Online tracking Of A Moving Object Using Kalman Filter

Table of Contents

Praharsha Sirsi (s8prsirs@stud.uni-saarland.de) (2557724)
Shruthi Yaddanapalli (shruthiyaddanapalli@gmail.com) (2551067)

Table of Figures

1 Introduction

The project attempts to implement online tracking of moving object using a WebCamera or a video file as input. Object tracking is used in many areas for different reasons. In this project, video tracking is a process of tracking a moving object. Video tracking is used in computer vision, robotics, Traffic Management, Security and many more application areas. The object trackers usually need some initialization steps such as the initial object location which can be provided manually or automatically by using an object detector.

Kalman filters, although used in numerous other applications, can be used for object tracking too. The filters are very useful to track objects whose motion is known and this can be used to predict their position. Kalman filters are useful in reducing the contribution of measurement noise and process noise over time to get the position of the object.

2 Theory

2.1 Object Segmentation and Recognition

Objects are distinguished from each other in an image based on their characteristics. Efficient object recognition algorithms use multiple characteristics together to distinguish between different objects. There are many difficulties in object recognition like the image is a 2-dimensional representation of the 3-dimensional world, therefore objects can be occluded. The frame may not have the same constant background, which means that the object recognized in one frame may not be recognized the same way in the next frame. Objects are usually recognized with respect to a database, or a priori knowledge of the object and this database will not contain all the objects in the world. Therefore, considering all these difficulties, object recognition is still an ongoing research subject and many complex algorithms are being created (1).

The object segmentation and recognition used in this project is a very simple one based on color segmentation. The implementation of this is explained in further detail in Section 4.1.

2.1.1 Gaussian Noise Smoothing

The Gaussian kernel, also called as the Gaussian blur, is the convolution of the image with a Gaussian function (2). It is very commonly used in image processing and computer vision to remove the high frequency noise in the image, but it does however reduce the details in the image. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen. A Gaussian smoothening function acts as a low pass filter over the image. The horizontal and vertical smoothening can be done independently and the result is the same as 2-dimensional smoothening.

2.1.2 HSV color space

HSV stands for Hue-Saturation-Value, and is a cylindrical color space model representing the corresponding RGB colors (3). However the implementation of these models are not standardized, and there can be many different cylindrical models combined in different implementations. In computer vision we want to separate color components from intensity for various reasons, such as robustness to lighting changes, or removing shadows. Unlike RGB, HSV separates luma, or the image intensity, from chroma or the color information. This is very useful in many applications.

2.1.3 Morphological Filters

The morphological filters analyze the shape and geometry of the object (4). These methods are one of the most widely used classes in Image Analysis and have numerous applications in many fields. They are very simple to implement and mainly work on binary images. Erosion Morphological filter works by finding the infimum in the defined area and replaces all the pixels with this value. Whereas the Dilation Morphological filter works by finding the supremum in the defined area and replaces all the pixels with that supremum value.

2.2 Kalman Filter

The Kalman filter is over 50 years old but is still one of the most important and common data fusion algorithms in use today. Success of the Kalman filter is due to its small computational requirement, elegant recursive properties, and its status as the optimal estimator for one-dimensional linear systems with Gaussian error statistics. Kalman filters let you use mathematical models despite having error-filled real-time measurements (5). Typical uses of the Kalman filter include smoothing noisy data and providing estimates of parameters of interest (6). Applications include global positioning system receivers, phaselocked loops in radio equipment, smoothing the output from laptop trackpads, and many more. The Kalman filter is typically derived using vector algebra as a minimum mean squared estimator. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown (7).

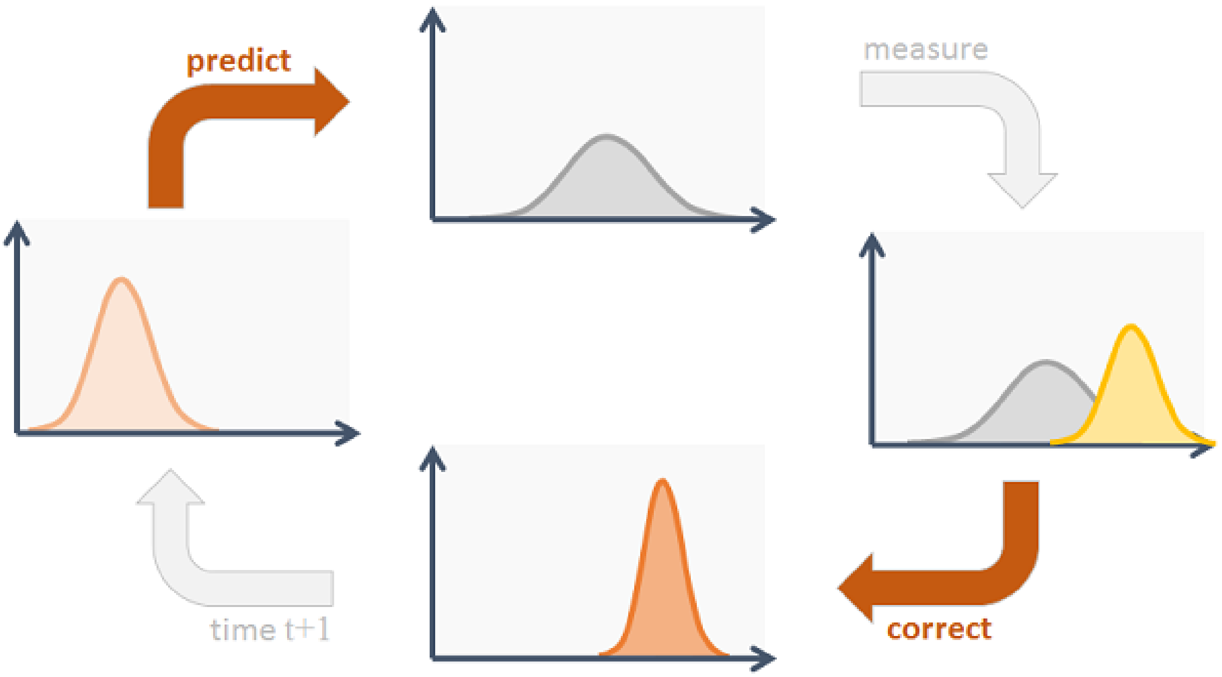


FIG 1. Kalman filter cycle

Praharsha Sirsi (s8prsirs@stud.uni-saarland.de) (2557724)
Shruthi Yaddanapalli (shruthiyaddanapalli@gmail.com) (2551067)

