Lab 3: Kalman Filtering for Object Tracking

Iuliia Alekseenko and Ricardo Luque

I. GENERAL COMMENTS

Tracking methods have as a main motivation the recognition and localization of objects in a scene, so that the object detection can be more robust to occlusions scenarios, lightning changes, and possible background modifications. Tracking algorithms are performed after obtaining objects of interest, and they work in such way that they monitor these objects through frames in a video sequence. Therefore, this lab assignment has as a main purpose, showing the implementation of **Kalman Filter** using two different models: a *constant velocity model* and a *constant acceleration model*. Some tuning is performed upon each video sequence in both, **Kalman Filter** parameters and **image segmentation** so that a correct object tracking is performed.

The mentioned tuning can be found in the script as comments, mentioning the values to change for every video sequence to achieve an optimal tracking.

II. INTRODUCTION

As we have seen in previous lab assignments, the implementation of object identification algorithms has been of wide interest due to their nature of providing information about relevant parts of scenes. In fact, these objects of interest (usually extracted using Blobs[1]) can be obtained from background subtraction methods such as dynamic, static and probabilistic ones.

After obtaining such information regarding the observed scenes, it is important to mention that several problems can arise from using an object identification approach solely. These issues may include the identification of objects that do not provide any useful information about relevant objects, weakness towards partial occlusions, and some more. However, they can be corrected through tracking methods such as particle filters or **Kalman Filtering**.

In general, tracking refers to a process of propagating a corrected state prediction based on a prior prediction and a prediction correction obtained through the measurements acquired in the scenes in context. And since tracking uses a state prediction to do so, it works through different frames in video sequences, with the aim of identifying an object or objects of relevance through *time*. In other words, tracking works in the way that it improves the detection of an object by monitoring it through the video sequence.

Having that, this report aims to explain the implementation of the Kalman Filtering approach through different video sequences.

III. METHOD

The *tracking* approach used in this laboratory session is the **Kalman Filter** which supports two models: *constant velocity* and *constant acceleration*. The algorithm implementation follows after the pre-processing steps mentioned below:

- Built-in OpenCV MOG for foreground detection;
- Morphological opening in order to filter noise from the foreground mask
- Center extraction of the biggest blob detected on the foreground mask

Since these points are not the part of the **Kalman Filter** implementation per Se, they will not be developed throughout the methodology. However, they will be briefly mentioned in the implementation (IV), as they influence the tracking performance.

A. *Kalman Filter*

The **Kalman Filter** is probably one of the most popular filtering algorithms used in many fields. Due to its simplicity and effectiveness, it is applied in applications such as GPS, robotics, control systems, and more. This tracking method takes into consideration a dynamic model of the system (e.g. the laws of motion), sequential measurements and possible corrections to improve the state estimate.

The algorithm consists of two repeatable steps:

Prediction and ***correction (update)***. First, the prediction of the next state is calculated while taking into account measurements inaccuracies. Second, a prediction correction is performed, based on the difference between the measured and the predicted output. For instance, new information from a sensor corrects the predicted values while also taking into account the inaccuracy and noisiness of these inputs.

*1) **Constant Velocity**:* For the constant velocity model, it is assumed that the velocity value of our object of interest does not change during tracking. In this model, the only variables to be considered are the position of the abject along x and y axes, and their speeds in both, x and y directions.

[1]A **Binary Large Object** (BLOB) refers to a collection of binary data which contains an identified object in the scene

*2) **Constant Acceleration**:* For the constant acceleration model, on the other hand, we assume that the velocity, and position of the objects of interest change, whereas their accelerations remain constant. Here we will include the state vector variables mentioned above ($position_x$, $position_y$, $velocity_x$ and $velocity_y$) and two extra variables will be added: $acceleration_x$ and $acceleration_y$.

IV. IMPLEMENTATION

The first step of the tracking process is substracting the background of the input video sequence. The implemented method is based on the createBackgroundSubtractorMOG2 function with such set parameters as *HISTORY*, *VARTHERSHOLD* and detection of shadows. Also, the learning rate parameter *LEARNING_RATE* is used when applying the method. The logic is implemented in the *bkg_subtraction* function.

