Soccer ball tracking with Kalman filter

1st Joshua Soutelo Vieira
Visual Recognition
University of Vigo
Pontevedra, Spain
jsoutelo@alumnos.uvigo.es

I. Introduction

This document presents the workings carried out to replicate the technique proposed in the technical paper [1]. This technique tries to address a fundamental problem when tracking a soccer ball over long-view sequences of a football field. This problem being the player occluding the ball for some arbitrary period of time. A regular Kalman filter under that condition can only make a pool prediction. The error gets worse the longer the player holds the ball. In order to tackle that, they propose a Kalman filter that can adapt its parameters dynamically. Furthermore, a velocity control mechanism is proposed. It would be in charge of adapting the velocity of the state to reflect that a player has the possession of the ball during several frames. In addition, in that piece of work is also proposed a procedure to segment the objects in the scene into player and ball candidates.

A regular Kalman filter was implemented successfully. But the dynamic version resulted into failure. Two possible reasons made that happened:

1) Incongruities in the mathematical formulation. Lets elaborate on this. The problem arises of wrong matrix shape in (3). The matrix **R** is defined in [1] as being of shape 4×4. Also, in [1] the **H** matrix is define as having a 2×4 shape. From the operation in between the parenthesis the shape of **P** could be extracted. Since we have $2 \times 4 \cdot P \cdot 4 \times 2$, P has to be of shape 4×4. Also, this makesSoccer ball tracking with Kalman filter 1st Joshua Soutelo Vieira Visual Recognition University of Vigo Pontevedra, Spain jsoutelo@alumnos.uvigo.es Abstract— Index Terms—Kalman filter, object tracking, soccer ball I. INTRODUCTION This document presents the workings carried out to replicate the technique proposed in the technical paper [1]. This tech- nique tries to address a fundamental problem when tracking a soccer ball over long-view sequences of a football field. This problem being the player occluding the ball for some arbitrary period of time. A regular Kalman filter under that condition can only make a pool prediction. The error gets worse the longer the player holds the ball. In order to tackle that, they propose a Kalman filter that can adapt its parameters dynamically. Furthermore, a velocity control mechanism is proposed. It would be in charge of adapting the velocity of the state to reflect that a player has the possession of the ball during several frames. In addition, in that piece of work is also proposed a procedure to segment the objects in the scene into player and ball candidates. A regular Kalman filter was implemented successfully. But the dynamic version resulted into failure. Two possible reasons made that happened: 1) Incongruities in the mathematical formulation. Lets elab- orate on this. The problem arises of wrong matrix shape in (3). The matrix R is defined in [1] as being of shape 4 × 4. Also, in [1] the H matrix is define as having a 2 × 4 shape. From the operation in between the parenthesis the shape of P could be extracted. Since we have 2×4·P ·4×2, P has to be of shape 4×4. Also, this makes sense, since the Kalman filter keeps track of four variables, x, y positions and velocities. The result of that operation returns a matrix of shape 2 × 2, that is not compatible to sum with R that is of shape 4 × 4. 2) Misunderstanding from my part of a core concept on either Kalman filtering or the concepts lied out in [1] itself. On the other hand, the method that is in charge of segment the objects in the scene into ball and players was successfully implemented. II. METHODS AND MATERIALS A. Kalman filter The Kalman filter belongs to a family of filters called Bayesian filters [2]. Its algorithm uses a series of measure- ments observed over time, including statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measure- ment alone, by estimating a joint probability distribution over the variables for each time-frame [3]. It can be decomposed in two different steps, prediction and correction. In the prediction step the predicted state and error covariance can be obtained as follows: ˆx k = Aˆxk1 + Buk1 (1) P k = APk1AT + Q (2) Where A is the transition matrix. Then, the correction step can be computed as follows: Kk = P k HT (HP k HT + R)1 (3) ˆxk = ˆx k + Kk(zk Hˆx k) (4) Pk = (I KkH)P k (5) The Kalman gain is calculated using the previously pre- dicted error covariance (3). Then the Kalman Filter corrects the state model by Kalman gain and measurement value that is the result of measurement in the sequence (4). Finally, the Kalman filter corrects the error covariance using the Kalman gain (5). It is also worth mentioning that here H corresponds to the transition matrix. Furthermore, [1] defines also the state model of the ball (6), along with

