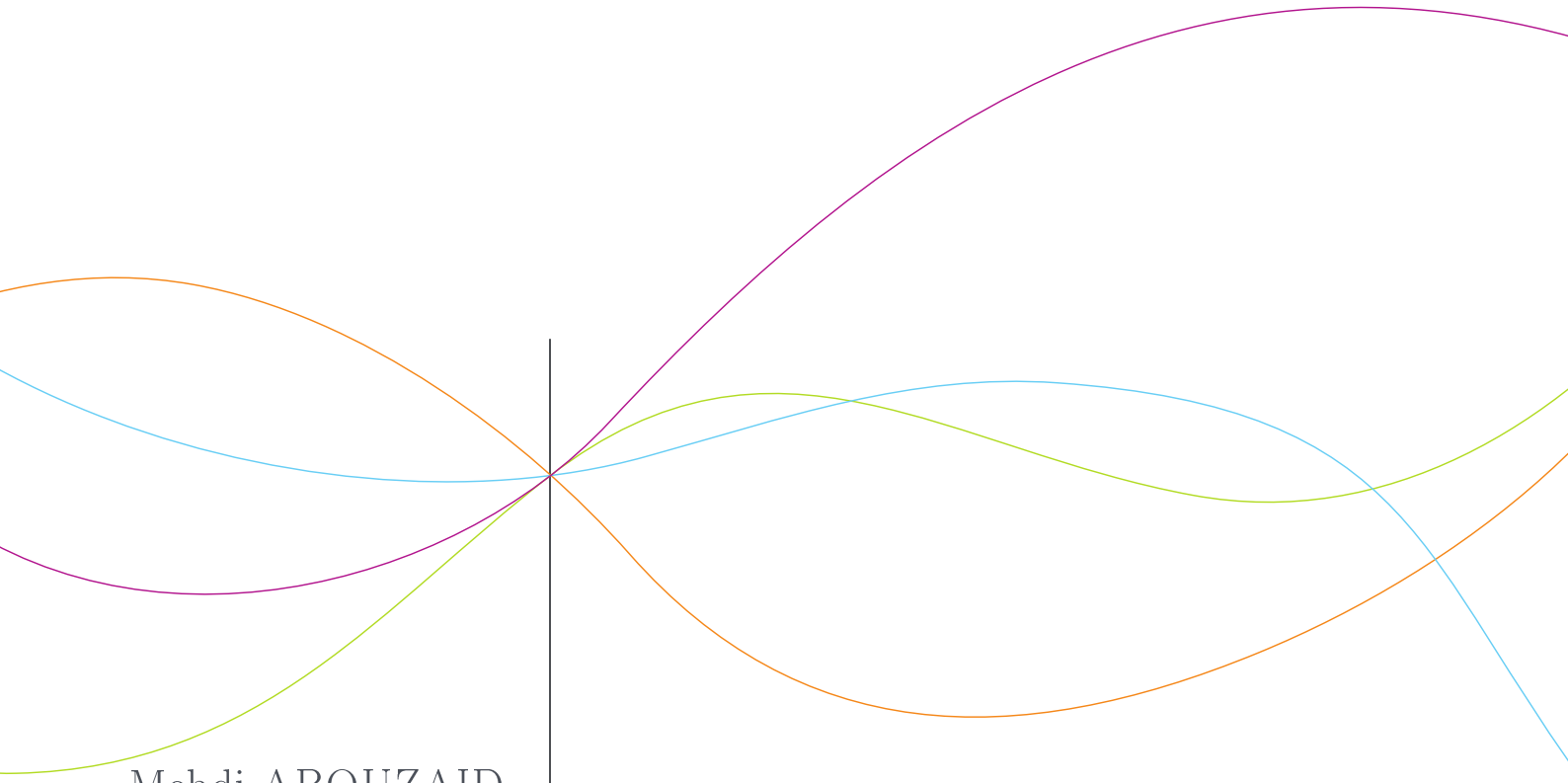
INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
ROUEN NORMANDIE



Mehdi ABOUZAID
Damien TOOMEY

À l'attention de
M. HÉRAULT

Filtre de Kalman/Suivi d'objets

Mini-projet TSA

TSA - ASI - 2017-2018

Table des matières

1 Étude bibliographique sur le modèle

1.1 Présentation

Le filtre de Kalman est un filtre à réponse impulsionnelle infinie (RII), c'est-à-dire qu'il est basé uniquement sur les valeurs en entrée du filtre ainsi que sur les valeurs antérieures de cette même réponse. Sa fonction est donc d'estimer les états d'un système dynamique à partir d'une série de mesures incomplètes ou bruitées.

Le filtre de Kalman fait appel à la dynamique de la cible qui définit son évolution dans le temps pour obtenir de meilleures données, éliminant ainsi l'effet du bruit. Ces données peuvent être calculées pour faire du filtrage ou pour de la prédiction.

1.2 Origine

Le filtre de Kalman a été nommé ainsi, suite à sa conception dans les années 1950 par Rudolf Kalman, mathématicien et automaticien américain d'origine hongroise. Pourtant, dès le XIX$^{\text{ème}}$ siècle, Thorvald Nicolai Thiele, astronome danois, puis au XX$^{\text{ème}}$ siècle, Peter Swerling, automaticien américain, avaient déjà tous les deux développé un algorithme similaire au filtre de Kalman.

Stanley Schmidt est reconnu comme ayant réalisé la première mise en œuvre du filtre. C'était lors d'une visite de Rudolf Kalman à la NASA Ames Research Center qu'il vit le potentiel de son filtre pour l'estimation de la trajectoire pour le programme Apollo. Ceci a conduit à l'utilisation du filtre dans l'ordinateur de navigation.

1.3 Fonctionnement

L'algorithme fonctionne en deux étapes, une étape de prédiction et une étape de mise à jour. La phase de prédiction utilise l'état estimé de l'instant précédent pour produire une estimation de l'état courant. Une fois les observations de l'instant courant obtenues, la seconde phase de mise à jour consiste à apporter une correction de l'état prédit dans le but d'obtenir une estimation plus précise. L'algorithme du filtre de Kalman est un processus dit récursif et markovien. Cela signifie que pour estimer l'état courant, seules l'estimation de l'état précédent et les mesures actuelles sont nécessaires, aucune autre information supplémentaire n'est requise.

Le filtre de Kalman est limité aux systèmes linéaires. Cependant, la plupart des systèmes physiques sont non linéaires. Le filtre n'est donc optimal que sur une petite plage linéaire des phénomènes réels. Une grande variété de filtres de Kalman a été depuis développée à partir de la formulation originale dite filtre de Kalman simple. Par exemple, Schmidt a développé le filtre de Kalman étendu, applicable aux phénomènes non linéaires.

Modèles du filtre

Modèle de dynamique ou système :
$x_k = Ax_{k-1} + q$
$x_k|x_{k-1} \sim \mathcal{N}(x_k|Ax_{k-1},\, Q)$

Modèle d'observation :
$y_k = Hx_k + r$
$y_k|x_k \sim \mathcal{N}(y_k|Hx_k,\, R)$

Distributions des états et des observations

Pour les états :
$P(x_k|y_{1:k-1}) \sim \mathcal{N}(x_k|m_k^-,\, P_k^-)$
$P(x_k|y_{1:k-1}) \sim \mathcal{N}(x_k|m_k^-,\, P_k^-)$

Pour les observations :
$P(x_k|y_{1:k-1}) \sim \mathcal{N}(x_k|m_k^-,\, P_k^-)$

Prédiction

Connaissant x[0 :k-1] et y[1 :k-1], on cherche à estimer x[k] :
$P(x_k|y_{1:k-1}) \sim \mathcal{N}(x_k|m_k^-,\, P_k^-)$
On applique le modèle de dynamique :
$m_k^- = Am_{k-1}$
$P_k^- = AP_{k-1}A^T + Q$

Mise à jour

Connaissant x[0 :k-1] et y[1 :k], on cherche à estimer x[k] :
$P(x_k|y_{1:k}) \sim \mathcal{N}(x_k|m_k,\, P_k)$
On corrige la prédiction de l'étape précédente par l'observation k :
$v_k = y_k - Hm_k^-$
$S_k = HP_k^-H^T + R$
$K_k = P_k^-H^TS_k^{-1}$
$m_k = m_k^- + K_kv_k$
$P_k = P_k^- - K_kS_kK_k^T$

Détail des matrices et vecteurs

Nous choisissons le modèle de l'accélération constante au vu des données que avons choisies.

Matrice de dynamique/transfert/système

$$A = \begin{pmatrix} 1 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Matrice d'observation

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Vecteur d'état

$$m_k = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{pmatrix}$$

Bruit de transition

$$Q = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 \\ 0 & b & 0 & 0 & 0 & 0 \\ 0 & 0 & c & 0 & 0 & 0 \\ 0 & 0 & 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 & e & 0 \\ 0 & 0 & 0 & 0 & 0 & f \end{pmatrix} \quad \forall a, b, c, d, e, f \in \mathbb{R}$$

Bruit d'observation

$$R = \begin{pmatrix} g & 0 \\ 0 & h \end{pmatrix} \quad \forall g, h \in \mathbb{R}$$

Covariance de l'état initial

$$p_0 = \begin{pmatrix} i & 0 & 0 & 0 & 0 & 0 \\ 0 & j & 0 & 0 & 0 & 0 \\ 0 & 0 & k & 0 & 0 & 0 \\ 0 & 0 & 0 & l & 0 & 0 \\ 0 & 0 & 0 & 0 & m & 0 \\ 0 & 0 & 0 & 0 & 0 & n \end{pmatrix} \quad \forall i, j, k, l, m, n \in \mathbb{R}$$

2 Étude bibliographique sur l'application

Le suivi d'objet correspond au fait de localiser un objet dans une vidéo ou une séquence d'images.