Later, the morphological opening operation is used in order to filter out noise regions on the foreground mask. The parameters are *type = MORPH_RECT* - element shape, *KERNEL* - size of the structuring element, and *Point(1,1)* - anchor position within the element. The logic is implemented in the *MorphologicalOpen* function.

The next step is blob detection using the built-in *connectedComponentsWithStats* function which computes the connected components labeled image and also produces a statistics output. The parameter of the algorithm is *connectivity*. Also, some post-processing logic is implemented in order to save blobs bigger than set *MIN_HEIGHT* and *MIN_WIDTH*, as well as overcome the problem when all frame is detected as a blob. Each blob is saved with its parameters to *bloblist* and the center of each blob is saved to *blob_centers*. The logic is implemented in the *extractBlobs* function.

All functions described above are implemented in *FgSegment.cpp*.

When a single-object (blob) is detected, the **Kalman Filter** is applied for tracking. The algorithm is implemented in *Kalman.cpp*.

First, depending on the model used, the state matrix is initialized (either 4*1 or 6*1), the measurement matrix is the same in both cases (2*1). Later, the **Kalman Filter** variables are initialized, this is, all the matrices required for this algorithm. The KalmanFilter OpenCV class is used for this purpose. The matrix are the state transition matrix, process noise covariance matrix, measurement matrix, priori error estimate covariance matrix, measurement noise covariance matrix. The first three matrix are set in a different way for each of the model implemented, while the last two are the same in both cases.

When the first detection happens, the target state is initialized. The center coordinates of the detected blob are taken as the previous state coordinates, which will be used for the prediction. Later, the values will be called by the **predict** function to compute a new prediction.

Then, the center of the object will be sent to the *measurement* value, which will become the input of the correct. In its turn, it updates the prediction and measurement, thus resulting in the final state value.

The logic behind two models - constant velocity *constantVelocity* and acceleration *constantAcceleration*, is the same, but the difference is the size of the matrix operated.

V. DATA

In total there are 8 video sequences for task 3.2 and task 3.3. While videos in 3.2 represent quite simple situations - toy examples, videos in 3.3 represent real life scenarios. One more baseline video is given for testing the Kalman Filter.

A. Baseline video - singleball

This single ball video is a good start to evaluate the work of the Kalman Filer algorithm since it is a simple scenario - single-color black background, good contrast between foreground and background (ball color is bright), and simple hindrance. The ball goes through a box, and in some frames the ball is not seen (no measurements of the ball location). The video sequence helps to understand how good the algorithm predicts the position, and how close it is when the ball.

B. Task 3.2

video2 and *video3* both have relatively simple background, but has the problem of reflecting the ball from the table surface which affects the blob centers values. Also, in

video3 the motion of the ball has quite challenging trajectory - it moves to the wall and then returns back.

video6 is similar to first two videos in some since as shows the horizontal motion. Although the scene looks complicated (for instance, the small size of the ball, dark light conditions), foreground is segmented properly which makes tracking easier.

video5 has a vertical motion opposite to the previous sequences, and this video has a great noise which makes the foreground mask is difficult to work with.

Also, one of the main challenges in these videos is that motion is fast in some frames, which makes the object's shape blurred.

C. Task 3.3

This set of videos is more complex than previous toy sequences as they have more challenges which affect prediction and requires careful tuning for suitable results.

abandonedBox has the background with many details as well as the object makes a stop in some frames and later resumes the route (physical model describing the motion is complex).

boats has the main problem connected with the dynamic background, which makes the extraction of blobs challenging. At the same time, the boat has great difference between its weight and height (not like any other object to track in other sequences), thus this fact must be taken into account.

pedestrians is a video sequences with illuminance problems - all the frames are overexposed due to bright sunlight. Also, the object casts the big shadow which mistakenly can be detected as the object to tract.

streetCornerAtNight has the set of problems. First, noise in all frames due to quality of the camera as well as low light conditions. Second, the object motion is fast which smooths its shapes, as well as in front of the object there is light source.