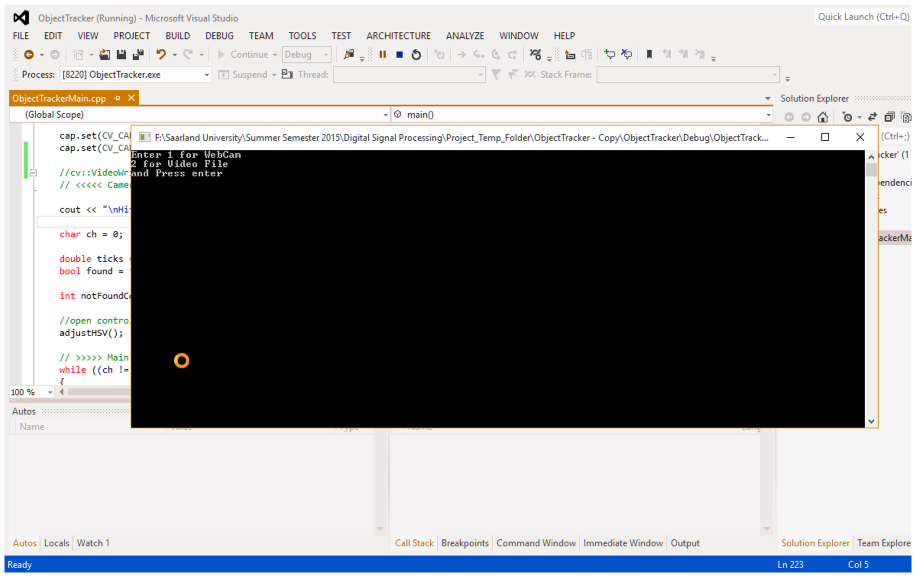
Much of what the Kalman filter does can be reduced to propagating and updating Gaussians and updating their covariances. First the filter predicts the next state from the provided state transition (e.g. motion model), then if applicable, the noisy measurement information is incorporated in the correction phase. The cycle is repeated (8). They are especially convenient for objects which motion model is known, plus they incorporate some extra information in order to estimate the next object position more robustly. They can be used for general purpose single object tracking assuming some constraints.

If we have a linear motion model, and process and measurement noise are Gaussian-like, then the Kalman filter represents the optimal solution for the state update (in our case tracking problem). Those conditions are satisfied for a vast majority of applications.

3 Using the software

3.1 Initial Setup

- Please copy the entire folder, and open the "ObjectTracker.sln" file in Visual Studio.
- Hit "Ctrl+F5" to run the program.
- The following screen will appear



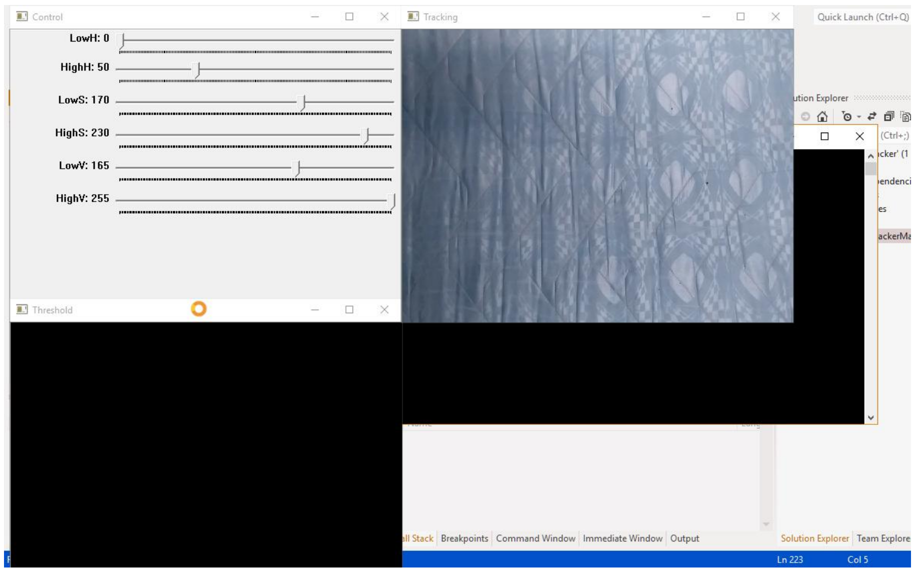
ObjectTracker (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [8220] ObjectTracker.exe
ObjectTrackerMain.cpp
Solution Explorer
F:\Saarland University\Summer Semester 2015\Digital Signal Processing\Project_Temp_Folder\ObjectTracker - Copy\ObjectTracker\Debug\ObjectTrack...
Autos Locals Watch 1
Call Stack Breakpoints Command Window Immediate Window Output
Ready Ln 223 Col 5

```
Enter 1 for WebCam
2 for Video File
and Press enter
```