Dans les années 1960, le filtre de Kalman a été utilisé pour la première fois lors de la mission Apollo pour estimer la trajectoire de la fusée allant sur la Lune. Ce filtre est aussi utilisé pour corriger la trajectoire de missiles et calculer l'altitude d'une fusée.

A partir des années 1990, le filtre de Kalman s'est propagé à la société civile avec les GPS[1] (système de positionnement global).

Le filtre de Kalman permet également la fusion de données pour avoir une meilleur estimation de chaque donnée. Par exemple, si une voiture est équipée de trois capteurs :

- système de mesure inertiel (IMU[2]) : mesure l'accélération et la vitesse angulaire de la voiture
- Odomètre : donne la position relative de la voiture
- GPS : donne la position absolue de la voiture,

le filtre de Kalman permet de prendre en compte les données de chaque capteur et estimer une position plus précise de la voiture.

Aujourd'hui, il est possible de suivre un ou plusieurs objets dans une vidéo, en temps réel ou non, la position de la caméra étant fixe ou non. Le suivi d'objets est entre autres utilisé dans le domaine de sécurité, la réalité augmentée et le contrôle de la circulation. Si l'objet est détecté, sa position est corrigée sinon elle est prédite.

Dans le cadre de notre étude, nous nous concentrerons uniquement sur le suivi d'objets (tracking).

Vu le délai très court qui nous est imparti, nous n'étudierons le suivi d'un unique objet avec une caméra stationnaire (arrière plan constant).

1. GPS : Global Positioning System
2. IMU : Inertial Measurement Unit

3 Présentation du jeu de données choisi ou construit

3.1 Vidéo 1 : *singleball.mp4*

Pour notre projet, nous avons utilisé une vidéo nommée *singleball.mp4* trouvée dans les exemples fournis par *Matlab*. La vidéo correspond à une balle verte qui roule sur un sol foncé. Au cours de la vidéo, la balle passe sous une boîte en carton pendant un court instant puis réapparaît de l'autre de celle-ci. Pour faciliter le suivi de l'objet, la caméra est stationnaire (l'arrière-plan ne change pas).

Informations concernant la vidéo *singleball.mp4* :
- durée : 1.5 s
- largeur : 480
- hauteur : 360
- nombre d'images par seconde : 30
- bits par pixel : 24
- format : RGB24

3.2 Vidéo 2 : *singleballhomemade.mp4*

Nous souhaitions tester le code avec une seconde vidéo. Nous avons donc réalisé une vidéo avec la même mise en scène (caméra stationnaire), à savoir une balle qui suit une trajectoire et qui est cachée par un obstacle (un portefeuille dans notre cas) pendant un court instant puis réapparaît de l'autre côté de l'obstacle.

Informations concernant la vidéo *singleballhomemade.mp4* :
- durée : 3.0420 s
- largeur : 1280
- hauteur : 720
- nombre d'images par seconde : 30
- bits par pixel : 24
- format : RGB24

Dans les deux vidéos, nous avons donc deux missions :
- corriger la position de la balle quand la balle est détectée
- prédire la position de la balle quand elle passe sous la boîte (balle non détectée)

3 PRÉSENTATION DU JEU DE DONNÉES CHOISI OU CONSTRUIT

Nous avons commencé par étudier la vidéo *singleball.mp4*. Le code était donc bien adapté pour cette vidéo. En revanche, lorsque nous avons voulu tester le code avec la vidéo *singleballhomemade.mp4*, nous avons remarqué que le code n'était pas générique et ne fonctionnait donc pas pour cette seconde vidéo. En fait, le traitement d'image ne détectait pas la balle alors qu'elle était visible.

Malheureusement, nous n'avons pas eu le temps d'améliorer le code pour qu'il soit adapté à l'autre vidéo.

4 Explication de l'implémentation

Nous avons trouvé un code[3] *Matlab* sur le site MathWorks qui applique un filtre de Kalman sur la vidéo *singleball.mp4* mais cet exemple utilise la *toolbox vision* qui n'est pas accessible sur la version académique de *Matlab* de l'*INSA*.

Nous avons donc décidé de reproduire le code que nous avons trouvé sur ce site pour qu'il fonctionne sur la version académique. Nous avons obtenu le code suivant par nous-même et avec du code trouvé sur internet.

Dans ce code, nous avons considéré que la vidéo pouvait contenir l'objet dès la première image, ainsi, nous supposons que nous n'avons pas d'image de la vidéo avec uniquement l'arrière plan.

Le main se nomme *ProjetTSA.m*. Le code est commenté, ce qui sert également d'explication.

Initialisation/Ré-initialisation du fichier.

```
1  %%
2  clear all
3  clc
4  close all
```

Listing 1 – ProjetTSA.m

Utilisation de l'entrée standard sur *Matlab* pour permettre à l'utilisateur de choisir la vidéo qu'il souhaite étudier. Cela permet au programme de trouver le chemin vers le répertoire contenant la succession d'images.

```
6  %% Choix de la video a etudier
7
8  path=which('ProjetTSA.m');
9  path=path(1:(end-length('/Code/ProjetTSA.m')));
10
11 nomVideo=input('Quelle video voulez-vous etudier ?\nVoici le choix de video que
       vous avez : singleball.mp4, singleballhomemade.mp4\n\n', 's') ;
12
13 if strcmp(nomVideo, 'singleball.mp4')
14     path=fullfile(path, 'Donnees', 'Video1');
15     cd(path)
16     delta_t=1.2; % periode d'echantillonage de la video
17 else
18     if strcmp(nomVideo, 'singleballhomemade.mp4')
```

3. Code accessible dans /Code/kalmanFilterForTracking.m

```
19          path=fullfile(path, 'Donnees', 'Video2');
20          cd(path)
21          delta_t=1.2; % periode d'echantillonage de la video
22      else
23          disp('La video que vous avez entree n''existe pas');
24          return
25      end
26  end
27
28  informationSurDossier=dir(fullfile('VideoEnPNG', '*.png'));
29  NB_FRAMES=size(informationSurDossier,1);
```

Listing 2 – ProjetTSA.m

Tout d'abord, nous souhaitons ouvrir la vidéo dans *Matlab*.

```
31  %% Lecture de la video
32
33  %v=VideoReader(fullfile(path,nomVideo));
34
35  %% Visionnage de la video
36
37  %currAxes=axes;
38  %while hasFrame(v)
39  %    vidFrame=readFrame(v);
40  %    image(vidFrame, 'Parent', currAxes);
41  %    currAxes.Visible='off';
42  %    pause(1/v.FrameRate);
43  %end
```

Listing 3 – ProjetTSA.m

Malheureusement, sur la version *Matlab* 2017 de *Linux* de l'INSA, la lecture de vidéo n'est pas possible. Nous avons donc sauvegardé chaque image de la vidéo en format *png* et nous avons travaillé sur ces images.

```
45  %% Sauvegarde de chaque image de la video en png
46
47  %v.CurrentTime=0;
48  %nbFrame=0;
49  %while hasFrame(v)
50  %    vidFrame=readFrame(v);
51  %    nbFrame=nbFrame+1;
52  %    imwrite(vidFrame,fullfile('VideoEnPNG', [num2str(nbFrame) '.png']));
53  %end
```

Listing 4 – ProjetTSA.m

4.1 Traitement d'image

Pour détecter l'objet, on compare deux images à la fois (l'image courante et l'image suivante). Par conséquent, l'utilisation de la fonction *imabsdiff* peut entraîner l'apparition de deux objets car si l'image à l'instant k contient l'objet et l'image à l'instant $k+1$ contient également l'objet, la valeur absolue de la différence de ces deux images va faire apparaître deux objets. En fait, c'est le même objet à deux instants différents. Ainsi, quand on obtient le centre de masse avec *regionprops*, il peut y avoir un ou deux centres de masse.

```
55 %% Partie 1 : Traitement d'image
56 %% Detection de l'objet et stockage des centres de masse dans
       cell_position_objet
57
58 cell_position_objet=cell(1);
59 % cell_position_objet stockera :
60 % les coordonnees des centres de masse des objets detectes
61 % [] si aucun objet n'a ete detecte
62
63 i=1;
64 nbFrame=1;
65 tempImage=imread(fullfile('VideoEnPNG', [num2str(i), '.png'])); % lecture de la
       premiere image
66 empilementImagesBinaires=zeros(size(tempImage,1), size(tempImage,2)); %
       initialisation de empilementImagesBinaires
67
68 for i=1:NB_FRAMES
69     image_tk=tempImage;
70
71     if i<NB_FRAMES
72         image_tkPlus1=imread(fullfile('VideoEnPNG', [num2str(i+1), '.png']));
73         nbFrame=nbFrame+1;
74         tempImage=image_tkPlus1;
75
76         A=rgb2gray(image_tk); % A contient image_tk en noir et blanc
77         B=rgb2gray(image_tkPlus1); % B contient image_tkPlus1 en noir et blanc
78         C=imabsdiff(A, B); % C contient la valeur absolue de A-B
79         thresh=graythresh(C); % calcul de l'intensite normalisee de l'image (
       entre 0 et 1)
80
81         if (thresh>0.05) % au moins un objet a ete detecte (0.05 a ete choisi de
        maniere empirique)
82             ib=imbinarize(C, thresh); % convertit l'image C en image binaire
83             ib_sansbruit=bwareaopen(ib, 20, 8); % retrait des objets de moins de
        20 pixels (bruit)
84
85             s=regionprops(bwlabel(ib_sansbruit), 'centroid'); % determine entre
       autres le centre de masse de chaque objet
```

```
86              c=[s.Centroid]; % centre(s) de masse
87
88              cell_position_objet{nbFrame}=c; % stocke le/les centres de masse
89
90              empilementImagesBinaires=empilementImagesBinaires+ib_sansbruit; %
    empilement des images binaires
91
92          else % aucun objet n'a ete detecte
93              cell_position_objet{nbFrame}=[]; % stocke le fait qu'aucun objet n'a
     ete detecte
94          end
95      end
96  end
```