the transition matrix (7) and the measurement matrix (8). xk = [px, vx, py , vy] (6) A = 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 (7) H = [1 0 0 0 0 0 1 0] (8) B. Dataset 900 frames from a Premier League football match between the Chelsea and the Manchester City were used. All these frames were shot from a long-view position. Furthermore, the implemented solution was compared to some ground-truth ball positions. This data was automatically generated using the YOLOv5 model. Although, this model failed to recognize the sports ball in some instances where the it was over-lapped with a player. So data is missing on those situations. III. PROPOSED TECHNIQUE An overview of the implemented technique can be seen in Fig. 1. The last block representing the Kalman filtering can be expanded and visualized in Fig. 2. Fig. 1. Overview of the implemented technique. Fig. 2. Typical Kalman filter diagram. A. Ground detection In [1] they propose the following method to classify pixels in an RGB image as being part of the ground or not. Ground(x, y) = 1 (g(x, y) ¿ r(x, y) ¿ b(x, y)) (edge(x, y) == F alse) 0 otherwise (9) To compute Edge(x,y) the Canny edge detection algorithm implemented in the OpenCV library was used. B. Needless object removal First, objects such as the score and the public seats remain relatively fixed across frames. So, with that information com- bined with eccentricity and extent metrics was sufficient to remove them. The most complex scenario was removing the lines that were in touch with a player. In order to remove them, first the most dense areas in the image were found by using a mean filter with a kernel size of 15×15. Then, the pixels from a small region around those areas were marked as containing an object of interest. C. Player and ball candidate separation The binary blobs obtained from the previous step were scrutinized and compared against some features defined in [1]. For example, with respect to the player we can establish that w h ¡ 2. Also, that f rame h 16 ¡ player h ¡ f rame h 4 . In addition, since the ball width and height are similar to each other the following feature can be defined: w h ¡ 2 w h ¿ 0.5. From Fig. 3 results obtained form the above mentioned processes can be seen. Fig. 3. Results obtained from: a) Ground detection. b) Needless object removal. c) Player and ball candidate selection. IV. EXPERIMENTAL RESULTS From Fig. 3 we can see the results of the implemented technique. In both plots the horizontal axes means the frame number and the vertical axis means the distance of the measurement from the origin. In a) we can see some gaps. Those correspond to scenes where the ball is overlapped by the player. On the other hand, in b) we can see that every frame has a ball position, although it presents a lot of variations. sense, since the Kalman filter keeps track of four variables, x, y positions and velocities. The result of that operation returns a matrix of shape 2×2, that is not compatible to sum with $\mathbf{R}$ that is of shape 4×4.

2) Misunderstanding from my part of a core concept on either Kalman filtering or the concepts lied out in [1] itself.

On the other hand, the method that is in charge of segment the objects in the scene into ball and players was successfully implemented.

II. Methods and materials

A. Kalman filter

The Kalman filter belongs to a family of filters called Bayesian filters [2]. Its algorithm uses a series of measurements observed over time, including statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each time-frame [3]. It can be decomposed in two different steps, prediction and correction. In the prediction step the predicted state and error covariance can be obtained as follows:

$$\hat{x}_k^- = \mathbf{A}\hat{x}_{k-1} + \mathbf{B}u_{k-1} \tag{1}$$

$$\mathbf{P}_k^- = \mathbf{A}P_{k-1}\mathbf{A}^T + \mathbf{Q} \tag{2}$$

Where $\mathbf{A}$ is the transition matrix. Then, the correction step can be computed as follows:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \tag{3}$$

$$\hat{x}_k = \hat{x}_k^- + \mathbf{K}_k(z_k - \mathbf{H}\hat{x}_k^-) \tag{4}$$

$$\mathbf{P}_k = (I - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- \tag{5}$$

