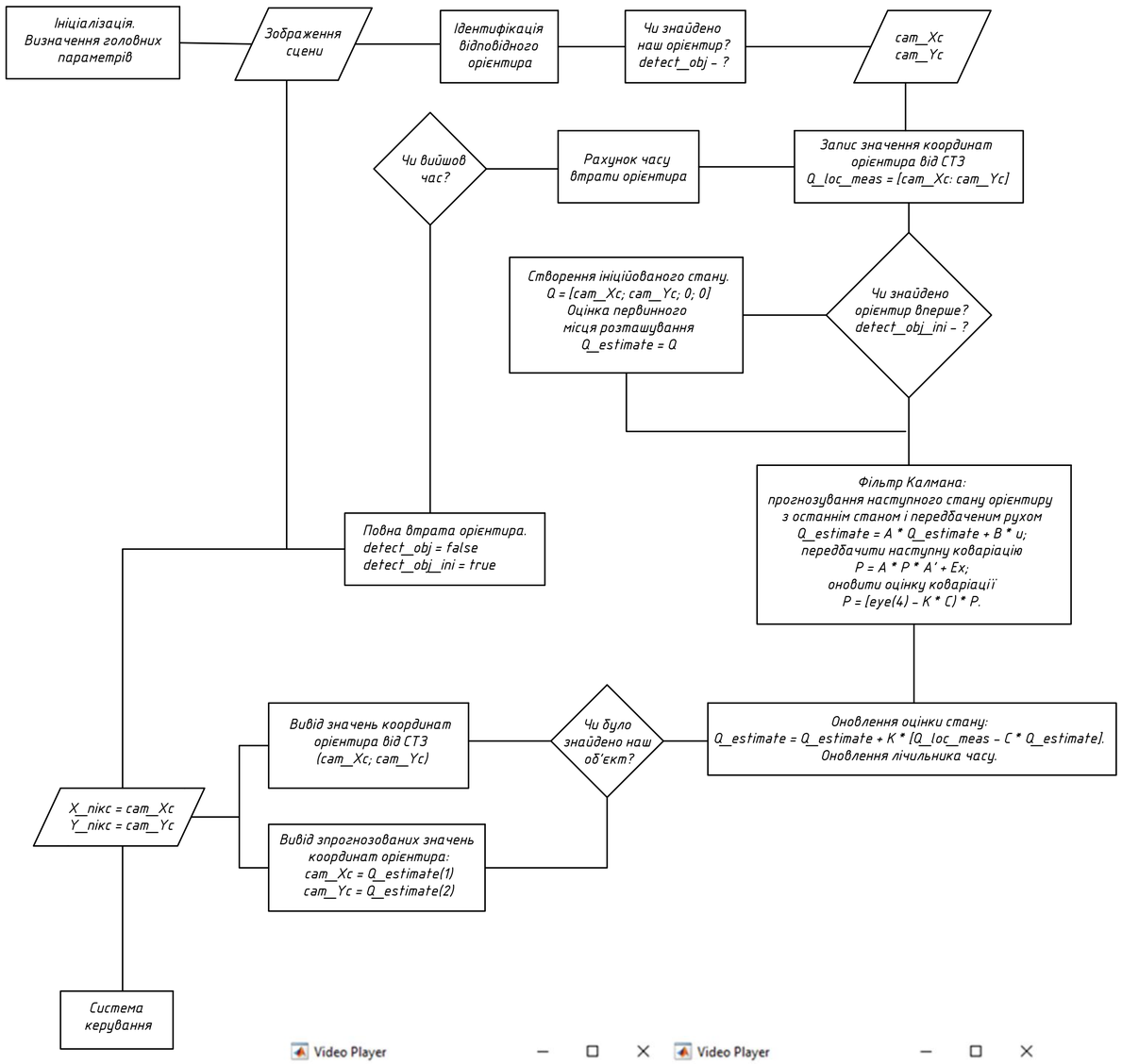

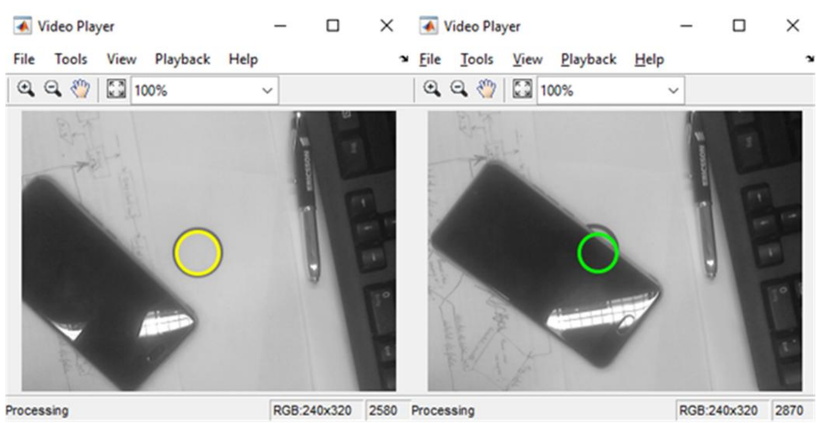

Результат роботи СТЗ з фільром Калмана: жовте коло - результат роботи СТЗ, зелене - прогнозування за фільтром Калмана