Listing 5 – ProjetTSA.m

Comme la détection de l'objet est imprécise, la détection du centre de masse l'est aussi. En comparant l'image 1 et l'image 2 (comparaison A) puis l'image 2 et l'image 3 (comparaison B), la détection du centre de masse de l'objet de l'image 2 dans les comparaisons A et B ne sont pas les mêmes (mais proches). Pourtant nous savons qu'il s'agit du même objet au même instant. Le code suivant permet de faire la moyenne des centres de masse lorsqu'il s'agit du même objet au même instant :

```
98  %% Moyennage des centres de masse
99
100 cell_pos_bis=[];
101 % cell_pos_bis stockera :
102 % la moyenne de deux centres de masse cell_position_objet
103 % un centre de masse de cell_position_objet
104 % [] si aucun objet n'a ete detecte
105
106 i=1;
107 k=1;
108 estVraiment2_4=true;
109 while i<=(length(cell_position_objet)-1)
110
111     if isempty(cell_position_objet{i})
112         cell_pos_bis{k}=[];
113         estVraiment2_4=true;
114     else if ((length(cell_position_objet{i})==2) && ((length(cell_position_objet
    {i+1})==4)))
115             if (estVraiment2_4==true)
116                 cell_pos_bis{k}=[];
117                 k=k+1;
118             end
119             cell_pos_bis{k}=mean([cell_position_objet{i}(1:2);
```

```
    cell_position_objet{i+1}(1:2)]);
120             cell_position_objet{i+1}(1:2)=[];
121             estVraiment2_4=false;
122         else if ((length(cell_position_objet{i})==4) && ((length(
    cell_position_objet{i+1})==4)))
123                 cell_pos_bis{k}=cell_position_objet{i}(1:2);
124                 k=k+1;
125                 cell_pos_bis{k}=mean([cell_position_objet{i}(3:4);
    cell_position_objet{i+1}(1:2)]);
126                 cell_position_objet{i+1}(1:2)=[];
127                 estVraiment2_4=false;
128             else if ((length(cell_position_objet{i})==4) && ((length(
    cell_position_objet{i+1})==2)))
129                     cell_pos_bis{k}=cell_position_objet{i}(1:2);
130                     k=k+1;
131                     cell_pos_bis{k}=mean([cell_position_objet{i}(3:4);
    cell_position_objet{i+1}(1:2)]);
132                     cell_position_objet(i+1)=[];
133                 else if ((length(cell_position_objet{i})==2) && ((length(
    cell_position_objet{i+1})==2)))
134                         cell_pos_bis{k}=cell_position_objet{i};
135                         else if ((length(cell_position_objet{i})==2) && (isempty
    (cell_position_objet{i+1})))
136                             cell_pos_bis{k}=cell_position_objet{i};
137                                 else if ((length(cell_position_objet{i})==4) &&
    (isempty(cell_position_objet{i+1})))
138                                     cell_pos_bis{k}=cell_position_objet{i}(1:2);
139                                     k=k+1;
140                                     cell_pos_bis{k}=cell_position_objet{i}(3:4);
141                                     i=i+1;
142                                     end
143                             end
144                     end
145                 end
146             end
147         end
148     end
149     k=k+1;
150     i=i+1;
151 end
152
153 % Dans le cas ou l'objet apparait sur l'image au dernier frame
154 if ~isempty(cell_position_objet{i})
155     cell_pos_bis{k}=cell_position_objet{i}(1:2);
156 end
```

Listing 6 – ProjetTSA.m

On retire les positions **[ ]** en fin de vidéo car on considère qu'on n'aura pas à prédire ces positions (on considère que l'objet est hors champ de la caméra même si cela n'est pas nécessairement vrai) :

```
158 %% Retrait des positions quand l'objet n'est plus detecte en fin de video
159
160 % Retrait des [] a la fin de cell_pos_bis
161 % car cela veut dire que l'objet n'est plus detecte dans la video
162 j=length(cell_pos_bis);
163 while isempty(cell_pos_bis{j})
164     if isempty(cell_pos_bis{j})
165         cell_pos_bis(j)=[];
166     end
167     j=length(cell_pos_bis);
168 end
```

Listing 7 – ProjetTSA.m

En raison d'imprécisions dans l'analyse d'image, il reste des points très proches mais il s'agit en fait des mêmes points donc nous faisons la moyenne de ces points. Nous avons décidé que si la distance entre deux points est inférieure à 3 alors il s'agit du même point (<= 3 choisi de manière empirique).

```
170 %% Moyenne des points qui sont encore trop pres (en fait ce sont les memes
    points)
171
172 i=1;
173 while (i<=length(cell_pos_bis)-1)
174     if (~isempty(cell_pos_bis{i}) && ~isempty(cell_pos_bis{i+1}))
175         if ((abs(cell_pos_bis{i}(1)-cell_pos_bis{i+1}(1))<=3) && (abs(
    cell_pos_bis{i}(2)-cell_pos_bis{i+1}(2))<=3))
176             cell_pos_bis{i}=mean([cell_pos_bis{i}; cell_pos_bis{i+1}]);
177             cell_pos_bis(i+1)=[];
178         end
179     end
180     i=i+1;
181 end
```

Listing 8 – ProjetTSA.m

Affichage des points détectés sur l'image binaire et sur l'image de départ en noir et blanc.

```
183 %% Affichage des detections des centres de masse de l'objet
184
185 % cell_pos_bis en matrice pour faciliter l'affichage
```

```
186 matrice_pos_bis=[];
187
188 for i=1:length(cell_pos_bis)
189     if ~isempty(cell_pos_bis{i})
190         matrice_pos_bis=[matrice_pos_bis; cell_pos_bis{i}(1), cell_pos_bis{i}(2)
        ];
191     end
192 end
193
194 figure
195 subplot(2,1,1)
196 imshow(empilementImagesBinaires)
197 hold on
198     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
199     title('Image binaire');
200     xlabel('Axe x');
201     ylabel('Axe y');
202     legend('Positions detectees avec traitement d''image')
203 hold off
204
205 subplot(2,1,2)
206 imshow(A)
207 hold on
208     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
209     title('Image de depart en noir et blanc');
210     xlabel('Axe x');
211     ylabel('Axe y');
212     legend('Positions detectees avec traitement d''image')
213 hold off
```

Listing 9 – ProjetTSA.m

4.2 Filtre de Kalman

Appel de la fonction *trackingObjet* pour appliquer le filtre de Kalman sur chaque positions de *cell_pos_bis*.

```
215 %% Partie 2
216 %% Filtre de Kalman
217
218 clearvars -except cell_pos_bis empilementImagesBinaires delta_t matrice_pos_bis
    A
219 clear trackingObjet
220 clear kalmanFilter
221 clear kalmanFilterAjustementsVitesseAcceleration
222
```

```
223 Matrice_y=trackingObjet(cell_pos_bis, delta_t);
```

Listing 10 – ProjetTSA.m

Pour tester le filtre de Kalman avec ajustements de la vitesse, de l'accélération et $dt = 1/30\ s$ il faut dé-commenter les lignes 11 et 15 puis commenter les lignes 10 et 14 dans *trackingObjet.m*. Les lignes 10 et 14 permettent d'appliquer le filtre de Kalman sans ajustement de la vitesse et de l'accélération avec *dt* choisi à la ligne 16 ou 21 dans *ProjetTSA.m* selon la vidéo choisie.

```
1  function Matrice_y=trackingObjet(cell_positions, delta_t)
2
3  Matrice_y=[]; % contiendra les positions corrigees et les predictions pour les
       points non detectes
4  objetDetectePremiereFois=false;
5
6  for i=1:length(cell_positions)
7      if ~isempty(cell_positions{i})
8          objetDetectePremiereFois=true;
9          z=cell_positions{i}';
10         y=kalmanFilter(z, delta_t,'objet detecte');   % prediction/correction de
       la position detectee avec un filtre de Kalman
11         %y=kalmanFilterAjustementsVitesseAcceleration(z, 'objet detecte');
12         Matrice_y=[Matrice_y y];
13     else if (objetDetectePremiereFois==true)
14         y=kalmanFilter(y, delta_t,'objet manquant'); % prediction de la position
       non detectee avec un filtre de Kalman
15         %y=kalmanFilterAjustementsVitesseAcceleration(y, 'objet manquant');
16         Matrice_y=[Matrice_y y];
17         end
18     end
19 end
20
21 Matrice_y=Matrice_y';
22
23 end
```

