NA	263
194	247	264
195	241	247
200	227	227
205	224	224

It is to be noted that, when the object is invisible, or breaks the Line Of Sight, the Blue plot drops down to Zero, indicating invisibility. During this period of Invisibility, our system continues to predict the object that needs to be tracked.

Table-2 represents the difference between XD and XP. The difference is high when the object breaks Line Of Sight. This is a testament to the capability of our system as it is more efficient and takes lesser time to actually converge to the detected, original, value. Fig-19 is the graphical representation of the tabulated values from Table-2.

Table-2. Tabulated values of Difference between pixel values of centroids

Frame	differential
1	5.099019514
5	0
13	6
20	3.16227766
30	2.236067977
50	3.605551275
70	14.31782106
95	5
110	3.16227766
120	3.605551275
126	16.76305461
132	0
160	3
172	0
194	25.49509757
195	6
200	0
205	1

B. Analysis of our Second Result:

Here, analysis is done for testing for camera recorded video concerning Fig-15, 16,17. In the second simulation of our algorithm, we used a camera recorded video where the object would disappear for a few seconds behind a book. Thus, our system had to guess the location of the toy red car. From the Table-3, it is very clear that the toy car was invisible for the frames 20 through 35. Even though the LOS was broken, our tracking mechanism did not deviate much from the path attributes. Fig-20 is the plot analysis of Table-3. We can see from the downward slope that the slope fall is not abrupt and maintains the slope of the existing predicted path, more or less.

In Table-4, differential values are indicated from frames 15 through 55. In this brief period of time, it is maximum from frames 20 through 35. As soon as the object appears again, the tracked value coincides with the predicted value. Fig-21 is the graphical representation of the values from Table-4. As observed, the difference peaks around the 45th Frame. But, as soon as the object appears again, the convergence starts and the end result is fast convergence of estimates and detections.

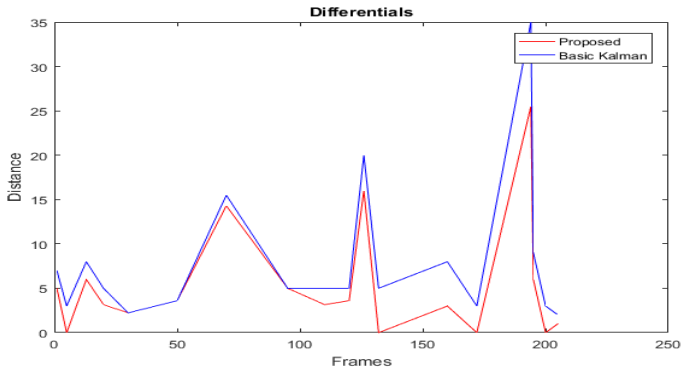

Fig-19. Graphical representation of Deviation of Predicted from Detected Location

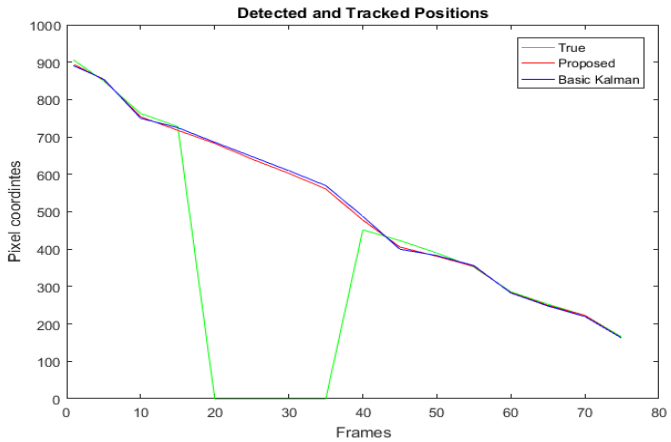

Fig-20 Graphical representation of Detected and Predicted Locations

Table-3. Tabular representation of Pixel values of centroids of Detected and Predicted Boundary Boxes.

Frame	XD	XT
0	905	894
5	850	853
10	763	754
15	728	718
20	NA	683
25	NA	641
30	NA	603
35	NA	561
40	452	478
45	423	406
50	390	381
55	352	354
60	286	284
65	253	250
70	223	223
75	165	162

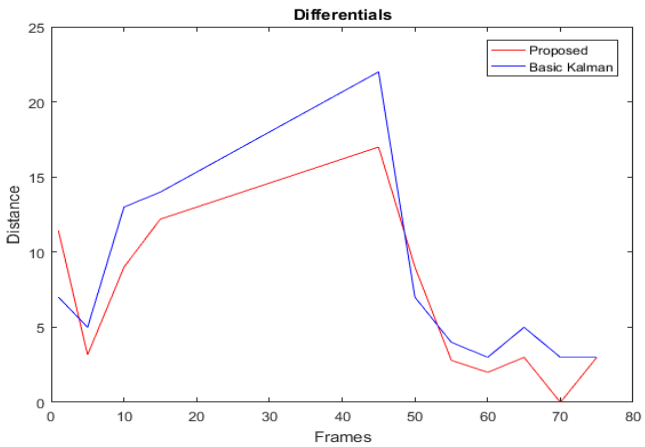

Fig-21. Graphical representation of of Deviation of Predicted from Detected Location

Table-4. Tabulated values of Difference between pixel values of centroids.

