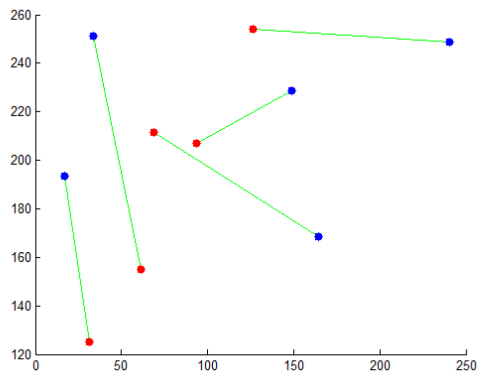
Machine Vision Coursework 2

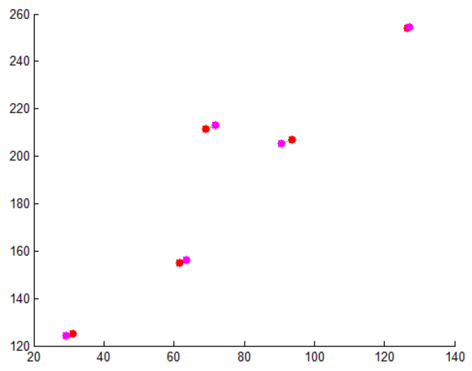
Homographies Part I

A)

This plot shows several pre-defined Cartesian points, and the results of their projective transformation (homography). Routine: transform the Cartesian points to homogeneous representation, and multiply the results by a homography matrix then convert back to Cartesian points.



This plot shows the effect of the estimated homography matrix: two sets of points are matched together. Acquiring the two set of points, the best estimated homography matrix can be calculated by solving a Ah = 0 equation. Due to some noise and error in calculation, the results couldn't be matched exactly.



Scale ambiguity

Multiplying the homography by a constant factor will not cause big change to the results. The mathematical reason is that the homography is not scale sensitive, due to the homogeneous calculation of the points. Converting homogeneous points to Cartesian points needs to eliminate the extra elements in coordinates, where the scaling of homography will be eliminate as well.

Exact mapping of pairs of four points

The routine calcBestHomography make sure the accuracy of mapping pairs of points. Firstly, transfer the two set of Cartesian points in to homogeneous representations, then construct the matrix A. Because normally the transformation of two sets of points is projective transformation (homography), there is no constraint of extrinsic matrix. So we have to solve Ah = 0 by taking the last column of the V in svd of matrix A. A minimal solution would require 3 pairs of points. So four points or five points can guarantee the accuracy of mapping of pairs of points.

Document TO DOs

```
%first turn points to homogeneous
```

Add ones to the last row of the matrix.

```
%then construct A matrix which should be (10 x 9) in size
```

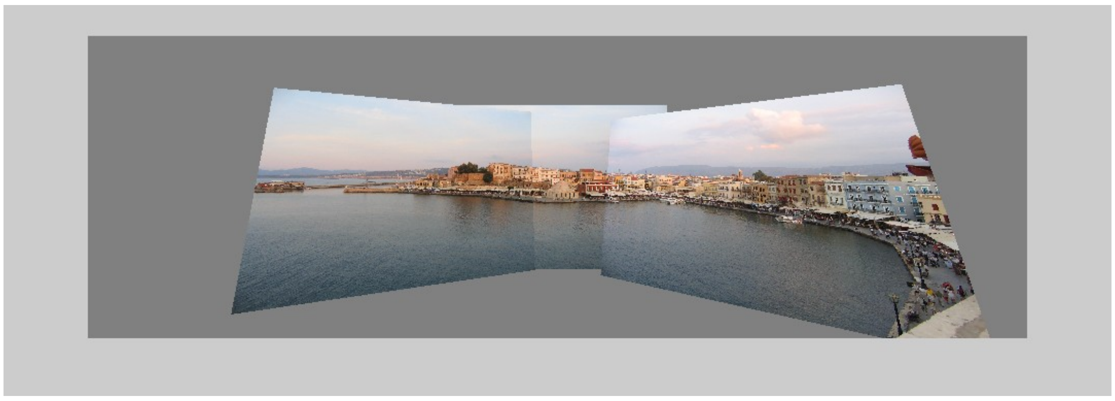
The pattern of matrix A can be seen so for even and odd row add in different elements.

```
%solve Ah = 0 by calling h = solveAXEqualsZero(A);
```

Taking the last column of V in svd is to solve the minimum direction problem. Reshape the result can acquire the extrinsic matrix.