Вихідний код Matlab застосування фільтру Калмана для системи технічного зору

```
clear all;
close all;
clc;
%% INITIALISATION ____________________________________________________________________________________________
%define main variables
dt = 1;  %наша частота дискретизації (для прикладу роботи алгоритму = 1)
u = .01; % величина прискорення, береться від IHC; для прикладу роботи алгоритму було взято 0.01
HexAccel_noise_mag = .1; % технологічнийшум: мінливість в тому, як швидко орієнтир прискорюється
tkn_x = 1;  % вимірювальний шум у горизонтальному напрямку (осі x).
tkn_y = 1;  % вимірювальний шум у горизонтальному напрямку (осі y).
Ez = [tkn_x 0; 0 tkn_y];
Ex = [dt^4/4 0 dt^3/2 0; ...
   0 dt^4/4 0 dt^3/2; ...
   dt^3/2 0 dt^2 0; ...
   0 dt^3/2 0 dt^2].*HexAccel_noise_mag^2; % Ex перетворює технологічний шум (stdv) в матрицю коваріації
P = Ex; % оцінка вихідної дисперсії положення орієнтира (матриця коваріації)
%% рівняння оновлення в 2-D! (Коефіцієнт матриць): модель на основі фізики, для якої ми очікуємо, що орієнтир буде [перехідний стан (стан + швидкість)] + [контроль входу
(прискорення)]
A = [1 0 dt 0; 0 1 0 dt; 0 0 1 0; 0 0 0 1]; %оновлення матриці стану
B = [(dt^2/2); (dt^2/2); dt; dt];
C = [1 0 0 0; 0 1 0 0];  % це наша функція вимірювання C, яку ми застосовуємо до оцінки стану Q, щоб отримати наше очікування наступного / нового вимірювання
%% ініціювати змінні результату
% Ініціалізувати для швидкості
Q_loc_meas = []; % значення координат орієнтира від СТЗ
%% ініціалізувати оцінювальні змінні
Q_loc_estimate = []; %  позиція
vel_estimate = []; % швидкість
P_estimate = P;
predic_state = [];
predic_var = [];
webcamlist
cam = webcam('Logitech')
cam.Resolution = '320x240';
videoFrame = snapshot(cam);
frameSize = size(videoFrame);
videoPlayer = vision.VideoPlayer('Position', [100 100 [frameSize(2), frameSize(1)]+30]);
runLoop = true;
numPts = 0;
time = 0;
Center_null = [1 1];
Radii_null = 1;
Metric = 1;
detected_someth = false;  %чи було знайдено будь-який схожий об'єкт інтересу?
detected_obj = false;  %чи було знайдено наш об'єкт інтересу?
detected_obj_ini = true;  %чи було знайдено наш об'єкт інтересу раніше *true - NO?
%%%%MAIN LOOP_________________________________________________________________________________________________
while runLoop
    videoFrame = snapshot(cam);
   sceneImage = rgb2gray(videoFrame);
   radiusRange = [10 30];
   [centers,radii, metric] = imfindcircles(sceneImage,radiusRange, 'ObjectPolarity','bright'); %для тестування використовуємо простий алгоритм ідентифікації орієнтира -
знаходження кола
try
   Center = centers(1,:);
   Radii = radii(1,:);
   Metric = metric(1,:);
   detected_someth = true;
catch
   detected_obj = false;
end
if detected_someth == true
   if Metric > 0.4
      cam_Xc = Center(1);
      cam_Yc = Center(2);
      detected_obj = true;
      Radii_c = Radii;
   else
      detected_obj = false;
   end
else
   detected_obj = false;
end
if detected_obj == true
   if detected_obj_ini == true
       Q= [cam_Xc; cam_Yc; 0; 0]; % ініційований стан - він має чотири компоненти: [positionX; позиція Y; швидкість X; швидкість Y] орієнтира
Q_estimate = Q;  % оцінка первинного місця розташування, де знаходиться орієнтир (що ми оновлюємо)
      time = 0;
      detected_obj_ini = false;
   end
Q_loc_meas = [cam_Xc; cam_Yc];
end
if detected_obj == false
        time = time + 2;
      if time >= 60
         detected_obj_ini = true;
         detected_obj = false;
      end
end
if detected_obj_ini == false
   % Прогнозування наступного стану орієнтиру з останнім станом і передбаченим рухом.
   Q_estimate = A * Q_estimate + B * u;
   %predic_state = [predic_state; Q_estimate(1)] ;
   % передбачити наступну коваріацію
   P = A * P * A' + Ex;
   %predic_var = [predic_var; P] ;
   % Kalman Gain
   K = P*C'*inv(C*P*C'+Ez);
   % Оновити оцінку стану.
   if detected_obj == true
      Q_estimate = Q_estimate + K * (Q_loc_meas - C * Q_estimate);
      time = 0;
      sceneImage = insertShape(sceneImage, 'circle', [[cam_Xc cam_Yc] Radii], 'LineWidth', 3);
   else
      cam_Xc = Q_estimate(1); cam_Yc = Q_estimate(2);
      sceneImage = insertShape(sceneImage, 'circle', [[cam_Xc cam_Yc] Radii], 'LineWidth', 3, 'Color', 'g');
   end
    % Оновити оцінку коваріації.
   P =  (eye(4)-K*C)*P;
else
   cam_Xc = -1; cam_Yc = -1;
   sceneImage = insertShape(sceneImage, 'circle', [Center_null Radii_null], 'LineWidth', 3);
end
    step(videoPlayer, sceneImage);
   runLoop = isOpen(videoPlayer);
end
```

Изм.	Лист	№ докум.	Подп.	Дата	ВЛ61М04 16.73.00.02			
Разраб.		Котвицький		09.05.2019	Алгоритм роботи фільтру Калмана в системі технічного зору	Лит.	Масса	Масштаб
Пров.								
Т. контр.						Лист	Листов 1	
Нач.отд.						НТУУ "КПІ ім. І. Сікорського", ФАКС		
Н. контр.								
Утв.		Збруцький						

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал | Формат А1