The Kalman gain is calculated using the previously predicted error covariance (3). Then the Kalman Filter corrects the state model by Kalman gain and measurement value that is the result of measurement in the sequence (4). Finally, the Kalman filter corrects the error covariance using the Kalman gain (5). It is also worth mentioning that here $\mathbf{H}$ corresponds to the transition matrix.

Furthermore, [1] defines also the state model of the ball (6), along with the transition matrix (7) and the measurement matrix (8).

$$x_k = [p_x, v_x, p_y, v_y] \tag{6}$$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{8}$$

B. Dataset

900 frames from a Premier League football match between the Chelsea and the Manchester City were used. All these frames were shot from a long-view position.

Furthermore, the implemented solution was compared to some ground-truth ball positions. This data was automatically generated using the YOLOv5 model. Although, this model failed to recognize the sports ball in some instances where the it was overlapped with a player. So data is missing on those situations.

III. Proposed technique

An overview of the implemented technique can be seen in Fig. 1. The last block representing the Kalman filtering can be expanded and visualized in Fig. 2.

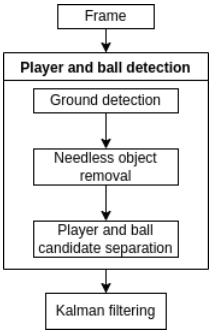

Fig. 1. Overview of the implemented technique.

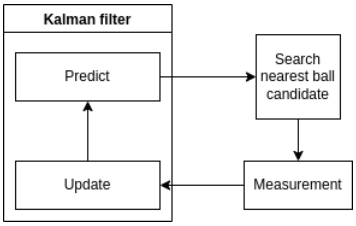

Fig. 2. Typical Kalman filter diagram.

A. Ground detection

In [1] they propose the following method to classify in an RGB image as being part of the ground or not.

$$Ground(x,y) = \begin{cases} 1 & (g(x,y) > r(x,y) > b(x,y))\& \\ & (edge(x,y) == False) \\ 0 & otherwise \end{cases} \quad (9)$$

To compute *Edge(x,y)* the Canny edge detection algorithm implemented in the OpenCV library was used.

B. Needless object removal

First, objects such as the score and the public seats remain relatively fixed across frames. So, with that information combined with eccentricity and extent metrics was sufficient to remove them. The most complex scenario was removing the lines that were in touch with a player. In order to remove them, first the most dense areas in the image were found by using a mean filter with a kernel size of 15×15. Then, the pixels from a small region around those areas were marked as containing an object of interest.

C. Player and ball candidate separation

The binary blobs obtained from the previous step were scrutinized and compared against some features defined in [1]. For example, with respect to the player we can establish that $\frac{w}{h} < 2$. Also, that $\frac{frame\_h}{16} < player\_h < \frac{frame\_h}{4}$. In addition, since the ball width and height are similar to each other the following feature can be defined: $\frac{w}{h} < 2 \ \& \ \frac{w}{h} > 0.5$.

From Fig. 3 results obtained form the above mentioned processes can be seen.

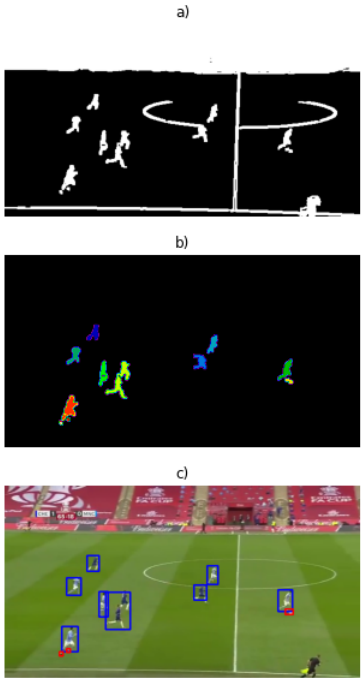

Fig. 3. Results obtained from: **a)** Ground detection. **b)** Needless object removal. **c)** Player and ball candidate selection.