B)

To achieve this panorama, two homographies are computed by using several feature points. These homographies can help to map pixels of two images to the middle main image. A much faster way for mapping is created in my code.



Document TO DOs

All the TO DOs are almost the same as the previous script, except this time the complexity of mapping pixels needs to be considered. So I take advantage of matrix calculation to map pixels from image1 to 2 and 3. I extract all the coordinates of pixels in image 1 and multiply them by homography. And therefore the speed of running the script is increased.

Homographies Part II

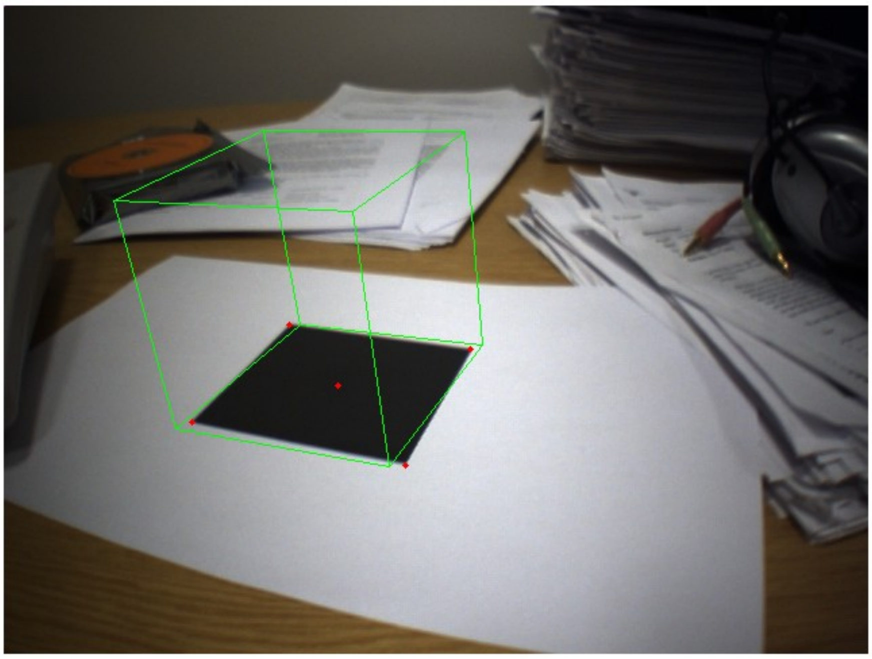
C)

The result shows the re-estimate T matrix (Extrinsic matrix) by taking intrinsic matrix K and one set of Cartesian points and its transformed version. The results may vary due to the affect of the noise. The routine of estimating the Extrinsic matrix T is slightly different from the method in the textbook, because the homography contains zero element in Omega (rotation matrix).

```
I =

    0.9851   -0.0492    0.1619   46.0000
   -0.1623   -0.5520    0.8181   70.0000
    0.0490   -0.8324   -0.5518  500.8900
         0         0         0    1.0000


IEst =

    0.9816   -0.0412    0.1865   46.5470
   -0.1779   -0.5522    0.8145   71.7506
    0.0694   -0.8327   -0.5493  503.6300
         0         0         0    1.0000
```

D)

Using the same algorithm to get the estimated T matrix (Extrinsic matrix), standard 3D wire-frame cube can be projected to 2D image. Some drawback may lie in the inaccurate of points matching.



QUESTIONS in script:

1. Do the results look realistic?
Not so realistic because of estimating error.

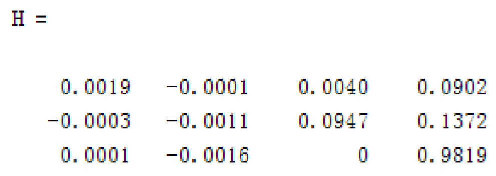
2. If not, then what factors do you think might be causing this?
One factor may be that slight computing error (when computing SVD and estimate PlanePose), another factor may be the lack of shading and

shadowing.