Listing 11 – trackingObjet.m

```
1  function y=kalmanFilter(z, delta_t, message)
2
3  % Matrice de dynamique/transfert/systeme
4  A=[1 0 delta_t 0 0 0;...      % x
5     0 1 0 delta_t 0 0;...      % y
6     0 0 1 0 delta_t 0;...      % Vx
```

```
7       0 0 0 1 0 delta_t;...       % Vy
8       0 0 0 0 1 0 ;...            % Ax
9       0 0 0 0 0 1];               % Ay
10
11  % Matrice d'observation
12  H=[1 0 0 0 0 0; 0 1 0 0 0 0];
13
14  % Bruit de transition
15  Q=100000*diag([25 25 10 10 1 1]);
16
17  % Bruit d'observation
18  R=25*eye(2);
19
20  % Une variable persistent est locale a la fonction mais sa valeur est
21  % conservee entre appels de fonction
22  persistent x_estime p_estime
23  if isempty(x_estime)
24      x_estime=zeros(6, 1);       % x_estime=[x,y,Vx,Vy,Ax,Ay]'
25      p_estime=100000*eye(6);     % covariance de l'etat initial
26      x_estime(1:2)=z;            % position initiale
27  end
28
29  % Prediction
30  x_predit=A*x_estime;
31  p_predit=A*p_estime*A'+Q;
32
33  if (strcmp(message, 'objet detecte')==1)
34      % Mise a jour
35      v=z-H*x_predit;
36      S=H*p_predit*H'+R;
37      K=(S\(H*p_predit'))';
38
39      x_estime=x_predit+K*v;
40      p_estime=p_predit-K*S*K';
41  else % strcmp(message, 'objet manquant')==1
42    x_estime=x_predit;
43    p_estime=p_predit;
44  end
45
46  % Estimation des observations
47  y=H*x_estime;
48
49  end
```

Listing 12 – kalmanFilter.m

Affichage des résultats :

```
225 %% Affichages des resultats
226
227 A=imread(fullfile('VideoEnPNG', [num2str(1), '.png'])); % 1.png (souvent arriere
        plan)
228
229 %%
230
231 figure
232 hold on
233     grid on
234     plot(Matrice_y(:,1), Matrice_y(:,2), 'gO');
235     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'b+')
236     set(gca,'Ydir','reverse')
237     title('Positions successives de l''objet');
238     xlabel('Axe x');
239     ylabel('Axe y (inverse)');
240     legend('Positions tracking objet (filtre de Kalman)',...
241         'Positions detectees avec traitement d''image')
242 hold off
243
244 %%
245
246 figure
247 hold on
248 subplot(2,1,1)
249 imshow(empilementImagesBinaires)
250 hold on
251     plot(Matrice_y(:,1), Matrice_y(:,2), 'gO-')
252     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
253     title('Image binaire');
254     xlabel('Axe x');
255     ylabel('Axe y');
256     legend('Positions tracking objet (filtre de Kalman)',...
257         'Positions detectees avec traitement d''image')
258 hold off
259
260 subplot(2,1,2)
261 imshow(A)
262 hold on
263     plot(Matrice_y(:,1), Matrice_y(:,2), 'gO-')
264     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
265     title('Image de depart en couleur');
266     xlabel('Axe x');
267     ylabel('Axe y');
268     legend('Positions tracking objet (filtre de Kalman)',...
269         'Positions detectees avec traitement d''image')
```

```
270 hold off
271 hold off
272
273 %%
274
275 figure
276 hold on
277 image(A)
278 hold on
279     plot(Matrice_y(:,1), Matrice_y(:,2), 'gO-')
280     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
281     set(gca,'Ydir','reverse')
282     ax = gca;
283     ax.Visible = 'off';
284     title('Image de depart en couleur');
285     xlabel('Axe x');
286     ylabel('Axe y');
287     legend('Positions tracking objet (filtre de Kalman)',...
288         'Positions detectees avec traitement d''image')
289 hold off
290 hold off
```

Listing 13 – ProjetTSA.m

5 Choix des hyper-paramètres

Le choix des hyper-paramètres pour le filtre de Kalman est particulièrement difficile. En effet, dans un cas réel, il est difficile de quantifier le bruit (covariance de l'état initial (p_0), bruit de transition (Q) et bruit d'observation (R)). Nous nous sommes donc basés sur les hyper-paramètres choisis dans la vidéo MathWorks (5min27s) qui traite *singleball.mp4* avec un filtre de Kalman.

$$p_0 = 100000 \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad Q = 100000 \cdot \begin{pmatrix} 25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad R = \begin{pmatrix} 25 & 0 \\ 0 & 25 \end{pmatrix}$$

Nous avons rencontré un problème lors de la définition de la matrice de transition A.

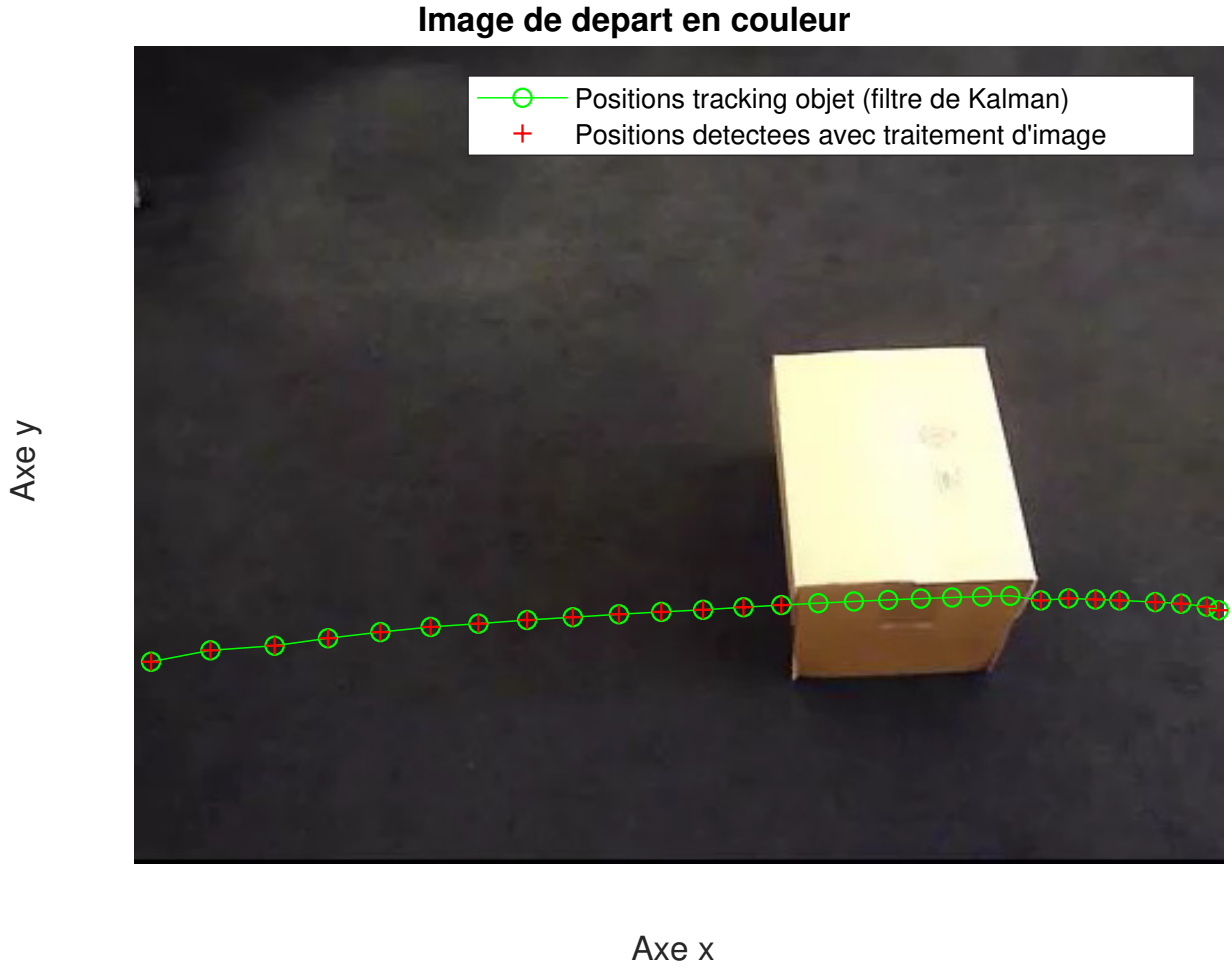
$$A = \begin{pmatrix} 1 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Pour la vidéo *singleball.mp4* par exemple, une des caractéristiques de cette vidéo était 30 *images/seconde*. Nous pensions alors que $dt = 1/30\ s$ mais cela donnait des résultats erronés. Nous avons alors mis $dt = 1\ s$ et les résultats étaient satisfaisants. Finalement, nous avons choisi $dt = 1.2\ s$. Malheureusement, nous ne comprenons pas pourquoi cette valeur de dt permet de mieux paramétrer le filtre.