FIG 2. Initial Run

3.2 Using WebCam for tracking

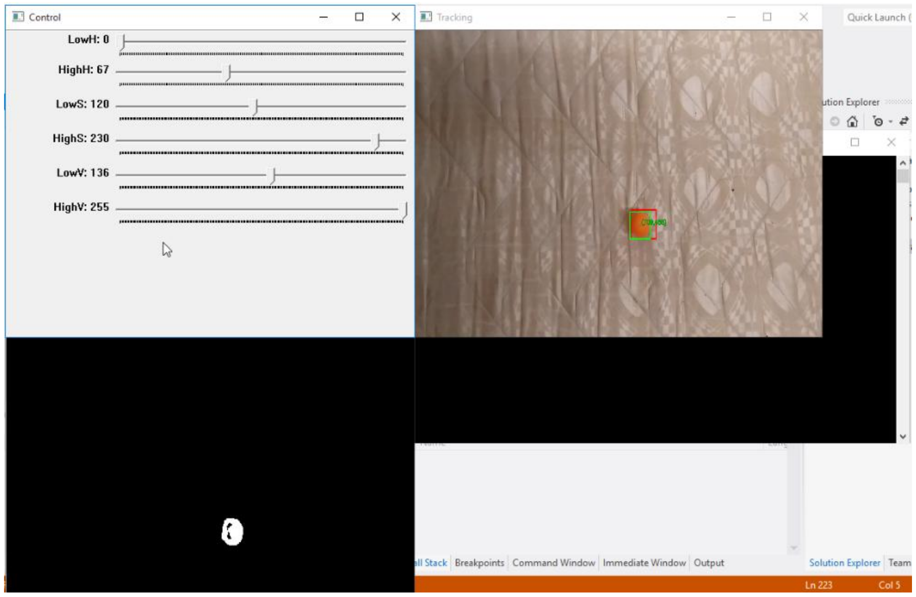
- Press "1" and hit enter after the window in FIG 2 appears.
- The WebCam is turned ON and the video from the webcam is taken as input for the program.
- 3 Windows will appear called "Control", "Threshold" and "Tracking" as shown in FIG 3



LowH: 0
HighH: 50
LowS: 170
HighS: 230
LowV: 165
HighV: 255

FIG 3. 3 Window Pop-Up

- Adjust the HSV bars in the "Control" window to filter the object to be tracked. The result of this filtering is shown in the window "Threshold".
- Once the threshold parameters are set so that only the object to be tracked is visible in the threshold window, move the object.
- The detected object will be highlighted by a green box around it, and the Kalman filter predictions is displayed by the red box (Notice the white ball structure in "Threshold" window).
  *Stop Execution in Visual Studio once done

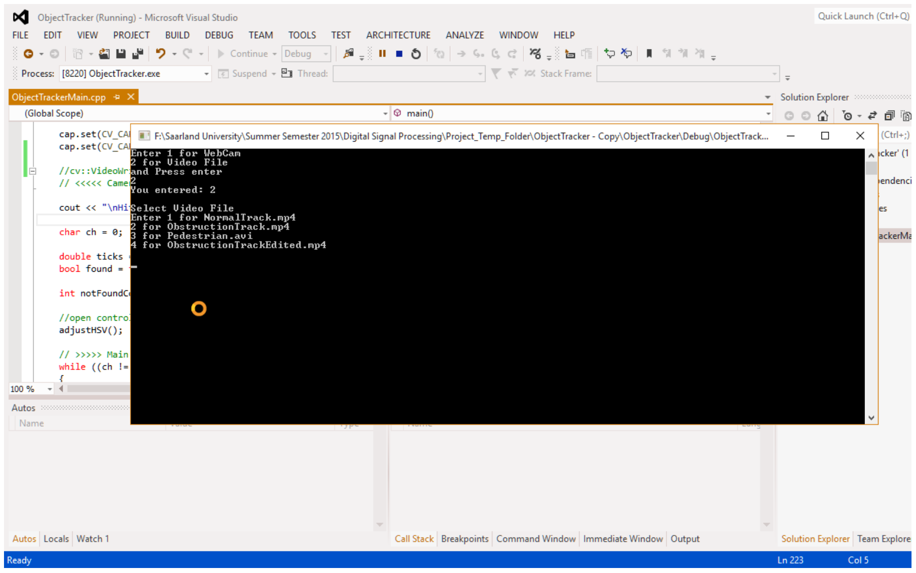


LowH: 0
HighH: 67
LowS: 120
HighS: 230
LowV: 136
HighV: 255

FIG 4. Tracking Example

3.3 Playing Test Videos

- Press “2” and hit enter after the window in FIG 2 appears.
- The message in FIG 5 will be displayed.
- Enter the number according to the file and then hit enter to take that particular video file as input for the program.
  *The HSV values are pre-stored with respect to the video file to give optimum object filtering. However, if needed, they can be changed using the “Control” window
  *The video files run in a loop until the program execution is stopped
  *Stop Execution in Visual Studio once done


ObjectTracker (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [8220] ObjectTracker.exe
ObjectTrackerMain.cpp
Solution Explorer
F:\Saarland University\Summer Semester 2015\Digital Signal Processing\Project_Temp_Folder\ObjectTracker - Copy\ObjectTracker\Debug\ObjectTrack...
Autos
Call Stack Breakpoints Command Window Immediate Window Output
Ready Ln 223 Col 5

```
Enter 1 for WebCam
2 for Video File
and Press enter
2
You entered: 2

Select Video File
Enter 1 for NormalTrack.mp4
2 for ObstructionTrack.mp4
3 for Pedestrian.avi
4 for ObstructionTrackEdited.mp4
```

FIG 5. Select Video File

4 Implementation

4.1 Object Recognition

Object filtering and recognition is achieved based on color difference between the object and the background. The HSV values in the “Control” window are used as thresholds for filtering. The HSV color space is most suitable for such color based segmentation. However, the thresholds need to have lower and upper bounds to take into consideration the varying illuminations of the object. The object segmentation is poor in dimly lit environments. Following are the steps taken for each frame in the video for object recognition.

- Each frame is read as an RGB image
- The image is convolved with a Gaussian kernel of size 5*5 pixels and sigma = 3. This act as low-pass filters over the entire image and smoothens out the noise. Larger Gaussian kernels with higher sigma result in more smoothening, but reduce the information of edges. If the kernel is too small, then noise is not smoothened. Hence an optimum Gaussian kernel needs to be used.