VI. RESULTS AND ANALYSIS

A. *Task 3.2: Analysis of toy data*

The evaluation of the toy video sequences was performed in order to have a deeper understanding of some parameters and matrices involved in the Kalman Filtering updating process. In this model there are three main matrices that we have looked into. First, the observation matrix, which is in charge of measuring the amount of noise the observation is going to include into the model. In other words, how trustworthy the observations are given the fact that we modify any of its parameters.

The second one is the process noise, Q. Which has pretty much the same effect as the observation process, however if it decreases, then we trust the process more, and vice-versa. This means that if the process noise increases, we can trust it less, hence we can trust the measurement one more. Both matrices are in fact, always fluctuating. The process one; nevertheless, has decoded in itself information regarding the dynamic model of the system. In this scenario, it has information regarding the laws of motion and corrections, updates and states estimates are going to be performed taking this model as a starting point. The model, however can consist of the constant velocity model or the constant acceleration one, meaning they will either have information regarding ($position_x$, $position_y$, $velocity_x$ and $velocity_y$) or ($position_x$, $position_y$, $velocity_x$, $velocity_y$, $acceleration_x$ and $acceleration_y$) respectively.

We can observe the analysis in the following subsections.

*1) **Acceleration model Toy Example**:* We can observe a good behavior of the Kalman Filter, since it in fact tracks the image of interest, which is the green ball. In rough terms, the model estimate position is corrected with the observation obtained further. We can observe how even under occlusion scenarios the filter performs accurately. This can be seen in the following Figure 1 .

*2) **Constant Velocity Model Toy Example**:* It is observed that this model of the Kalman Filter works in a very similar way than the Constant acceleration model. This is probably because the acceleration can be dismissed, therefore it can be assumed that the velocity is constant, and that a value of 0 acceleration is given. What happens is that the correction step occurs in such way that the acceleration is shrinked to a very small value, close to zero. Therefore, the constant velocity model and constant acceleration model can be equivalent when the object in context has none or almost zero acceleration. This can be observed in the following Figure 2.

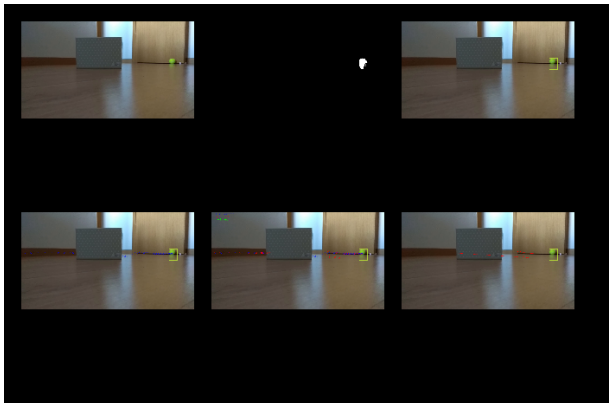

Fig. 1. **Constant Acceleration Model** Toy example with default values of R, Q, A matrices given in the slides

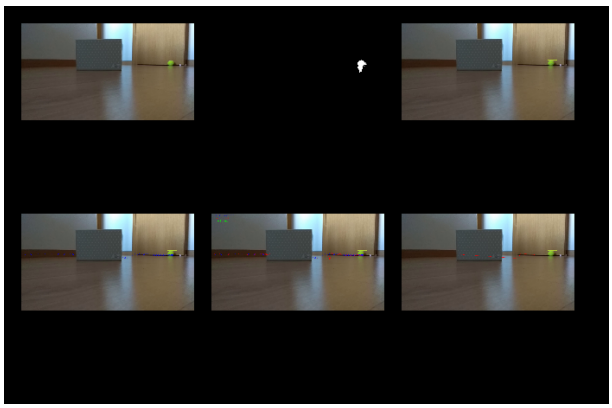

Fig. 2. **Constant Velocity Model** Toy example with default values of R, Q, A matrices given in the slides