IV. Experimental results

From Fig. 3 we can see the results of the implemented technique. In both plots the horizontal axes means the frame number and the vertical axis means the distance of the measurement from the origin. In a) we can see some gaps. Those correspond to scenes where the ball is overlapped by the player. On the other hand, in b) we can see that every frame has a ball position, although it presents a lot of variations.

Furthermore, from Table IV we can see that this method is suitable for real-time operation.

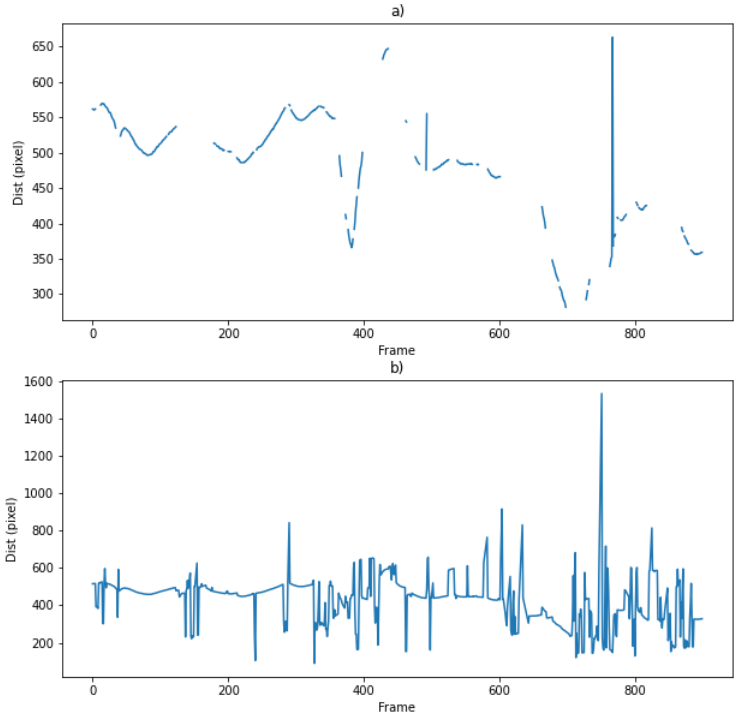

Fig. 4. Results obtained from: **a)** Ground truth ball positions generated by YOLOv5. **b)** Predicted ball positions by regular Kalman filter.

Total time (s)	Frames/s
19.80	45.44

TABLE I
ALGORITHM TIME METRICS.

V. CONCLUSIONS AND FURTHER WORK

Even with the regular implementation of the Kalman filter this proposed technique looks very promising. Adding those extra updates to the Kalman filter internal parameters will not add too much computational overhead. So this technique will fit real-time requirements. Lastly, a more robust approach to segment the objects in the scene and classify them into player and ball candidates should researched. Having in mind that it must meet the real-time criteria.

Lastly, working on this project demystified the Kalman filtering and showed me its importance and applicability. I owe a lot of this to [2]. It is a wonderful resource to learn about this topic in a practical fashion.

REFERENCES

[1] Kim, J.-Y., amp; Kim, T.-Y. (2009). Soccer ball tracking using dynamic Kalman filter with Velocity Control. 2009 *Sixth International Conference on Computer Graphics, Imaging and Visualization.* https://doi.org/10.1109/cgiv.2009.87

[2] R. Labbe, R. (2020). Kalman and Bayesian Filters in Python. https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python

[3] En.wikipedia.org. 2022. Kalman filter - Wikipedia. [online] Available at: ¡https://en.wikipedia.org/wiki/Kalman_filter¿ [Accessed 31 March 2022].