Document TO DOs

Routine projectiveCamera:
This routine mainly transfer Cartesian points to homogenious points and multiply them by extrinsic matrix and intrinsic matrix, so that to project points to 2D image.

Routine estimatePlanePose:
One trick in this routine is to calculate the matrix R: R_hat = U*[1,0;0,1;0,0]*V'. Because the estimated homography contains zero element in the rotation part, the method in textbook or slides cannot be used. Following the instruction in the comments, R_hat can be calculated by using first two column of the homography.



```
H =

    0.0019   -0.0001    0.0040    0.0902
   -0.0003   -0.0011    0.0947    0.1372
    0.0001   -0.0016         0    0.9819
```

Another thing changed is the structure of matrix A. It is larger because more extrinsic elements are needed.

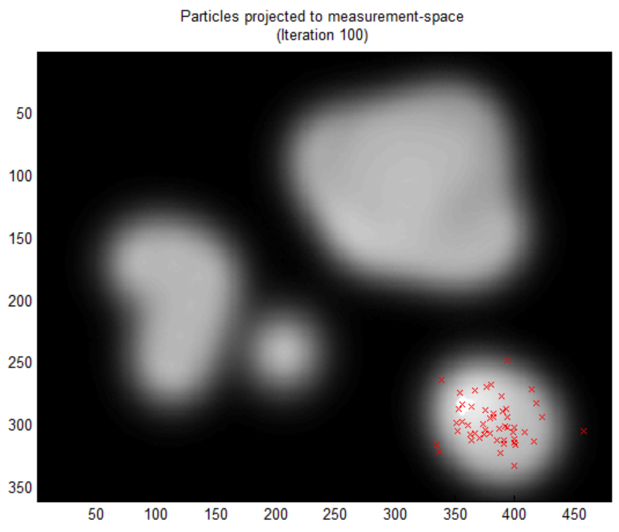
```
%TO DO Estimate the translation t by finding the appropriate scaling factor k and applying it to the third colulmn of H
k = sum(sum(R_hat(:,1:2)./H(:,1:2)));
t = k.*H(:,4); %fourth column
```

Here thought the comment requires to apply the k to the third column, I apply it to the fourth column because that's where the translation factor t located in my version.

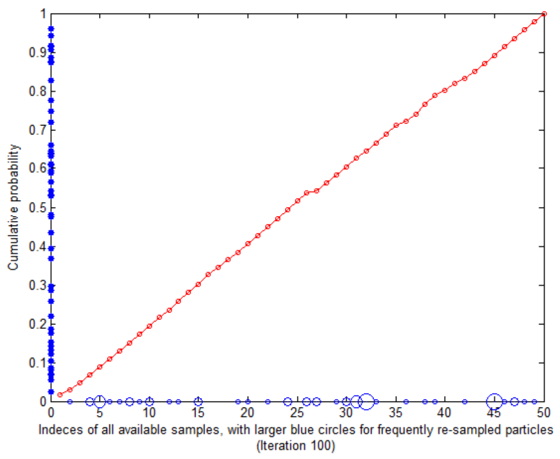
Condensation

E)

In this algorithm, by normalizing and updating the weight, the particles will propagate to the place that has high posterior probabilities. The result is the location of about 50 particles after 100 iterations.



This diagram shows the cumulative sum of the normalized weight. By randomly discarding the low weight, the particles can get together in place that has high posterior probabilities.



Document TO DOs

```
% TO DO: normalize the weights (may be trivial this time)
```

Just divide the weight by its sum can do.

```
% TO DO: compute the cumulative sume of the weights. You could refer to
% the MATLAB built-in function 'cumsum'.
```

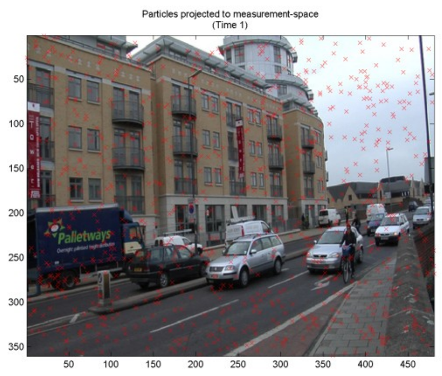
Use the function: `cum_hist_of_weights = cumsum(weight_of_samples);`