*3) **Acceleration model Toy Example Modified R**:* Before analysing the images it is important to notice that the increment in values of the observation Matrix noise would mean that we trust our observations less. Therefore, it means that we trust our model more. Having mentioned this, is important to compare the same model under two different values of R. One when the observation matrix is trusted more, meaning the value is lower (When R=1). And another one: when the observation matrix is trusted less (almost equivalent to saying the process one is trusted more) meaning when (R=50). We can observe this in the following Figures 3 and Figure 4

We can observe how on Figure 3 the points of the prediction are more accumulated towards the observations (the blue dots), meaning that in fact the correction is performed upon the intuition that the observation matrix is more weighted for the update, and therefore more trustworthy.

On the other hand, we can see how on Figure 4 the points of the prediction not necessarily tend towards a point, in

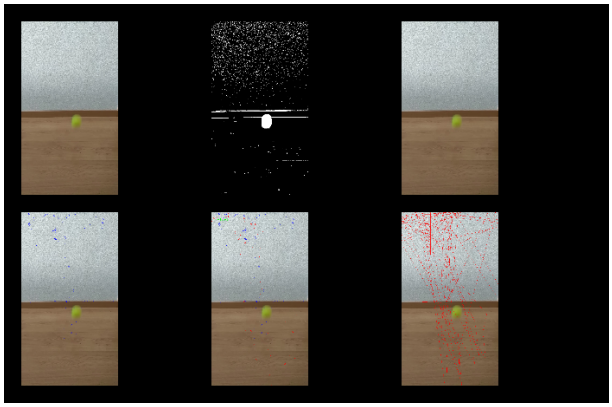

Fig. 3. **Constant Acceleration Model** Toy example with R values of 1

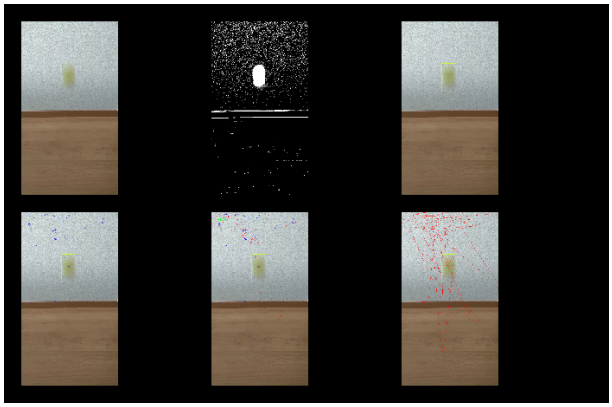

Fig. 4. **Constant Acceleration Model** Toy example with R values of 50

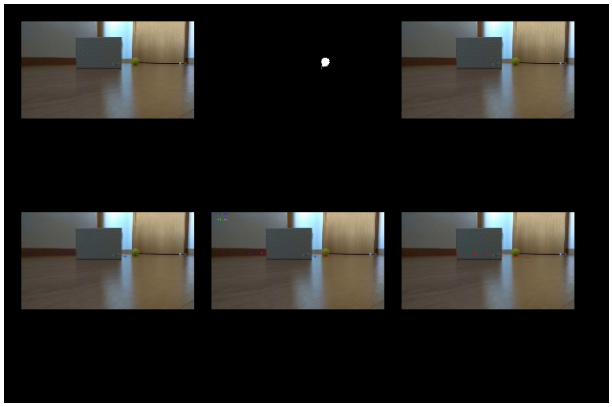

Fig. 5. **Constant Velocity Model** Toy example with R values of 1

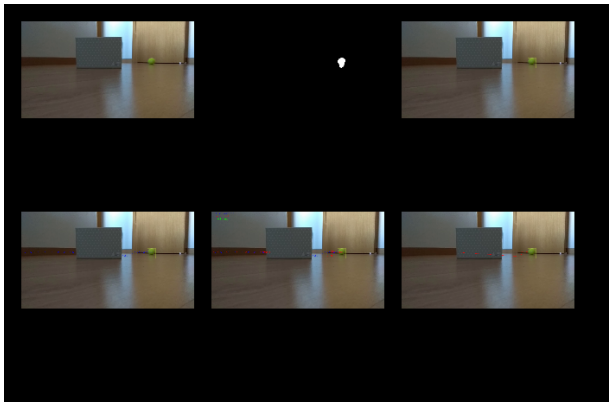

Fig. 6. **Constant Velocity Model** Toy example with R values of 50

other words, they do not tend towards the observations (the blue dots). This means the observation is not really relevant, at least not as much as the process itself. In other words, the observation matrix is not important for the update, and therefore less trustworthy.

*4) **Velocity model Toy Example, Modified R**:* In a similar way to the last subsection, we can observe the next two images for which a similar case scenario is happening. The first one with an R of 1 has a more trustworthy observation reason why we can observe how the values in fact approximate better to the ball after obtaining the blob location. In other words, observations are less noisy. This can be seen in Figure 5.

On Figure 6, on the other hand, it can be observed that regardless of the observations the prediction (bottom right image) do not follow the ball, therefore it's not been corrected by the observation, meaning that the observation is less trustworthy. In other words, regardless of the position of the ball, the prediction will keep the linear constant velocity model that was given.

B. Task 3.3: Analysis of real data

Since **Kalman Filtering** is strongly dependent on the blob extraction strategies, there is a big segmentation dependency. We have observed in the overall analysis that the better the blob extraction of the object of interest, the less tuning is needed for the Kalman Filter to do an accurate object tracking. Thus, complex scenarios work optimally even with standardized default Kalman's matrices values.