- After noise smoothening, the entire image is converted to the “Hue-Saturation-Value” color space from the RGB color space. The HSV is more suitable for such color based filtering because representation of color is more natural. The color component is separated from luminosity in HSV, which helps in segmentation based on color under illumination changes and shadows.
- The predefined HSV values, or the values from the “Control” window, are used as lower and upper bound thresholds. All HSV values within these thresholds are set to 1, whereas all other HSV values outside these thresholds are set to zero. This results in a binary image consisting of only black (0) and white (1) values.
- The binary image can have false alarms of white pixels in the frame. To remove these we apply simple morphological filters. Morphological Opening filter is done by first eroding and then dilating using the same filter kernel. There is also a possibility that black pixels might be present that are surrounded by white pixels. We can filter out these black pixels by applying a Morphological Closing filter. This is achieved by first dilating and then eroding the binary image with the same kernel.
- After applying the morphological filters, the valid contours are calculated by connecting all the edge pixels of the filtered object. Only closed contours are found. This step can be improved by using a Canny Edge detector before finding the contours. All the points of the contour are stored as a vector of points. The contour is just the outer line of the filtered object in the binary image.
- For each of the contours that are found, a bounding box is calculated that inscribes the contour within the rectangle. The rectangles whose area is less than 256 pixels are ignored. This kind of hard filtering based on area is undesirable in real world object recognition. Hence, more complicated methods exist, and the methods are being improved actively. For example, “GoogLeNet” uses a huge database and neural network to detect and recognize objects.
- Once the object is found whose area is greater than 256 pixels, a green box is drawn around it with the centroid of the rectangle which indicates the position. The contour that is inscribed within the rectangle is also drawn just for reference.

The method used in this implementation for object segmentation is very simple and is not very robust to be used in real world applications. Methods like “Speeded Up Robust Features (SURF)”, Homography, Face detection using Haar Wavelets, and many more are used to get better results in object recognition and segmentation.

4.2 Tracking using Kalman Filter

The Kalman filter implemented is a simple Kalman filter (7). We make use of a Discrete and Linear Kalman Filter without the optional control input. The Kalman filter is used to estimate the position of the object by taking into consideration the previous measurements. Measurement noise and process noise are simulated. The centre of the rectangle and size of the rectangle after object recognition is used as input for the Kalman filter. Following are the steps taken in implementing the Kalman filter.

- To represent state of the object, we use 6 parameters. The centroid of the bounding rectangle gives the position values of x and y. The size of the rectangle gives the width and height of the rectangle. Velocity of the object in x and the velocity in y is calculated with respect to the time elapsed which is given by clock ticks in the program. The velocity representation here may not be the absolute velocity of the object in the real world; however it is a fair representation when the velocity is linear, as we are mainly interested in the relative change rather than the absolute values.

- The measurements, however, are only the position (x, y) and size (width, height) of the bounding box of the object. So there are only 4 measurements, but 6 parameters are used to represent the state of the object. The optional control matrix (or matrix B in (7)) and the control input is used mainly to simulate the physical process under consideration. But in our implementation, we have no knowledge of the control system under consideration or the user control that should be applied. In finely tuned Kalman filters that are used in practical applications, acceleration limits can be set in these matrices along with other control factors.
- Transition matrix is the relation between the previous position and the present position of the object. The transition matrix is initialised assuming that the object moves in a linear way. However this assumption might not hold true for many applications. This matrix is denoted as A.
- The measurement matrix H denotes the values to be considered when a new measurement of the position of the object occurs. This matrix is multiplied with the state vector to give the measurement vector.
- The estimated process noise covariance matrix, denoted as Q, is a measure of the noise introduced by the process on the predicted state of the object. For fine tuning of Kalman filters, experiments are performed to determine the exact values to be filled into the matrix. The values can change over time in a real process. In our implementation, we assume certain values.
- The measurement noise covariance matrix, denoted as R, is a measure of noise introduced by the sensor, or the webcam in our implementation. Again, for fine tuned Kalman filter we use data from experiments and 'Data Sheet' of sensors to fill the matrix. But, in our implementation, we again assume very low values.
- Whenever an object is found for the first time, the filter is initialised with the predictor equation containing the time update algorithm. Here we assume an error of 1pixel in the predictor equation initially. The error then converges to a small value as long as tracking of the object is continuous. The measured position and size of the box is updated as state of the object in the Kalman filter. This is done as initialisation step for the filter.
- After the object is found and the initialisation of the filter is done. Time difference (Δt) is updated in the transition matrix of the filter for every iteration. Smaller time difference between each iteration results in better filter predictions and in general better tracking. The Kalman filter is used continuously to predict the position of the object by using the "predict" method in the program. These predictions are drawn as red boxes in the output video. All new measurements are updated in the Kalman filter using the corrector equations defined in the "correct" method of the filter. This process of predicting and correcting occurs in a loop as long as the object exists.
- If the object is obstructed or is lost from tracking process for more than 100 frames, the tracking is stopped. But if the object is detected again, then tracking is re-initialised. The 100 frames is just a best estimate, and should not be considered as standard value for any application. In

practical applications of tracking, there are models used which simulate the behaviour of the objects, and have rough estimate of how long the predictions of the Kalman filter can be trusted. For example, a pre-defined maximum error or a time window is defined depending upon the speed of the object and the update rate

The Kalman filter implementation here is for only for a single track. It can be extended by initialising multiple filters for each detected object to achieve multiple object tracking. However the handling of object collisions, cross-over, overlap, and sensor resolution to distinguish closely spaced objects has to be taken into consideration. This is achieved in Object tracker algorithms which can either pre-filter the data before updating the Kalman filter, or they can post-filter the data while handling instances of the Kalman filter for each track.