6 Résultats et interprétations – singleball.mp4

6.1 Résultats avec $dt = 1.2\ s$

FIGURE 1 – Positions successives de la balle – Image 1 de la vidéo



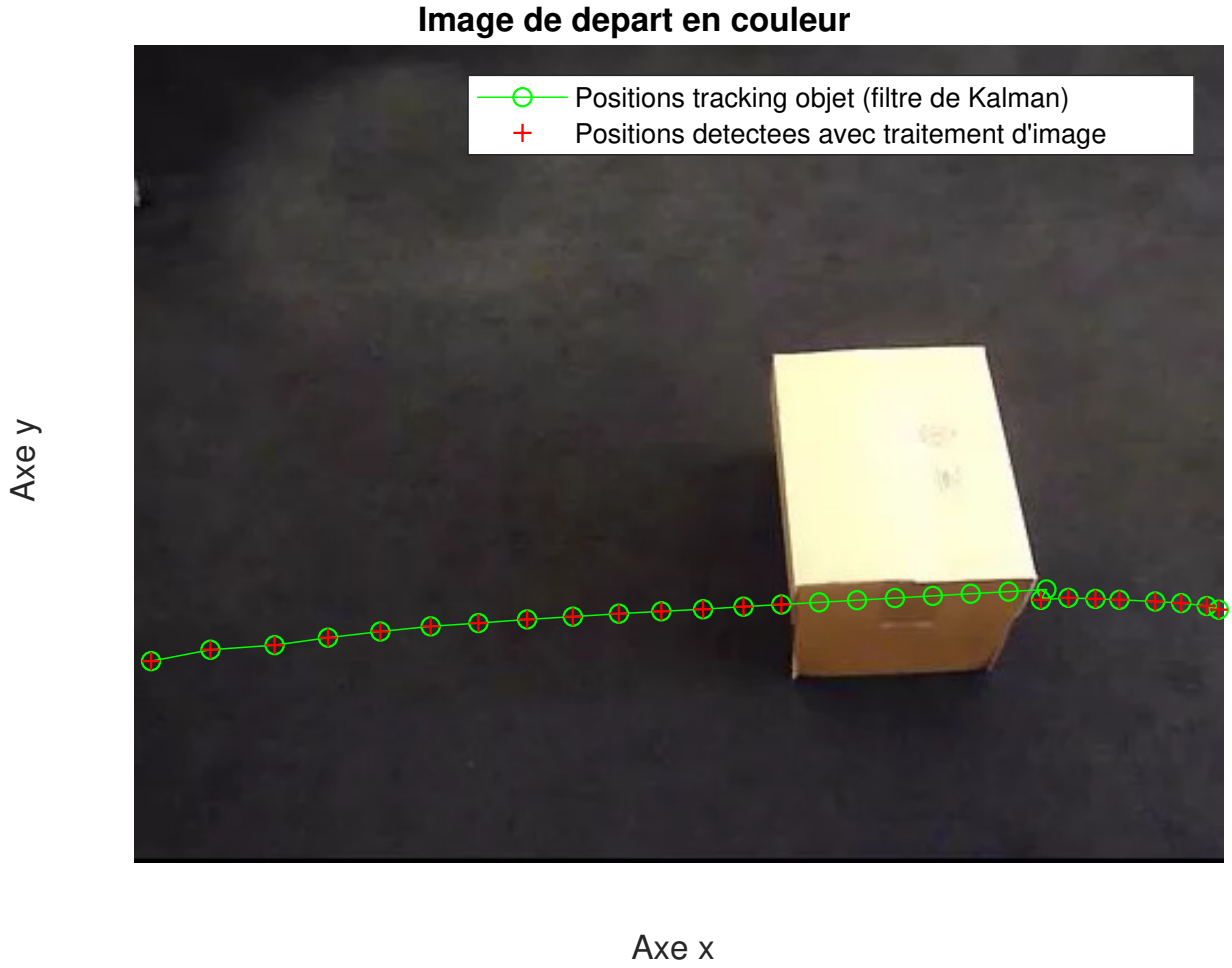
Sur la figure 1, les corrections des positions détectées semblent cohérentes. On remarque également que les prédictions des positions de la balle lorsqu'elle n'est pas détectée sont cohérentes.

FIGURE 2 – Positions successives de la balle

Sur la figure 2, on peut oberserver avec plus de détail les positions de la balle. On remarque que la prédiction des positions de la balle lorsqu'elle n'est pas détectée s'éloigne de sa vraie position. En effet, au bout de la septième prédiction, on voit que ce point est un peu éloigné de la position détectée lorsque la balle sort de l'autre côté de la boîte.

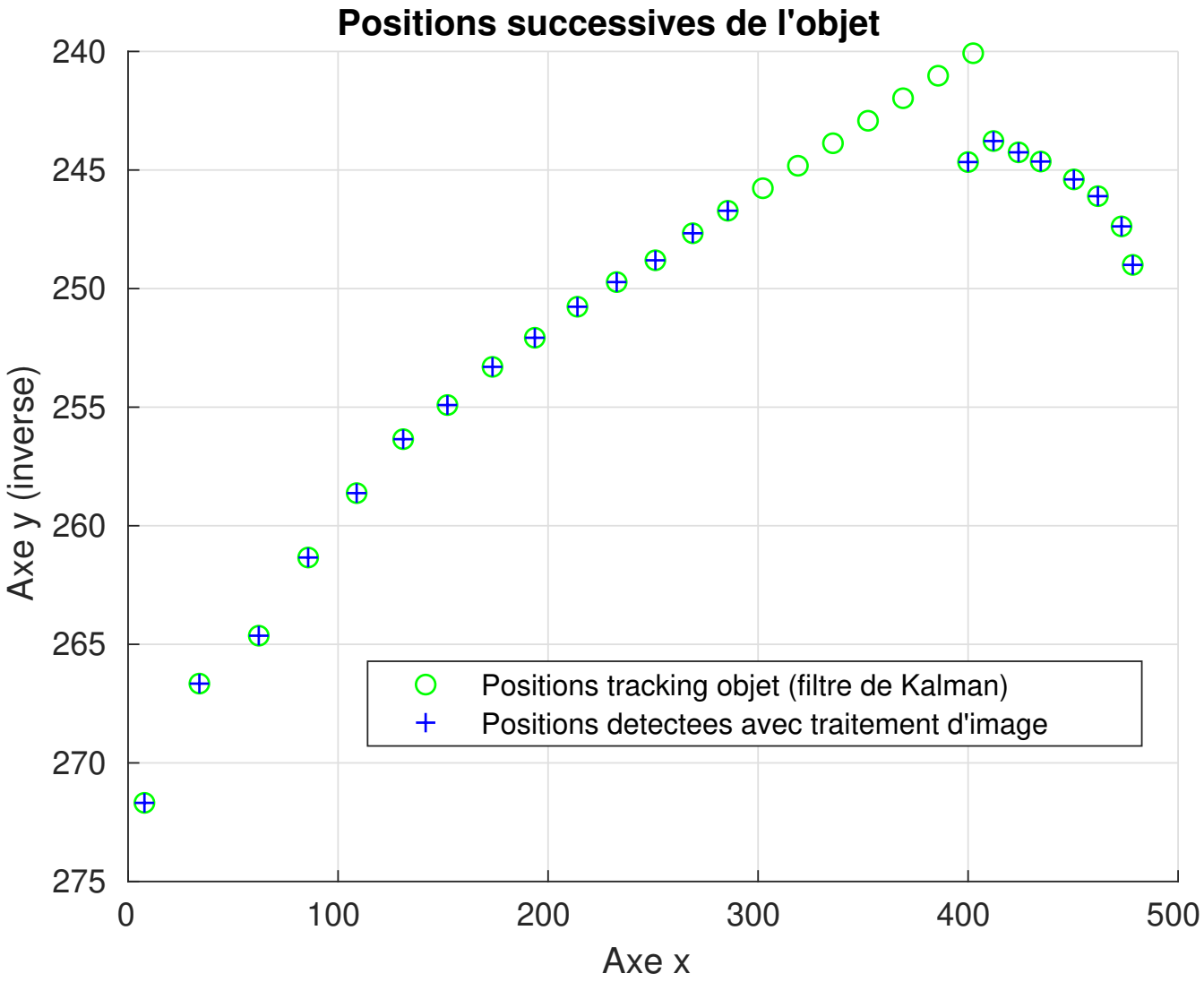
6.2 Résultats avec $dt = 1/30\ s$ et ajustements de la vitesse et de l'accélération

FIGURE 3 – Positions successives de la balle – Image 1 de la vidéo



Sur la figure 3, on remarque un problème lorsqu'on prédit la position de la balle quand elle n'est pas détectée. En effet, à la sortie de la boîte, le point prédit est trop loin.

FIGURE 4 – Positions successives de la balle



Sur la figure 4, on remarque que les prédictions de la position de la balle quand elle n'est pas détectée ne sont pas bonnes. En effet, ces sept points ont une tendance linéaire et le septième point est éloigné du point détecté à la sortie de la boîte.