The tuning process is described in more depth aiming to explain the reason behind the variables' modification, and the way they improved each of the video sequences' tracking. We can observe this, in the following subsections, for which generic names of video sequences were given.

*1) **Abandoned Box**:* The first video sequence had good results with no modification of the toy value samples' matrices. For this, we used the constant acceleration model, since we assumed the bicycle decreased its velocity with the same acceleration.

Although this assumption was correct, we encountered that the Minimum and Maximum Blob size influenced the tracking quality. This happened since blob detections of different objects before identifying the actual object of interest (***cycling man*** in this case scenario) produced misleading predictions based on the observations. This is depicted in Figure 7 Tuning parameters such as the process noise would have not make much sense since the initial observations were wrong, and not trusting these observations would not have made sense either since they were the the

ones building the veracity of our model.

Fig. 7. Tracking results using initial ***Min_Width*** and ***Min_Height*** of 10 and 10 pixels respectively

However, after increasing the ***Min_Width*** and ***Min_Height*** values we can observe descent results for the whole video sequence, a snap of this is shown in in Fig. 8. It should be highlighted that the Kalman filtering approach can deal with process noise; however, if a model is built upon wrong observations, it will be a noisy model. Meaning, neither the observations nor the predictions can be trusted. Therefore, an initial good observation had to be performed, which is what was achieved by increasing the blob size to be detected.

Fig. 8. Tracking results using initial ***Min_Width*** and ***Min_Height*** of 20 and 20 pixels respectively

2) ***Pedestrians****:* Similarly to the last sequence, no modification of the toy sequences' values was performed. In fact the sequence worked perfectly for the same ***Min_Width*** and ***Min_Height*** values. We can point out that there are no misleading predictions nor observations when tracking the object of interest (***walking man*** in this sequence). This is depicted in Figure 9 , where a whole trace of the prediction and observation is shown. Since it seemed to work just fine, no tuning was performed, meaning we trusted equally the measurements and the predictions.

Furthermore, it is important to mention that the image has an almost unchanging background, and the only shadow is well identified and integrated in the background model. Therefore a correct foreground extraction is obtained and seen, in which almost all of it (expect for maybe a couple of pixels) correspond to the walking man, which is the **BLOB** used for tracking.