5 Results

5.1 Normal ball tracking

A normal video was made where a ball moves across the area containing a fairly constant color background. This video was fed to the program by selecting the Video File 1 (Refer FIG 5). The HSV colors for object recognition were calculated offline and fed into the program. The ball was tracked continuously along with the predictions of the Kalman filter. The prediction filter had undershoots initially (as shown in FIG 6) trying to catch up to the ball which is as expected. Once the ball decelerated quickly on the surface, the Kalman filter had overshoots (as shown in FIG 7) which is due to the linear velocity model that we have assumed. However once the velocity stabilized the predictions coincided with the measurements closely (as shown in FIG 8).

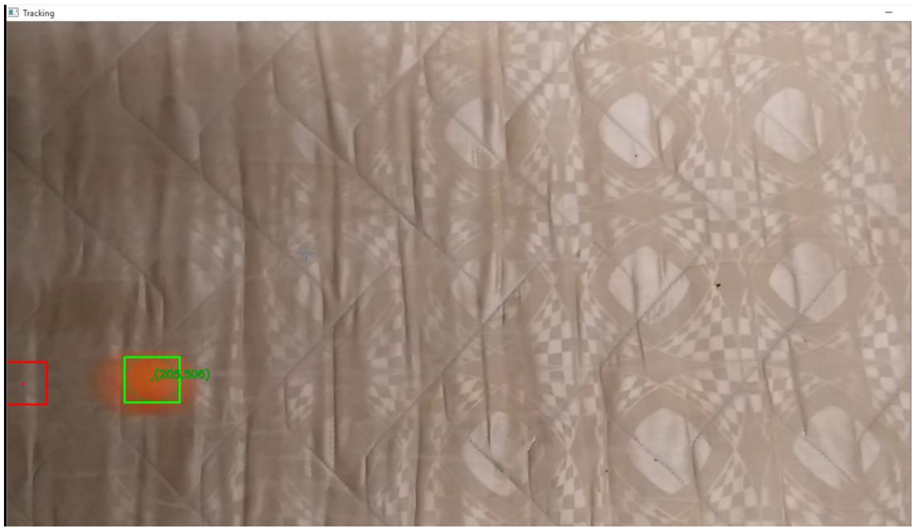


FIG 6. Undershoot while tracking

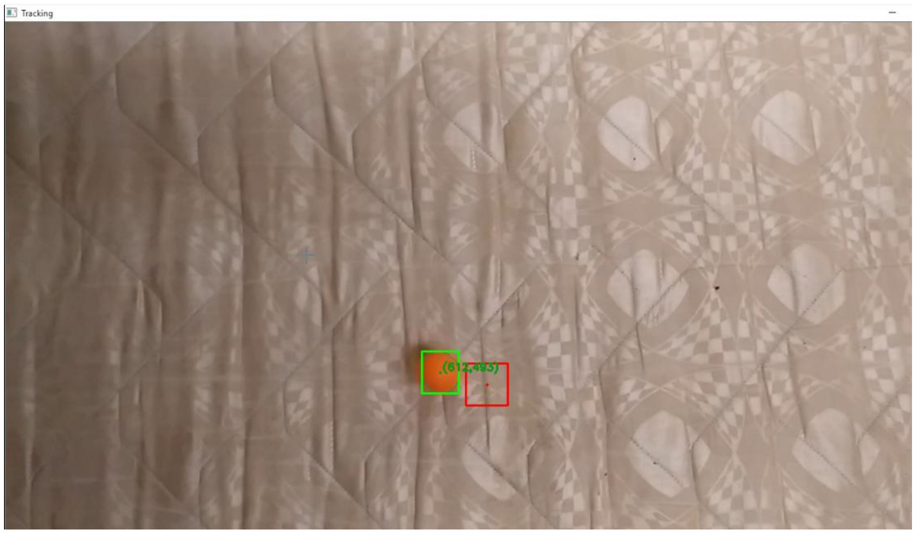


FIG 7. Overshoot while tracking

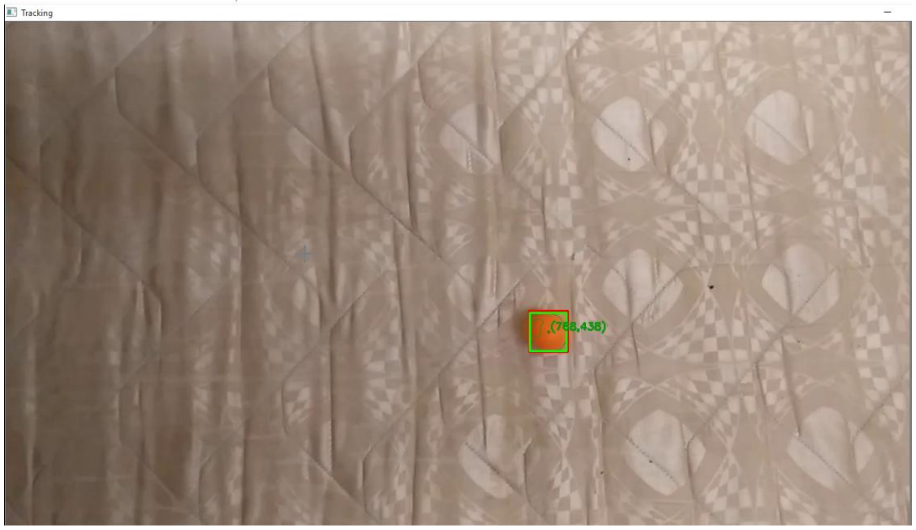


FIG 8. Normal stabilized Track

5.2 Ball tracking with Obstruction

5.2.1 Original video file

An obstruction to the path of the ball was introduced to check the performance of predictions of the Kalman filter. This video was fed to the program by selecting the Video File 2 (Refer FIG 5). The initial track is fairly good (as shown in FIG 9) with a slight error, as the ball is still decelerating. Once the object is behind the obstruction, the Kalman filter is predicting the position of the object with no measurement updates (as shown in FIG 10). The model used for Kalman Filter is a linear velocity model; hence the predictions follow the path of linear velocity (as shown in FIG 11). But the ball is constantly decelerating under the obstruction and hence the predictions are not close to the actual position of the ball. With no new measurements to correct the filter under obstruction, the Kalman filter predicts that the ball is about to leave the frame. However the ball is just about to appear out of the obstruction (as shown in FIG 12).