7 Bibliographie

- **Étude bibliographique sur l'application**
  Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives]
  https://ieeexplore.ieee.org/document/5466132/
  Kalman filter - Applications in Image processing
  https://www.slideshare.net/raviteja1926/kalman-filter-applications-in-image-proces
  Applications of Kalman Filtering in Aerospace 1960 to the Present
  https://pdfs.semanticscholar.org/b7da/dbff2c53bc30bd910fc0db00e5071a37acfd.pdf
  Kalman filter
  https://en.wikipedia.org/wiki/Kalman_filter#Example_application
  Visual Object Tracking using Powell's Direct Set Method and Kalman Filtering
  https://www.youtube.com/watch?v=whwsLjLjEiY

- **Explication de l'implémentation**
  MathWorks - Understanding Kalman Filters
  https://fr.mathworks.com/videos/series/understanding-kalman-filters.html
  MathWorks - configureKalmanFilter (Computer Vision System Toolbox)
  https://fr.mathworks.com/help/vision/ref/configurekalmanfilter.html
  MathWorks - Using Kalman Filter for Object Tracking
  https://fr.mathworks.com/help/vision/examples/using-kalman-filter-for-object-track.html
  MathWorks - configureKalmanFilter
  https://www.mathworks.com/help/vision/ref/configurekalmanfilter.html#btiaa8o-Initi
  MathWorks - Vidéos et Webinars - Object Tracking using Kalman Filters
  https://www.mathworks.com/videos/introduction-to-kalman-filters-for-object-trackin.html
  MathWorks - Tracking Objects : Acquiring and Analyzing Image Sequences in MATLAB
  https://www.mathworks.com/company/newsletters/articles/tracking-objects-acquiring-.html
  C Code Generation for a MATLAB Kalman Filtering Algorithm
  https://www.mathworks.com/help/coder/examples/c-code-generation-for-a-matlab-kalma.html
  vision.KalmanFilter class
  https://www.mathworks.com/help/vision/ref/vision.kalmanfilter-class.html
  Compute Missing data Kalman
  https://stats.stackexchange.com/questions/140990/using-kalman-filters-to-impute-mi

8 Annexes

8.1 ProjetTSA.m

```
%%
clear all
clc
close all

%% Choix de la video a etudier

path=which('ProjetTSA.m');
path=path(1:(end-length('/Code/ProjetTSA.m')));

nomVideo=input('Quelle video voulez-vous etudier ?\nVoici le choix de video que
    vous avez : singleball.mp4, singleballhomemade.mp4\n\n', 's') ;

if strcmp(nomVideo, 'singleball.mp4')
    path=fullfile(path, 'Donnees', 'Video1');
    cd(path)
    delta_t=1.2; % periode d'echantillonage de la video
else
    if strcmp(nomVideo, 'singleballhomemade.mp4')
        path=fullfile(path, 'Donnees', 'Video2');
        cd(path)
        delta_t=1.2; % periode d'echantillonage de la video
    else
        disp('La video que vous avez entree n''existe pas');
        return
    end
end

informationSurDossier=dir(fullfile('VideoEnPNG', '*.png'));
NB_FRAMES=size(informationSurDossier,1);

%% Lecture de la video

%v=VideoReader(fullfile(path,nomVideo));

%% Visionnage de la video

%currAxes=axes;
%while hasFrame(v)
%    vidFrame=readFrame(v);
%    image(vidFrame, 'Parent', currAxes);
%    currAxes.Visible='off';
```

```
42 %     pause(1/v.FrameRate);
43 %end
44
45 %% Sauvegarde de chaque image de la video en png
46
47 %v.CurrentTime=0;
48 %nbFrame=0;
49 %while hasFrame(v)
50 %     vidFrame=readFrame(v);
51 %     nbFrame=nbFrame+1;
52 %     imwrite(vidFrame,fullfile('VideoEnPNG', [num2str(nbFrame) '.png']));
53 %end
54
55 %% Partie 1 : Traitement d'image
56 %% Detection de l'objet et stockage des centres de masse dans
       cell_position_objet
57
58 cell_position_objet=cell(1);
59 % cell_position_objet stockera :
60 % les coordonnees des centres de masse des objets detectes
61 % [] si aucun objet n'a ete detecte
62
63 i=1;
64 nbFrame=1;
65 tempImage=imread(fullfile('VideoEnPNG', [num2str(i), '.png'])); % lecture de la
       premiere image
66 empilementImagesBinaires=zeros(size(tempImage,1), size(tempImage,2)); %
       initialisation de empilementImagesBinaires
67
68 for i=1:NB_FRAMES
69     image_tk=tempImage;
70
71     if i<NB_FRAMES
72         image_tkPlus1=imread(fullfile('VideoEnPNG', [num2str(i+1), '.png']));
73         nbFrame=nbFrame+1;
74         tempImage=image_tkPlus1;
75
76         A=rgb2gray(image_tk); % A contient image_tk en noir et blanc
77         B=rgb2gray(image_tkPlus1); % B contient image_tkPlus1 en noir et blanc
78         C=imabsdiff(A, B); % C contient la valeur absolue de A-B
79         thresh=graythresh(C); % calcul de l'intensite normalisee de l'image (
       entre 0 et 1)
80
81         if (thresh>0.05) % au moins un objet a ete detecte (0.05 a ete choisi de
        maniere empirique)
82             ib=imbinarize(C, thresh); % convertit l'image C en image binaire
83             ib_sansbruit=bwareaopen(ib, 20, 8); % retrait des objets de moins de
```

```
    20 pixels (bruit)
84
85              s=regionprops(bwlabel(ib_sansbruit), 'centroid'); % determine entre
    autres le centre de masse de chaque objet
86              c=[s.Centroid]; % centre(s) de masse
87
88              cell_position_objet{nbFrame}=c; % stocke le/les centres de masse
89
90              empilementImagesBinaires=empilementImagesBinaires+ib_sansbruit; %
    empilement des images binaires
91
92          else % aucun objet n'a ete detecte
93              cell_position_objet{nbFrame}=[]; % stocke le fait qu'aucun objet n'a
     ete detecte
94          end
95      end
96  end
97
98  %% Moyennage des centres de masse
99
100 cell_pos_bis=[];
101 % cell_pos_bis stockera :
102 % la moyenne de deux centres de masse cell_position_objet
103 % un centre de masse de cell_position_objet
104 % [] si aucun objet n'a ete detecte
105
106 i=1;
107 k=1;
108 estVraiment2_4=true;
109 while i<=(length(cell_position_objet)-1)
110
111     if isempty(cell_position_objet{i})
112         cell_pos_bis{k}=[];
113         estVraiment2_4=true;
114     else if ((length(cell_position_objet{i})==2) && ((length(cell_position_objet
    {i+1})==4)))
115             if (estVraiment2_4==true)
116                 cell_pos_bis{k}=[];
117                 k=k+1;
118             end
119             cell_pos_bis{k}=mean([cell_position_objet{i}(1:2);
    cell_position_objet{i+1}(1:2)]);
120             cell_position_objet{i+1}(1:2)=[];
121             estVraiment2_4=false;
122         else if ((length(cell_position_objet{i})==4) && ((length(
    cell_position_objet{i+1})==4)))
123             cell_pos_bis{k}=cell_position_objet{i}(1:2);
```

```
124                 k=k+1;
125                 cell_pos_bis{k}=mean([cell_position_objet{i}(3:4);
    cell_position_objet{i+1}(1:2)]);
126                 cell_position_objet{i+1}(1:2)=[];
127                 estVraiment2_4=false;
128             else if ((length(cell_position_objet{i})==4) && ((length(
    cell_position_objet{i+1})==2)))
129                     cell_pos_bis{k}=cell_position_objet{i}(1:2);
130                     k=k+1;
131                     cell_pos_bis{k}=mean([cell_position_objet{i}(3:4);
    cell_position_objet{i+1}(1:2)]);
132                     cell_position_objet(i+1)=[];
133                 else if ((length(cell_position_objet{i})==2) && ((length(
    cell_position_objet{i+1})==2)))
134                         cell_pos_bis{k}=cell_position_objet{i};
135                         else if ((length(cell_position_objet{i})==2) && (isempty
    (cell_position_objet{i+1})))
136                             cell_pos_bis{k}=cell_position_objet{i};
137                                 else if ((length(cell_position_objet{i})==4) &&
    (isempty(cell_position_objet{i+1})))
138                                     cell_pos_bis{k}=cell_position_objet{i}(1:2);
139                                     k=k+1;
140                                     cell_pos_bis{k}=cell_position_objet{i}(3:4);
141                                     i=i+1;
142                                     end
143                             end
144                     end
145                 end
146             end
147         end
148     end
149     k=k+1;
150     i=i+1;
151 end
152
153 % Dans le cas ou l'objet apparait sur l'image au dernier frame
154 if ~isempty(cell_position_objet{i})
155     cell_pos_bis{k}=cell_position_objet{i}(1:2);
156 end
157
158 %% Retrait des positions quand l'objet n'est plus detecte en fin de video
159
160 % Retrait des [] a la fin de cell_pos_bis
161 % car cela veut dire que l'objet n'est plus detecte dans la video
162 j=length(cell_pos_bis);
163 while isempty(cell_pos_bis{j})
164     if isempty(cell_pos_bis{j})
```

```
165             cell_pos_bis(j)=[];
166         end
167         j=length(cell_pos_bis);
168     end
169
170     %% Moyenne des points qui sont encore trop pres (en fait ce sont les memes
        points)
171
172     i=1;
173     while (i<=length(cell_pos_bis)-1)
174         if (~isempty(cell_pos_bis{i}) && ~isempty(cell_pos_bis{i+1}))
175             if ((abs(cell_pos_bis{i}(1)-cell_pos_bis{i+1}(1))<=3) && (abs(
        cell_pos_bis{i}(2)-cell_pos_bis{i+1}(2))<=3))
176                 cell_pos_bis{i}=mean([cell_pos_bis{i}; cell_pos_bis{i+1}]);
177                 cell_pos_bis(i+1)=[];
178             end
179         end
180         i=i+1;
181     end
182
183     %% Affichage des detections des centres de masse de l'objet
184
185     % cell_pos_bis en matrice pour faciliter l'affichage
186     matrice_pos_bis=[];
187
188     for i=1:length(cell_pos_bis)
189         if ~isempty(cell_pos_bis{i})
190             matrice_pos_bis=[matrice_pos_bis; cell_pos_bis{i}(1), cell_pos_bis{i}(2)
        ];
191         end
192     end
193
194     figure
195     subplot(2,1,1)
196     imshow(empilementImagesBinaires)
197     hold on
198         plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
199         title('Image binaire');
200         xlabel('Axe x');
201         ylabel('Axe y');
202         legend('Positions detectees avec traitement d''image')
203     hold off
204
205     subplot(2,1,2)
206     imshow(A)
207     hold on
208         plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
```

```
209     title('Image de depart en noir et blanc');
210     xlabel('Axe x');
211     ylabel('Axe y');
212     legend('Positions detectees avec traitement d''image')
213 hold off
214 
215 %% Partie 2
216 %% Filtre de Kalman
217 
218 clearvars -except cell_pos_bis empilementImagesBinaires delta_t matrice_pos_bis
    A
219 clear trackingObjet
220 clear kalmanFilter
221 clear kalmanFilterAjustementsVitesseAcceleration
222 
223 Matrice_y=trackingObjet(cell_pos_bis, delta_t);
224 
225 %% Affichages des resultats
226 
227 A=imread(fullfile('VideoEnPNG', [num2str(1), '.png'])); % 1.png (souvent arriere
     plan)
228 
229 %%
230 
231 figure
232 hold on
233     grid on
234     plot(Matrice_y(:,1), Matrice_y(:,2), 'gO');
235     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'b+')
236     set(gca,'Ydir','reverse')
237     title('Positions successives de l''objet');
238     xlabel('Axe x');
239     ylabel('Axe y (inverse)');
240     legend('Positions tracking objet (filtre de Kalman)',...
241         'Positions detectees avec traitement d''image')
242 hold off
243 
244 %%
245 
246 figure
247 hold on
248 subplot(2,1,1)
249 imshow(empilementImagesBinaires)
250 hold on
251     plot(Matrice_y(:,1), Matrice_y(:,2), 'gO-')
252     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
253     title('Image binaire');
```

```
254     xlabel('Axe x');
255     ylabel('Axe y');
256     legend('Positions tracking objet (filtre de Kalman)',...
257         'Positions detectees avec traitement d''image')
258 hold off
259 
260 subplot(2,1,2)
261 imshow(A)
262 hold on
263     plot(Matrice_y(:,1), Matrice_y(:,2), 'gO-')
264     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
265     title('Image de depart en couleur');
266     xlabel('Axe x');
267     ylabel('Axe y');
268     legend('Positions tracking objet (filtre de Kalman)',...
269         'Positions detectees avec traitement d''image')
270 hold off
271 hold off
272 
273 %%
274 
275 figure
276 hold on
277 image(A)
278 hold on
279     plot(Matrice_y(:,1), Matrice_y(:,2), 'gO-')
280     plot(matrice_pos_bis(:,1), matrice_pos_bis(:,2), 'r+')
281     set(gca,'Ydir','reverse')
282     ax = gca;
283     ax.Visible = 'off';
284     title('Image de depart en couleur');
285     xlabel('Axe x');
286     ylabel('Axe y');
287     legend('Positions tracking objet (filtre de Kalman)',...
288         'Positions detectees avec traitement d''image')
289 hold off
290 hold off
```