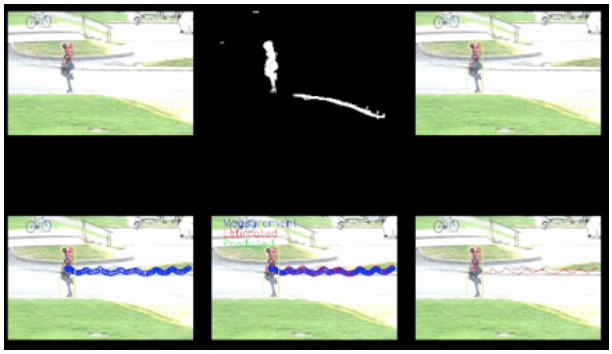

Fig. 9. ***Min_Width*** and ***Min_Height*** of 20 and 20 pixels respectively for Kalman model 2, *constant acceleration*

Although it was first tried with the Kalman model 2 ***constant acceleration***, it is important to mention that the ***Constant velocity*** model is good enough. This is because we can observe the way the pedestrian is walking probably maintains the same velocity throughout the whole sequence. We can observe the result of Kalman model 1 on Figure 10

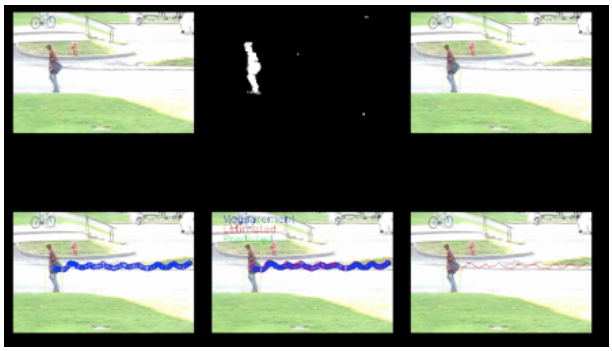

Fig. 10. ***Min_Width*** and ***Min_Height*** of 20 and 20 pixels respectively for Kalman model 1, *constant velocity*

3) ***Boats****:* For this model we assumed the constant acceleration model **Kalman 2**, due to the velocity changes nature of the boat when turning back to where it started. Although it probably did not have the same acceleration, the constant velocity model **Kalman 1** was not the best approach.

This was one challenging sequence since modification of the toy sequences' values was performed, as well as blob extraction values. In fact, the sequence ran into several problems. The first one occurred when obtaining the first blob center value, which was on the lower right part of the screen, which was further updated. This gave an incorrect initial center position of the boat, which can be seen on Figure 11 .

In this Figure (11) we can also see how there is some observation noise on the Y axis, therefore it was decided that values for the R matrix, corresponding to the Observation Matrix had to increase. If the observation noise increases it means that we trust it less, and we trust more the process. From this we decided to change the default value of the y dimension on the observation matrix from *25* to *100* pixels

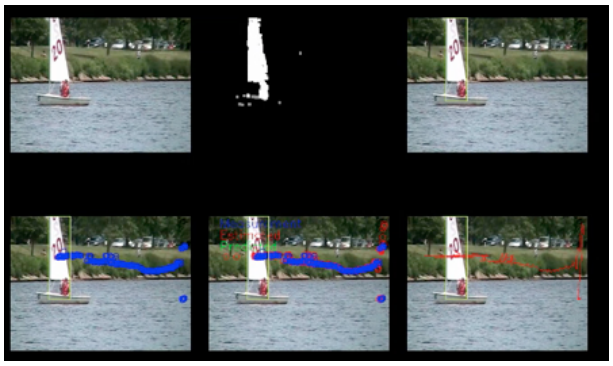

Fig. 11. ***Min_Width*** and ***Min_Height*** of 30 and 50 pixels respectively for Kalman model 2, *constant acceleration* with R(0,0)=25 R(1,1)=100

Although changing the ***Min_Width*** and ***Min_Height*** values to 30 and 80 respectively solved the first problem, we ran into another one: wrong detection of blobs, which biased the predictions due to wrong observations. In other words, noisy information was introduced in the model. From this we obtained the following, shown in Figure 12