Once the ball measurement occurs, the Kalman filter immediately corrects and starts tracking after the obstruction (as shown in FIG 13). This shows that the linear velocity model is not sufficient to track objects in real world. If an Object Tracker algorithm is used, it would not associate the same track identity if the predictions of the object were that far away from the new measurement. The tracker would initialize a new Track Identity once the measurement after obstruction occurred and delete the previous track. Hence, we can see that there is a need to model the physical process under consideration for tracking objects in real world applications. The Kalman filter is usually implemented with two different models, namely "Constant Velocity" and "Constant Acceleration". In advanced trackers, the model is changed between the two depending on measurements.

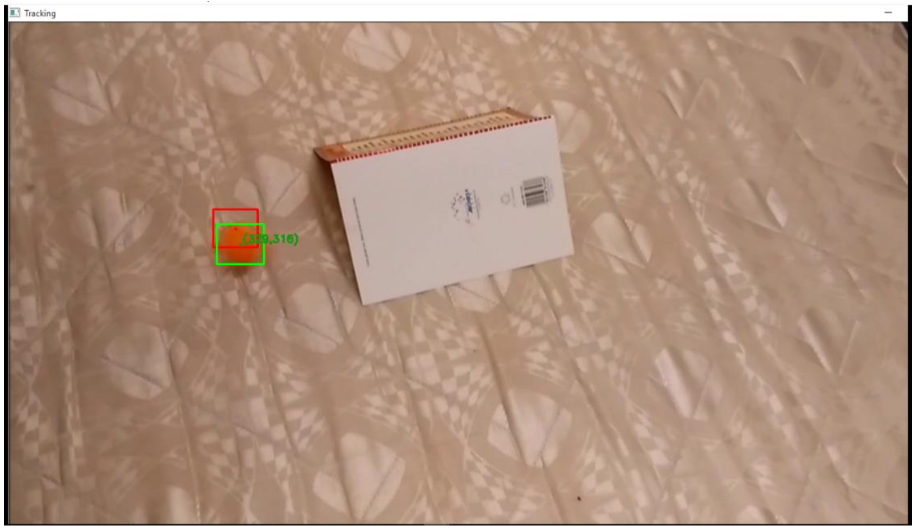


FIG 9.   Track before obstruction

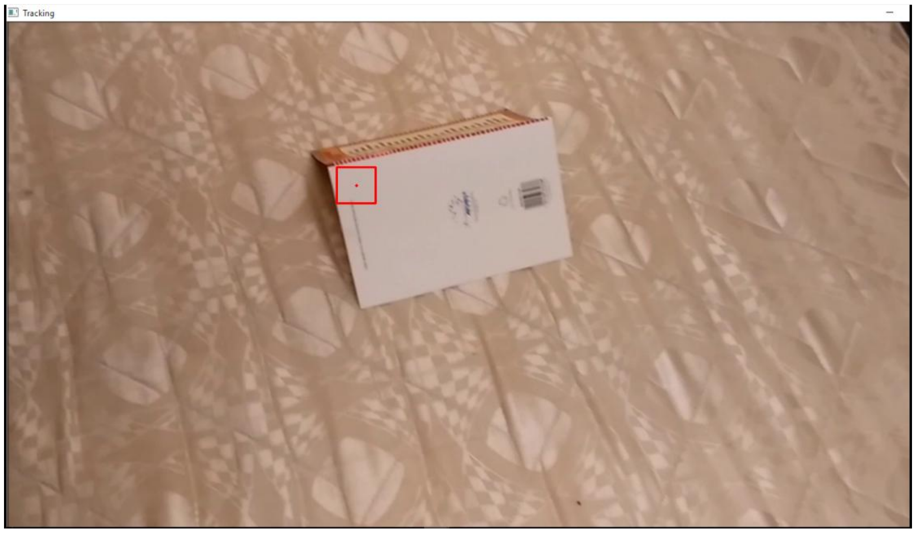


FIG 10. Track entering obstruction

Praharsha Sirsi (s8prsirs@stud.uni-saarland.de) (2557724)
Shruthi Yaddanapalli (shruthiyaddanapalli@gmail.com) (2551067)