Frame	differential
0	11.40175425
5	3.16227766
10	9
15	12.20655562
20	NO LOS
25	NO LOS
30	NO LOS
35	NO LOS
40	NO LOS
45	17.02938637
50	9.055385138
55	2.828427125
60	2
65	3
70	0
75	3

V. CONCLUSIONS

This project has given an insight as to how a selected vehicle is detected and tracked. So, our phase-by-phase process of implementing a Kalman filter has proven to be effective and successful. We started with algorithms to detect multiple cars on OpenCV, and successfully tracked, targeted cars in a given scenario. Finally, we have achieved the ultimatum of tracking only the desired car and additionally provided the user with an option to choose the color of the car he/she need to track. It is at this stage that Fractional Feedback is used.

REFERENCES

[1] HARPREET KAUR AND J. S. SAHAMBI, IEEE TRANSACTIONS ON COMPUTATIONAL IMAGING, VOL 2, NO. 4, DECEMBER 2016.

[2] D. KLEINMAN, T. FORTMANN, AND M. ATHANS, "ON THE DESIGN OF LINEAR SYSTEMS WITH PIECEWISE-CONSTANT FEEDBACK GAINS," IEEE TRANS. AUTOMAT.CONTROL, VOL. 13, NO. 4, PP. 354–361, AUG. 1968.

[3] R. E. KALMAN, "A NEW APPROACH TO LINEAR FILTERING AND PREDICTION PROBLEMS,"ASME J. BASIC ENG., VOL. 82, PP. 35–45, 1960.

[4] A. ANIL KUMAR, M. ANANTHASAYANAM, AND P. S. RAO, "A CONSTANT GAIN KALMAN FILTER APPROACH FOR THE PREDICTION OF RE-ENTRY OF RISK OBJECTS," ACTA ASTRONAUT., VOL. 61, NO. 10, PP. 831–839, 2007.

[5] G. COOK AND D. E. DAWSON, "OPTIMUM CONSTANT-GAIN FILTERS," IEEE TRANS. IND. ELECTRON. CONTROL INSTRUM., VOL. IECI-21, NO. 3, PP. 159–163, AUG. 1974.

[6] A. YADAV, P. AWASTHI, N. NAIK, AND M. ANANTHASAYANAM, "A CONSTANT GAIN KALMAN FILTER APPROACH TO TRACK MANEUVERING TARGETS," IN PROC. IEEE INT. CONF. CONTROL APPL., AUG. 2013, PP. 562–567.

[7] A. ALMAGBILE, J. WANG, AND W. DING, "EVALUATING THE PERFORMANCES OF ADAPTIVE KALMAN FILTER METHODS IN GPS/INS INTEGRATION," J. GLOBAL POSITIONING SYST., VOL. 9, NO. 1, PP. 33–40, 2010.

[8] C. ZENG AND W. LI, "APPLICATION OF EXTENDED KALMAN FILTER FOR TRACKING HIGH DYNAMIC GPS SIGNAL," *2016 IEEE INTERNATIONAL CONFERENCE ON SIGNAL AND IMAGE PROCESSING (ICSIP)*, BEIJING, CHINA, 2016, PP. 503-507.

[9] Y. Y. CHEN, P. H. CHEN AND S. C. CHIEN, "AN OBJECTIVE TRACKING METHOD BASED ON KALMAN FILTER," *2016 INTERNATIONAL CONFERENCE ON ADVANCED ROBOTICS AND INTELLIGENT SYSTEMS (ARIS)*, TAIPEI, TAIWAN, 2016, PP. 1-1.

[10] INDRABAYU, R. Y. BAKTI, I. S. ARENI AND A. A. PRAYOGI, "VEHICLE DETECTION AND TRACKING USING GAUSSIAN MIXTURE MODEL AND KALMAN FILTER," *2016 INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE AND CYBERNETICS*, MAKASSAR, INDONESIA, 2016, PP. 115-119.

[11] C. SETCHELL AND E. DAGLESS, "VISION-BASED ROAD-TRAFfiC MONITORING SENSOR," IEE PROC. VIS., IMAGE SIGNAL PROCESS., VOL. 148, NO. 1, PP. 78–84, FEB. 2001.

[12] M.-C. HUANG AND S.-H. YEN, "A REAL-TIME AND COLOR-BASED COMPUTER VISION FOR TRAFfiC MONITORING SYSTEM," IN PROC. IEEE INT. CONF.

MULTIMEDIA EXPO, JUN. 2004, VOL. 3, PP. 2119–2122.

[13] D. SIEROCUIK AND A. DZIELINSKI, "FRACTIONAL KALMAN FILTER ALGORITHM FOR THE STATES, PARARMETER AND ORDER OF FRACTIONAL SYSTEM ESTIMATION," INT. J. APPL. MATH. COMPUT. SCI, VOL. 16, NO. 1, PP. 1129–140, 2006.

[14] B. NI AND Q. ZHANG, "STABILITY OF THE KALMAN FILTER FOR OUTPUT ERROR SYSTEMS," IFAC-PAPERSONLINE, VOL. 48, NO. 28, PP. 1106–1111, 2015.

[15] S. SRI HARSHA AND K. R. ANNE, "A HIGHLY ROBUST VEHICLE DETECTION, TRACKING AND SPEED MEASUREMENT MODEL FOR INTELLIGENT TRANSPORT SYSTEMS," VOL. 11, NO. 5 PP 3733-3742, IJAER, 2016.

[16] A. GEETHA DEVI, T. MADHU, K. LAL KISHORE "DETECTION, IDENTIFICATION AND TRACKING OF MOVING OBJECTS IN A VIDEO USING SUPER RESOLUTION – A NOVEL APPROACH," VOL. 10, NO. 3, PP 5471-5487, IJAER, 2015.

[17] J. JOSEPH ANTONY AND M. SUCHETHA "VISION BASED VEHICLE DETECTION: A LITERATURE REVIEW," VOL. 11, NO. 5, PP 3128-3133, IJAER, 2016.