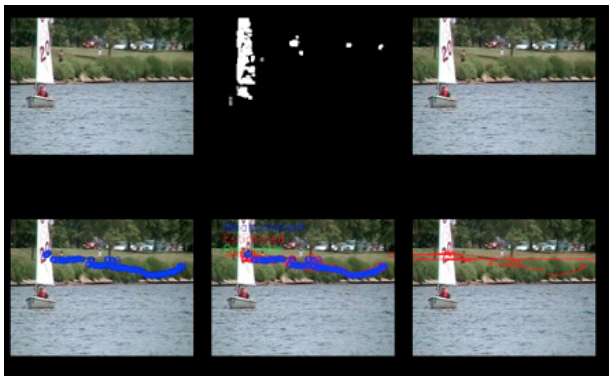

Fig. 12. ***Min_Width*** and ***Min_Height*** of 30 and 80 pixels respectively for Kalman model 2, *constant acceleration*, with R(0,0)=25 and R(1,1)=100

From this, it was decided that the default value of the x dimension on the observation matrix had to be changed as well. This was introduced my the detection of the cars which are not objects of interest to us, therefore they should not be tracked. We followed the same procedure, by avoiding observing values within 100 pixels around the center blob in context. Therefore R(0,0= changed from *25* to *100* as well. This can be seen in Figure 13 .

We can observe how a good tracking is observed, with a good starting blob detection. It is important to note that

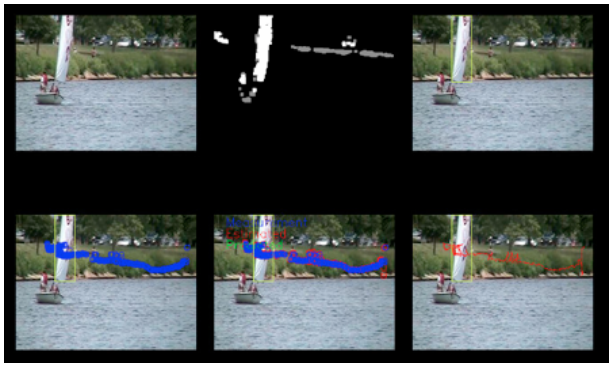

Fig. 13. ***Min_Width*** and ***Min_Height*** of 30 and 80 pixels respectively for Kalman model 2, *constant acceleration*, with R(0,0)=100 and R(1,1)=100

this sequence has a very complicated changing background, therefore, there are different blobs that are identified. Some of them can be deleted by changing the blob size to be extracted; however, others need more fine tuning of the Kalman Filtering Approach.

4) ***Street Corner At Night****:* Although it is not easy to notice the speed of the object in context (a car in this case), we supposed that the velocity was constant during the sequence. Therefore, it was decided that the model to be used was the constant velocity model **Kalman 1**. Although the assumption seemed to be correct, and the object was in fact identified, some problems arose, since the drastic change in lightning produced the identification of non existing objects and therefore blobs that introduced noise in our model. Not only so, but the noisy observations were introduced before the identification of the important object to be tracked. We can observe this behaviour in the following Figure 14

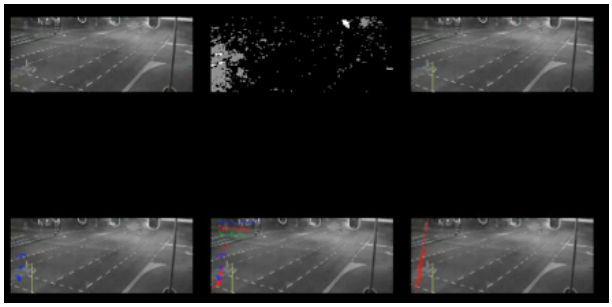

Fig. 14. Wrong detections due to sudden changes in Lightning

This was solved by modifying the parameters of the opening morphological operation, in which the Kernel size changed to *5*, which in rough terms means less noisy information being detected. With this, and a change in the min size of the blobs, we could see better results. This is shown in Figure 15