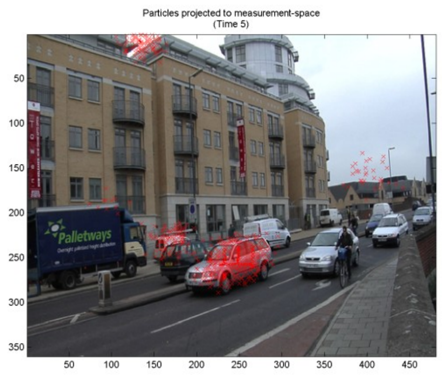
```
% TO DO: Incorporate some noise, e.g. Gaussian noise with std 10,
% into the current location (particles_old), to give a Brownian
% motion model.
```

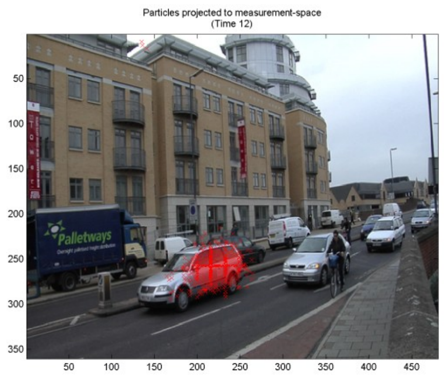
Use this normrnd(0,10,[1,col]) to create the noise. But the noise should be added to particles_new rather than multiplied.

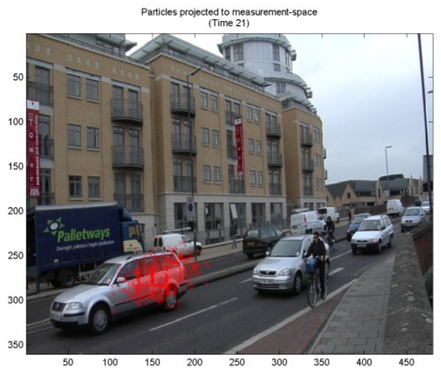
F)

A few frames are selected to show the result, which tracks the given shape (wheel) with 2 dimension state w. That is, the tracking will only happen in current frame and haven’t consider the previous state w (previous frame). The particles may be located on several place because where has similar geometry of the given shape.

