Assignment A10: Tracking

CS 5320/6320
Spring 2016

Assigned: 23 March 2016

Due: 20 April 2016

For this problem, handin a lab report A10.pdf (include name, date, assignment and class number in pdf) which develops and studies tracking.

1. Develop a background subtraction tracking method for the image sequence in the class data/A10/background directory. There is a red bead that moves through the image, and you should study how this can be tracked using the difference between each image and the background image.

2. Develop a Kalman filter to track the ball falling off the cliff (image sequence is in class data/A10/cliff directory). Assume that each pixel is 1/6 m in height, and that the frame rate is 1/30 sec. Use the results to produce an estimate of the gravitational acceleration constant for the sequence. Note that this requires a function (i.e., CS5320_detect_red_ball below) to extract the position of a red ball in an image).

3. **CS6320**: Compare the use of 3 distance measures for red ball tracking (i.e., look at thresholds on the distance for the different measures and the resulting classification):

- Euclidean distance
- Mahalanobis distance
- Gaussian probability

Develop the Matlab functions described below.

You should handin the report A10.pdf as well as the source code developed in the study. The code should conform to the style requested in the class materials.

In addition, please turn in a hardcopy of the report in class before the start of class on April 20, 2016.

Write a lab report in the format (please do not deviate from this format!) described in the course materials.

```
function t_im = CS5320_background_sub_tracking(im)
% CS5320_background_sub_tracking - track by background difference
% On input:
%     im (struct array): image sequence (p images)
%        im(k).im (mxnxd array): k_th image
% On output:
%     t_im (mxnxp array): subtracted images (image from average)
% Call:
%     t_im = CS5320_background_sub_tracking(ims);
% Author:
%     <Your name>
%     UU
%     Spring 2016
%

function p_i = CS5320_process_step(p_im1,D,R)
% CS5320_process_step - one step in linear process
% On input:
%     p_im1 (nx1 vector): state vector at (i-1)_th step
%     D (nxn array): linear process matrix
%     R (nxn array): covariance matrix for process
% On output:
%     p_i (nx1 vector): state vector at i_th step
% Call:
%     p = CS5320_process_step(p,D,R);
% Author:
%     <Your name>
%     UU
%     spring 2016
%
```

```
function y = CS5320_observe(x,M,Q)
% CS5320_observe - linear observation (sensor)
% On input:
%     x (nx1 vector): state vector
%     M (kxn array): observation matrix
%     Q (kxk array): observation covariance
% On output:
%     y (kx1 vector): observation (sensor) vector
% Call:
%     y = CS5320_observe(x,M,Q);
% Author:
%     <Your name>
%     UU
%     spring 2016
%

function trace = CS5320_const_vel(x0,y0,vx0,vy0,del_t,max_t,R)
% CS5320_const_vel - constant velocity forward simulation
% On input:
%     x0 (float): initial x location
%     y0 (float): initial y location
%     vx0 (float): x speed
%     vy0 (float): y speed
%     del_t (float): time step
%     max_t (float): maximum time for simulation
%     R (4x4 array): covariance of process
% On output:
%     trace (kx4 array): state trace
%      trace(i,:): state vector at step i
% Call:
%     tr = CS5320_const_vel(0,0,1,0,0.1,2,R);
% Author:
%     <Your name>
%     UU
%     Spring 2016
%

function trace = CS5320_const_acc(x0,y0,vx0,vy0,ax0,ay0,del_t,max_t,R)
% CS5320_const_acc - constant acceleration forward simulation
% On input:
```

```
%     x0 (float): initial x location
%     y0 (float): initial y location
%     vx0 (float): initial x speed
%     vy0 (float): initial y speed
%     ax0 (float): x acceleration
%     ay0 (float): y acceleration
%     del_t (float): time step
%     max_t (float): maximum time for simulation
%     R (6x6 array): covariance of process
% On output:
%     trace (kx4 array): state trace
%       trace(i,:): state vector at step i
% Call:
%     tr =
%     CS5320_const_acc(0,0,0,0,0,-9.8,0.1,2,[0.1,0.1;0.1,0.1;0,0]);
% Author:
%     <Your name>
%     UU
%     Spring 2016
%

function [x_i_plus,Sigma_i_plus] =
CS5320_Kalman_step(x_im1,Sigma_im1,...
    D,R,M,Q,y)
% CS5320_Kalman_step - one step in Kalman filter update
% On input:
%     x_im1 (nx1 vector): state vector at step i-1
%     Sigma_im1 (nxn array): state covariance array
%     D (nxn array): linear process matrix
%     R (nxn array): process covariance matrix
%     M (kxn arraay): linear observation matrix
%     Q (kxk array): observation covariance array
%     y (kx1 vector): observation vector
% On output:
%     x_i_plus (nx1 vector): updated state vector
%     Sigma_i_plus (nxn array): state covariance matrix
% Call:
%     [x,Sigma] = CS5320_Kalman_step(x,Sigma,D,R,M,Q,y);
% Author:
%     <Your name>
%     UU
```

```
%     Spring 2016
%

function [ta,ty,te] =
CS5320_const_acc_Kalman(x0,y0,vx0,vy0,ax0,ay0,...
    del_t,max_t,R,Q)
% CS5320_const_acc_Kalman - simulation of constant acceleration
% scenario
% On input:
%     x0 (float): initial x locaiton
%     y0 (float): initial y location
%     vx0 (float): initial x speed
%     vy0 (float): initial y speed
%     ax0 (float): x acceleration
%     ay0 (float): y acceleration
%     del_t (float): time step
%     max_t (float): max time for simulation
%     R (6x6 array): process covariance matrix
%     Q (2x2 array): observation covariance matrix
% On output:
%     ta (px6 array): actual state values for p steps
%     ty (px2 array): observation values for p steps
%     te (px6 array): estimated state values for p steps
% Call:
%     [ta,ty,te] = CS5320_const_acc_Kalman(0,0,0,0,0,-9.8,0.1,3,R,Q);
% Author:
%     <Your name>
%     UU
%     Spring 2016
%

function [row,col] = CS5320_detect_red_ball(im,model)
% CS5320_detect_red_ball - find red ball in image
% On input:
%     im (mxnx3 array): rgb image
%     model (1x3 vector): rgb model
% On output:
%    row (int): row of red ball centroid
%    col (int): col of red ball centroid
% Call:
%    [rc,cc] = CS5320_detect_red_ball(Falling_ball(1).cdata,[230,30,30]);
```

```
% Author:
%     <Your name>
%     UU
%     Spring 2016
%

function [ty,te] = CS5320_red_ball_Kalman(im_seq,ax,ay,del_t,R,Q)
% CS5320_red_ball_Kalman - track falling red ball
% On input:
%     im_seg (struct vector): image sequence of falling ball (p
%     images)
%     ax (float): acceleration in x
%     ay (float): acceleration in y
%     del_t (float): time step
%     R (6x6 array): process covariance matrix
%     Q (2x2 array): observation covariance matrix
% On output:
%     ty (px6 array): observation values for p steps
%     te (px6 array): estimated state values for p steps
% Call:
%     R = 0.0001*eye(6,6);
%     R(5:6,5:6) = 0;
%     Q = eye(2,2);
%     [ty,te] = CS5320_red_ball_Kalman(Falling_Ball,0,-9.8,1/30,R,Q);
% Author:
%     <Your name>
%     UU
%     Spring 2016
%
```