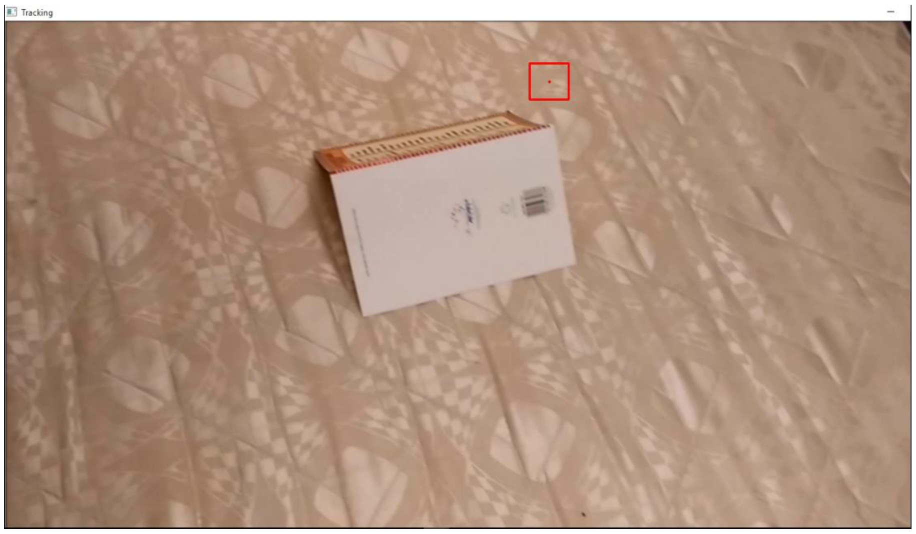


FIG 11. Linear velocity predictions

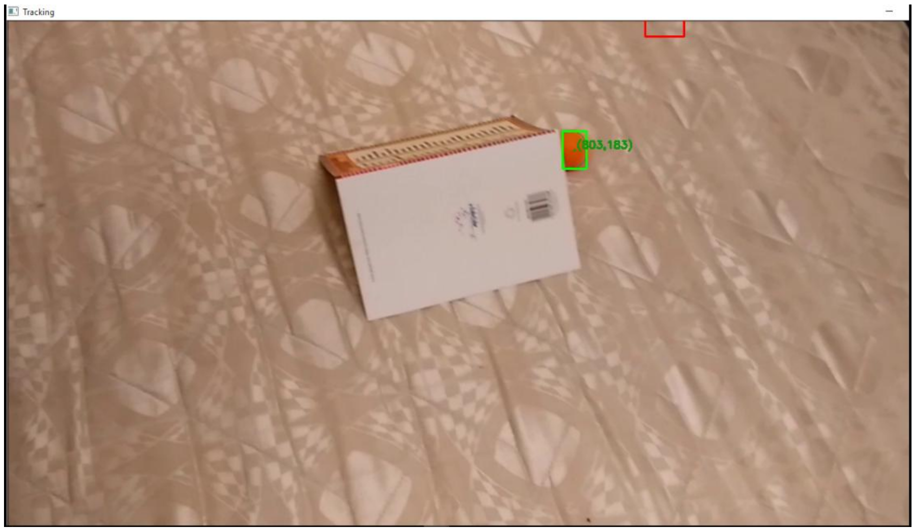


FIG 12. Object appearing out of the obstruction

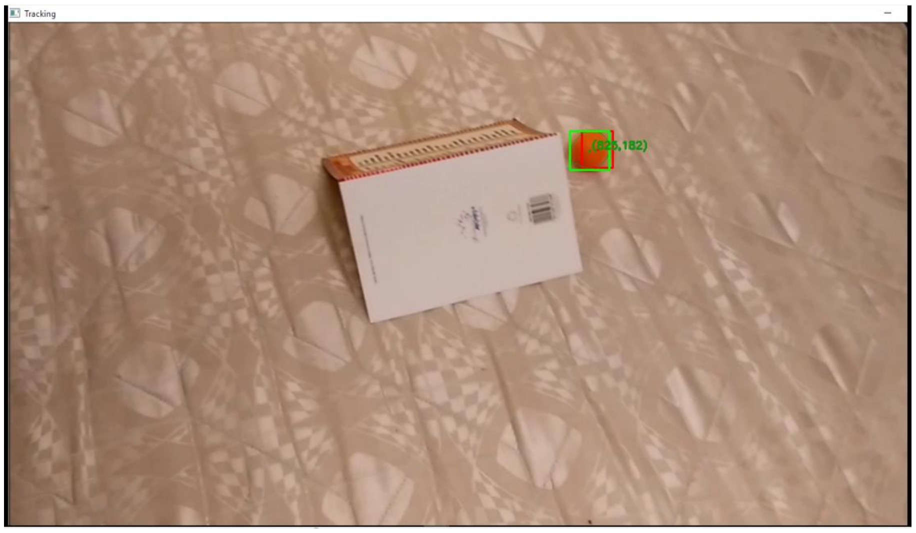


FIG 13. Track after obstruction

5.2.2 Edited video file

Changes were made to the video file by increasing the playback speed by 2 times and also removing frames when the object is behind the obstruction. This video was fed to the program by selecting the Video File 4 (Refer FIG 5). This is done to check how the filter behaves with faster updates and lesser frames with no measurements. The result shows that the predictions were fairly close to the path of the ball even under obstruction (as shown in FIG 14 and FIG 15)