Listing 14 – ProjetTSA.m

8.2 trackingObjet.m

```
1 function Matrice_y=trackingObjet(cell_positions, delta_t)
2 
3 Matrice_y=[]; % contiendra les positions corrigees et les predictions pour les
      points non detectes
4 objetDetectePremiereFois=false;
```

```
5
6   for i=1:length(cell_positions)
7       if ~isempty(cell_positions{i})
8           objetDetectePremiereFois=true;
9           z=cell_positions{i}';
10          y=kalmanFilter(z, delta_t,'objet detecte');   % prediction/correction de
        la position detectee avec un filtre de Kalman
11          %y=kalmanFilterAjustementsVitesseAcceleration(z, 'objet detecte');
12          Matrice_y=[Matrice_y y];
13      else if (objetDetectePremiereFois==true)
14          y=kalmanFilter(y, delta_t,'objet manquant'); % prediction de la position
        non detectee avec un filtre de Kalman
15          %y=kalmanFilterAjustementsVitesseAcceleration(y, 'objet manquant');
16          Matrice_y=[Matrice_y y];
17          end
18      end
19  end
20
21  Matrice_y=Matrice_y';
22
23  end
```

Listing 15 – trackingObjet.m

8.3 kalmanFilter.m

```
1   function y=kalmanFilter(z, delta_t, message)
2
3   % Matrice de dynamique/transfert/systeme
4   A=[1 0 delta_t 0 0 0;...      % x
5      0 1 0 delta_t 0 0;...      % y
6      0 0 1 0 delta_t 0;...      % Vx
7      0 0 0 1 0 delta_t;...      % Vy
8      0 0 0 0 1 0 ;...           % Ax
9      0 0 0 0 0 1];              % Ay
10
11  % Matrice d'observation
12  H=[1 0 0 0 0 0; 0 1 0 0 0 0];
13
14  % Bruit de transition
15  Q=100000*diag([25 25 10 10 1 1]);
16
17  % Bruit d'observation
18  R=25*eye(2);
19
20  % Une variable persistent est locale a la fonction mais sa valeur est
21  % conservee entre appels de fonction
```

```
22 persistent x_estime p_estime
23 if isempty(x_estime)
24     x_estime=zeros(6, 1);      % x_estime=[x,y,Vx,Vy,Ax,Ay]'
25     p_estime=100000*eye(6);    % covariance de l'etat initial
26     x_estime(1:2)=z;           % position initiale
27 end
28
29 % Prediction
30 x_predit=A*x_estime;
31 p_predit=A*p_estime*A'+Q;
32
33 if (strcmp(message, 'objet detecte')==1)
34     % Mise a jour
35     v=z-H*x_predit;
36     S=H*p_predit*H'+R;
37     K=(S\(H*p_predit'))';
38
39     x_estime=x_predit+K*v;
40     p_estime=p_predit-K*S*K';
41 else % strcmp(message, 'objet manquant')==1
42   x_estime=x_predit;
43   p_estime=p_predit;
44 end
45
46 % Estimation des observations
47 y=H*x_estime;
48
49 end
```

Listing 16 – kalmanFilter.m

8.4 kalmanFilterAjustementsVitesseAcceleration.m

```
1  function y=kalmanFilterAjustementsVitesseAcceleration(z, message)
2
3  delta_t=1/30;
4
5  % Matrice de dynamique/transfert/systeme
6  A=[1 0 delta_t 0 0 0;...      % x
7     0 1 0 delta_t 0 0;...      % y
8     0 0 1 0 delta_t 0;...      % Vx
9     0 0 0 1 0 delta_t;...      % Vy
10    0 0 0 0 1 0 ;...           % Ax
11    0 0 0 0 0 1];              % Ay
12
13 % Matrice d'observation
14 H=[1 0 0 0 0 0; 0 1 0 0 0 0];
```

```
15
16 % Bruit de transition
17 Q=100000*diag([25 25 10 10 1 1]);
18
19 % Bruit d'observation
20 R=25*eye(2);
21
22 % Une variable persistent est locale a la fonction mais sa valeur est
23 % conservee entre appels de fonction
24 persistent x_estime p_estime
25 if isempty(x_estime)
26     x_estime=zeros(6, 1);      % x_estime=[x,y,Vx,Vy,Ax,Ay]'
27     p_estime=100000*eye(6);    % covariance de l'etat initial
28     x_estime(1:2)=z;           % position initiale
29 end
30
31 % Vitesse de l'objet (deux images suffisent) : v=dx/dt
32 persistent countFrame X0 X1 Y0 Y1 VX0 VY0 AX AY VX0stocke VY0stocke;
33
34 if isempty(countFrame)
35     countFrame=1;
36 else
37     countFrame=countFrame+1;
38 end
39
40 if (strcmp(message, 'objet detecte')==1)
41     if (countFrame==1)
42         X0=z(1);
43         Y0=z(2);
44     end
45
46     if (countFrame>=2)
47         X1=z(1);
48         Y1=z(2);
49         if (countFrame<=3)
50             VX0=(X1-X0)/(delta_t);
51             VY0=(Y1-Y0)/(delta_t);
52             VX0stocke=VX0;
53             VY0stocke=VY0;
54         else
55             VX0=((X1-X0)/(delta_t))+delta_t*AX;
56             VY0=((Y1-Y0)/(delta_t))+delta_t*AY;
57         end
58
59         x_estime(3:4)=[VX0; VY0];
60         X0=X1;
61         Y0=Y1;
```

```
62
63          % Acceleration (on a besoin de trois images) : a=dv/dt
64          if (countFrame==3)
65              AX=(VX0-VX0stocke)/(delta_t);
66              AY=(VY0-VY0stocke)/(delta_t);
67              x_estime(5:6)=[AX; AY];
68          end
69      end
70  end
71
72  % Prediction
73  x_predit=A*x_estime;
74  p_predit=A*p_estime*A'+Q;
75
76  if (strcmp(message, 'objet detecte')==1)
77      % Mise a jour
78      v=z-H*x_predit;
79      S=H*p_predit*H'+R;
80      K=(S\(H*p_predit'))';
81
82      x_estime=x_predit+K*v;
83      p_estime=p_predit-K*S*K';
84  else % strcmp(message, 'objet manquant')==1
85    x_estime=x_predit;
86    p_estime=p_predit;
87  end
88
89  % Estimation des observations
90  y=H*x_estime;
91
92  end
```