Although intuitively the best model to be chosen is the constant velocity one, we can observe very similar results are obtained with the same values, with a modified kernel size, and a constant acceleration Kalman model. This is observed

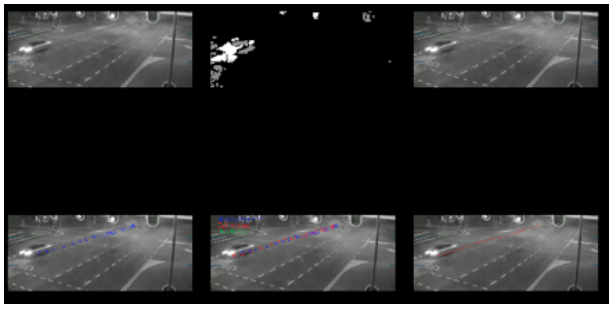

Fig. 15. Improved detection and correct tracking. ***Min_Width*** and ***Min_Height*** of 80 and 80 pixels respectively for Kalman model 1, *constant velocity*

in Figure 16

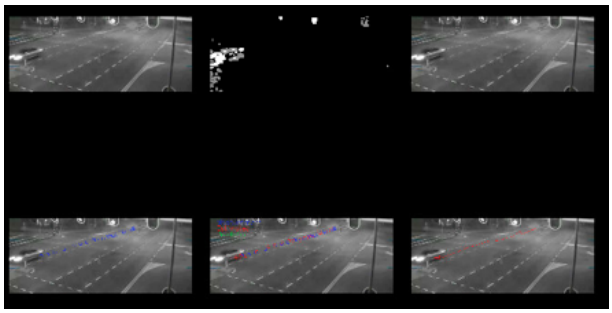

Fig. 16. Improved detection and correct tracking. ***Min_Width*** and ***Min_Height*** of 80 and 80 pixels respectively for Kalman model 2, *constant acceleration*

VII. CONCLUSIONS

Two comparison of two models of the Kalman Filer on toy examples and especially real world scenarios shows that the constant velocity model may lead to errors when the velocity of the objects is not constant and changes with some time. The constant acceleration approach allows considering this factor, which makes the prediction more accurate.

From this mentioned, the constant acceleration approach is more wholesome; therefore, although more computationally expensive, gives a more accurate prediction of the next state given the prediction model.

Moreover, the set of experiments proved that accuracy also depends on the measurement noise covariance (R matrix) - for instance, when the video sequence is considered to be noiseless, then the trust in the measurement is greater than in prediction.

Also, it is worth to notice that the correct work of the Kalman Filter cannot be considered without such steps as extracting the measurement as foreground detection, morphological opening and blob detection. The experiments show that setting right parameter values of these methods directly affects the performance of the Kalman Filter.

To conclude, the Kalman Filter indeed proves its reliability even in complex real world scenarios. However, to reach accurate results of tracking is highly important to fine tune the algorithm parameters and choose the proper model.

VIII. TIME LOG

A. *Kalman Filter*

- approximately **3 hours** were taken to read the paper and understand the main steps, as well as the idea behind each of matrix used in the Kalman filer;
- implementation and testing of the algorithm took around **3 hours** as well, later **1 hour** in total required to fix bugs.

B. *Coding Other Parts*

Since there was no structure provided, it took more time to develop the project from scratch. In total, it took around **4 hours** to implement other parts of the project - the *Main* and *FgSegment* functions.

C. *Report*

Around 7 hours: writing, analyzing the results, taking screenshots of the frames, and editing.

REFERENCES

[1] Lange, A. A. (2003): "Optimal Kalman Filtering for ultra-reliable Tracking", ESA CD-ROM WPP-237, Atmospheric Remote Sensing using Satellite Navigation Systems, Special Symposium of the URSI Joint Working Group FG, 13–15 October 2003, Matera, Italy.

[2] San Miguel J. "Applied Video Sequence Analysis Unit III Visual Tracking Prediction" Universidad Autónoma de Madrid