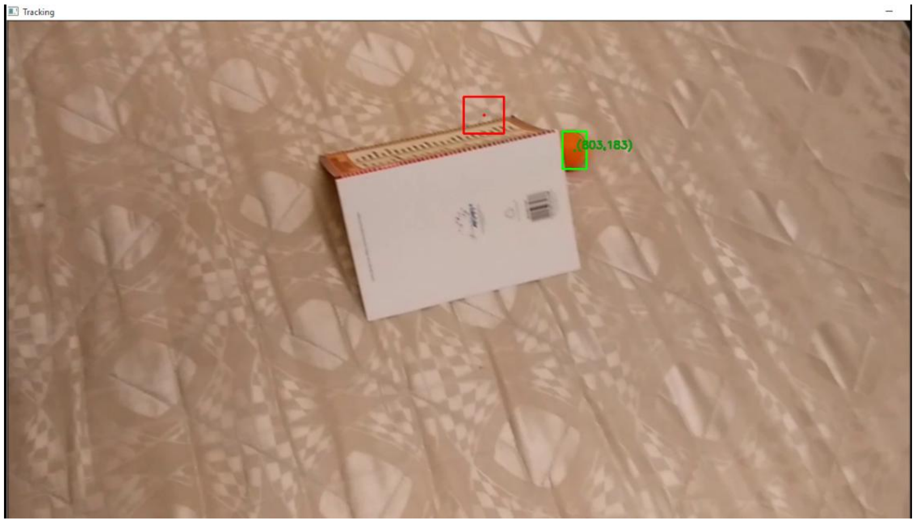


FIG 14. Faster update tracking under obstruction

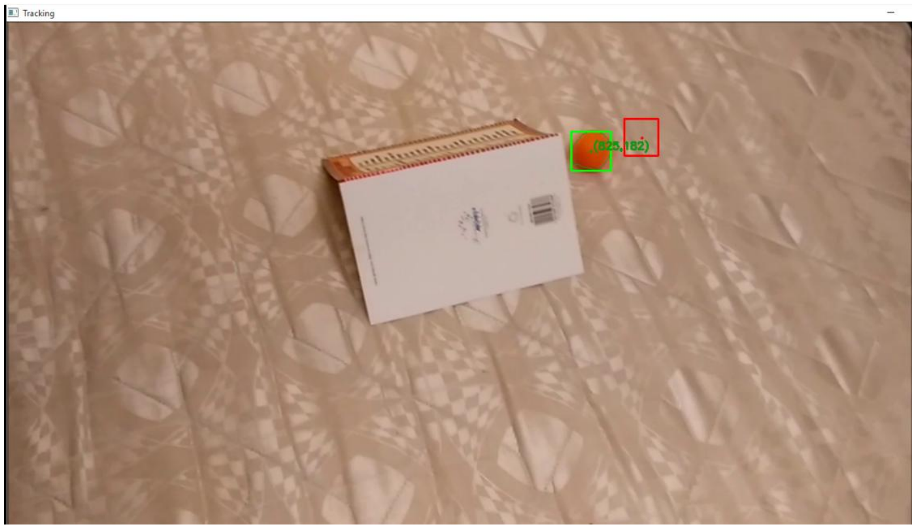


FIG 15. Faster update tracking after obstruction

5.3 Real world video

A video of real world of pedestrians crossing the road at a traffic signal was used as input for the program to check its performance. The Video File 3 was selected as input for the program (Refer FIG 5). The filter and the program do fairly well even under these conditions by tracking the person of interest (as shown in FIG 16). However the limitations of the implementation are clear, as the HSV filtering is recognizing multiple objects in the frame. More robust object segmentation and recognition methods are required to track only the person in the frame. As the person moves far away the area of the box reduces and hence our implementation of hard filtering according to area doesn't recognize the person as valid object anymore and shifts the tracking to another stationary object close to the person (as shown in FIG 17)

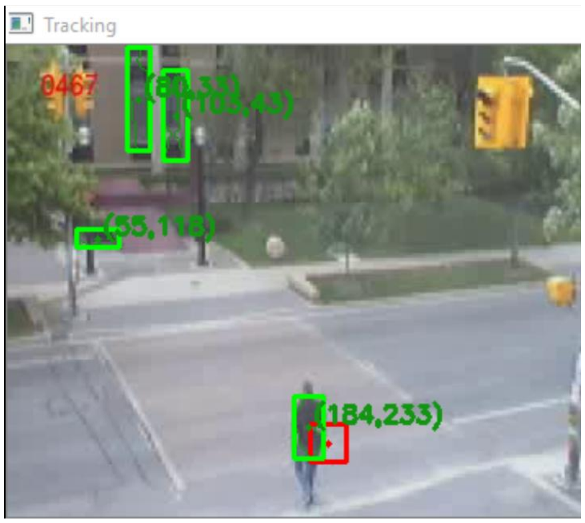


FIG 16. Initial Tracking of a Person

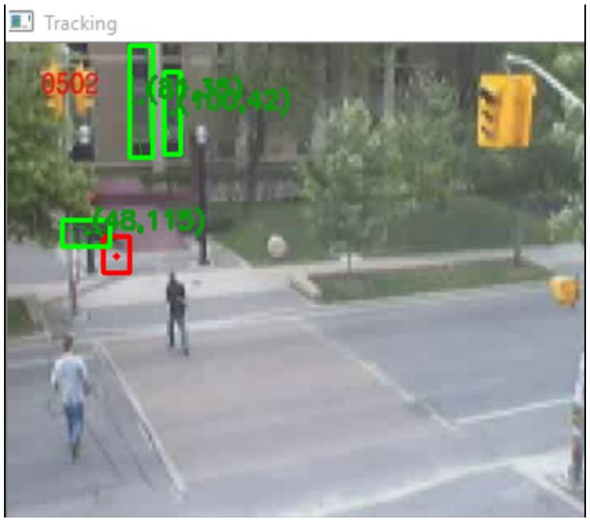


FIG 17. Area of person too small for recognition

Praharsha Sirsi (s8prsirs@stud.uni-saarland.de) (2557724)
Shruthi Yaddanapalli (shruthiyaddanapalli@gmail.com) (2551067)

6 Works Cited

1. **Szegedy, Christian.** Building a deeper understanding of images. [Online] September 5, 2014. http://googleresearch.blogspot.co.uk/2014/09/building-deeper-understanding-of-images.html.

2. **Wikipedia.** Gaussian blur. *Wikipedia.* [Online] Wikipedia. https://en.wikipedia.org/wiki/Gaussian_blur.

3. —. HSL and HSV. *Wikipedia.* [Online] Wikipedia. https://en.wikipedia.org/wiki/HSL_and_HSV.

4. —. Mathematical morphology. *Wikipedia.* [Online] Wikipedia. https://en.wikipedia.org/wiki/Mathematical_morphology.

5. **Faragher, Ramsey.** Understanding the Basis of the Kalman Filter. [Online] BAE Systems Advanced Technology Centre, United Kingdom, SEPTEMBER 2012. http://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf.

6. **Czerniak, Greg.** Greg Czerniak's Website. [Online] http://greg.czerniak.info/guides/kalman1/.

7. **Welch, Greg and Bishop, Gary.** An Introduction to the Kalman Filter. [Online] July 24, 2006. https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.

8. **Jurić, Darko.** Object Tracking: Kalman Filter with Ease. [Online] codeproject, Jan 15, 2015. http://www.codeproject.com/Articles/865935/Object-Tracking-Kalman-Filter-with-Ease.