Listing 17 – kalmanFilterAjustementsVitesseAcceleration.m

8.5 kalmanFilterForTracking.m

```
1  %%
2
3  function kalmanFilterForTracking
4
5  %%
6  showDetections();
7  showTrajectory();
8
9  frame             = [];  % A video frame
10 detectedLocation  = [];  % The detected location
11 trackedLocation   = [];  % The tracked location
```

```
label            = '';  % Label for the ball
utilities        = [];  % Utilities used to process the video

%% trackSingleObject

function trackSingleObject(param)
  % Create utilities used for reading video, detecting moving objects,
  % and displaying the results.
   utilities = createUtilities(param);

   isTrackInitialized = false;
   while ~isDone(utilities.videoReader)
     frame = readFrame();

     % Detect the ball.
     [detectedLocation, isObjectDetected] = detectObject(frame);

     if ~isTrackInitialized
       if isObjectDetected
         % Initialize a track by creating a Kalman filter when the ball is
         % detected for the first time.
         initialLocation = computeInitialLocation(param, detectedLocation);
         kalmanFilter = configureKalmanFilter(param.motionModel, ...
           initialLocation, param.initialEstimateError, ...
           param.motionNoise, param.measurementNoise);

         isTrackInitialized = true;
         trackedLocation = correct(kalmanFilter, detectedLocation);
         label = 'Initial';
       else
         trackedLocation = [];
         label = '';
       end

     else
       % Use the Kalman filter to track the ball.
       if isObjectDetected % The ball was detected.
         % Reduce the measurement noise by calling predict followed by
         % correct.
         predict(kalmanFilter);
         trackedLocation = correct(kalmanFilter, detectedLocation);
         label = 'Corrected';
       else % The ball was missing.
         % Predict the ball's location.
         trackedLocation = predict(kalmanFilter);
         label = 'Predicted';
       end
```

```
59      end
60
61      annotateTrackedObject();
62    end % while
63
64    showTrajectory();
65  end
66
67  %%
68  param = getDefaultParameters();  % get Kalman configuration that works well
69                                   % for this example
70
71  trackSingleObject(param);  % visualize the results
72
73  %%
74  param = getDefaultParameters();          % get parameters that work well
75  param.motionModel = 'ConstantVelocity'; % switch from ConstantAcceleration
76                                          % to ConstantVelocity
77  % After switching motion models, drop noise specification entries
78  % corresponding to acceleration.
79  param.initialEstimateError = param.initialEstimateError(1:2);
80  param.motionNoise          = param.motionNoise(1:2);
81
82  trackSingleObject(param); % visualize the results
83
84  %%
85
86  param = getDefaultParameters();  % get parameters that work well
87  param.initialLocation = [0, 0];  % location that's not based on an actual
        detection
88  param.initialEstimateError = 100*ones(1,3); % use relatively small values
89
90  trackSingleObject(param); % visualize the results
91
92  %%
93
94  param = getDefaultParameters();
95  param.segmentationThreshold = 0.0005; % smaller value resulting in noisy
        detections
96  param.measurementNoise      = 12500;  % increase the value to compensate
97                                        % for the increase in measurement noise
98
99  trackSingleObject(param); % visualize the results
100
101 %% getDefaultParameters
102
103 function param = getDefaultParameters
```

```
104   param.motionModel            = 'ConstantAcceleration';
105   param.initialLocation        = 'Same as first detection';
106   param.initialEstimateError   = 1E5 * ones(1, 3);
107   param.motionNoise            = [25, 10, 1];
108   param.measurementNoise       = 25;
109   param.segmentationThreshold = 0.05;
110 end
111
112 %% readFrame
113
114 function frame = readFrame()
115   frame = step(utilities.videoReader);
116 end
117
118 %% showDetections
119
120 function showDetections()
121   param = getDefaultParameters();
122   utilities = createUtilities(param);
123   trackedLocation = [];
124
125   idx = 0;
126   while ~isDone(utilities.videoReader)
127     frame = readFrame();
128     detectedLocation = detectObject(frame);
129     % Show the detection result for the current video frame.
130     annotateTrackedObject();
131
132     % To highlight the effects of the measurement noise, show the detection
133     % results for the 40th frame in a separate figure.
134     idx = idx + 1;
135     if idx == 40
136       combinedImage = max(repmat(utilities.foregroundMask, [1,1,3]), frame);
137       figure, imshow(combinedImage);
138     end
139   end % while
140
141   % Close the window which was used to show individual video frame.
142   uiscopes.close('All');
143 end
144
145 %% detectObject
146
147 function [detection, isObjectDetected] = detectObject(frame)
148   grayImage = rgb2gray(frame);
149   utilities.foregroundMask = step(utilities.foregroundDetector, grayImage);
150   detection = step(utilities.blobAnalyzer, utilities.foregroundMask);
```

```
151     if isempty(detection)
152       isObjectDetected = false;
153     else
154       % To simplify the tracking process, only use the first detected object.
155       detection = detection(1, :);
156       isObjectDetected = true;
157     end
158   end
159
160   %% annotateTrackedObject
161
162   function annotateTrackedObject()
163     accumulateResults();
164     % Combine the foreground mask with the current video frame in order to
165     % show the detection result.
166     combinedImage = max(repmat(utilities.foregroundMask, [1,1,3]), frame);
167
168     if ~isempty(trackedLocation)
169       shape = 'circle';
170       region = trackedLocation;
171       region(:, 3) = 5;
172       combinedImage = insertObjectAnnotation(combinedImage, shape, ...
173         region, {label}, 'Color', 'red');
174     end
175     step(utilities.videoPlayer, combinedImage);
176   end
177
178   %% showTrajectory
179
180   function showTrajectory
181     % Close the window which was used to show individual video frame.
182     uiscopes.close('All');
183
184     % Create a figure to show the processing results for all video frames.
185     figure; imshow(utilities.accumulatedImage/2+0.5); hold on;
186     plot(utilities.accumulatedDetections(:,1), ...
187       utilities.accumulatedDetections(:,2), 'k+');
188
189     if ~isempty(utilities.accumulatedTrackings)
190       plot(utilities.accumulatedTrackings(:,1), ...
191         utilities.accumulatedTrackings(:,2), 'r-o');
192       legend('Detection', 'Tracking');
193     end
194   end
195
196   %% accumulateResults
197
```

```
198 function accumulateResults()
199   utilities.accumulatedImage      = max(utilities.accumulatedImage, frame);
200   utilities.accumulatedDetections ...
201     = [utilities.accumulatedDetections; detectedLocation];
202   utilities.accumulatedTrackings  ...
203     = [utilities.accumulatedTrackings; trackedLocation];
204 end
205
206 %% computeInitialLocation
207
208 function loc = computeInitialLocation(param, detectedLocation)
209   if strcmp(param.initialLocation, 'Same as first detection')
210     loc = detectedLocation;
211   else
212     loc = param.initialLocation;
213   end
214 end
215
216 %% createUtilities
217
218 function utilities = createUtilities(param)
219   % Create System objects for reading video, displaying video, extracting
220   % foreground, and analyzing connected components.
221   utilities.videoReader = vision.VideoFileReader('singleball.mp4');
222   utilities.videoPlayer = vision.VideoPlayer('Position', [100,100,500,400]);
223   utilities.foregroundDetector = vision.ForegroundDetector(...
224     'NumTrainingFrames', 10, 'InitialVariance', param.segmentationThreshold);
225   utilities.blobAnalyzer = vision.BlobAnalysis('AreaOutputPort', false, ...
226     'MinimumBlobArea', 70, 'CentroidOutputPort', true);
227
228   utilities.accumulatedImage      = 0;
229   utilities.accumulatedDetections = zeros(0, 2);
230   utilities.accumulatedTrackings  = zeros(0, 2);
231 end
232
233 end
```

Listing 18 – kalmanFilterForTracking.m