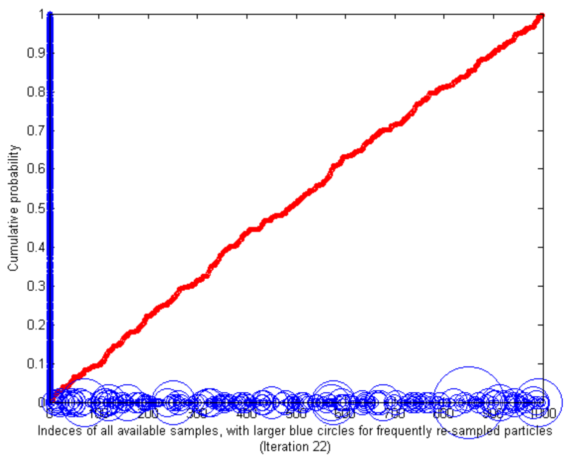






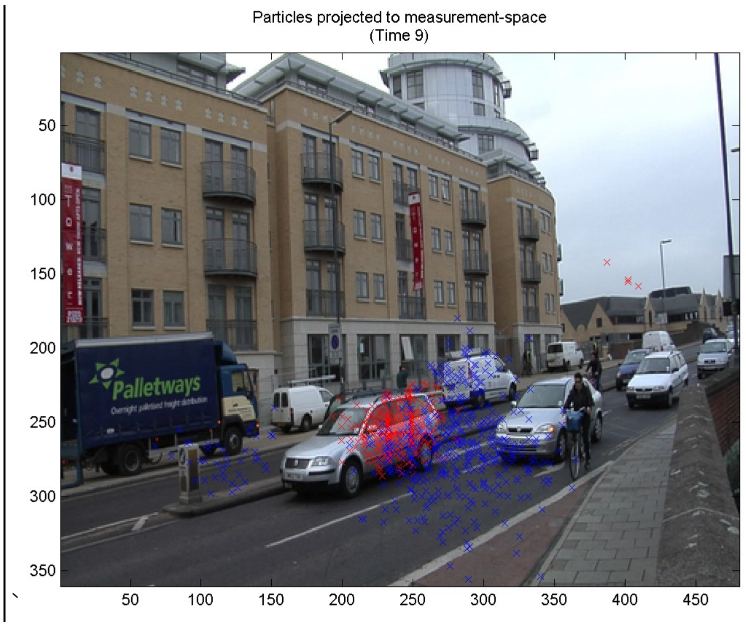


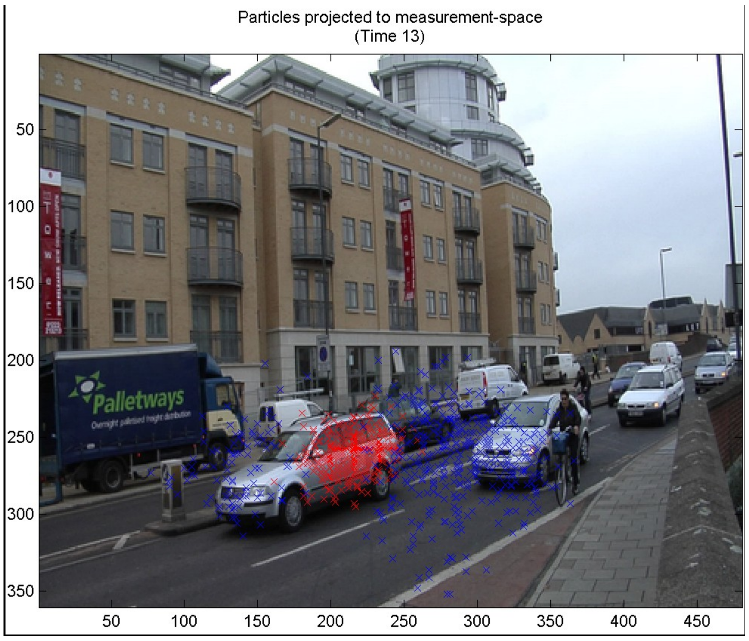
Again the cumulative sum of the normalized weight is shown.



I used four dimensions of state w and try to track two objects, one is the wheel and the other is the pedestrian on the bike.

(But I failed to use four dimension of state w that consider the previous state (w-1))





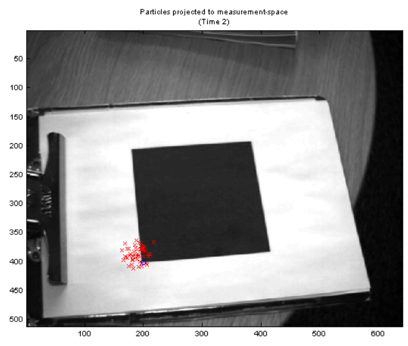
Document TO DOs

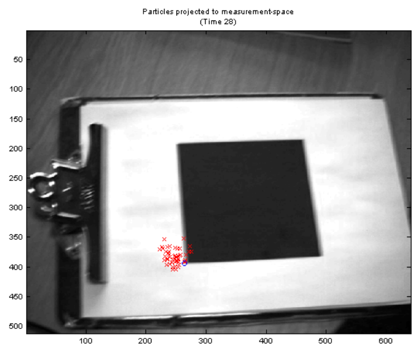
The same as the previous script.

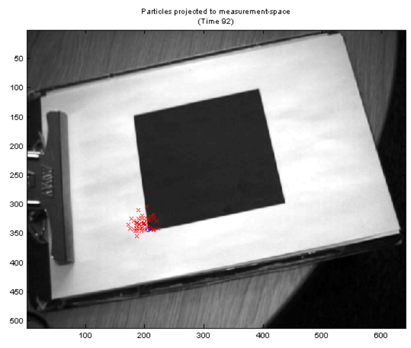
Combining Tracking and Homographies

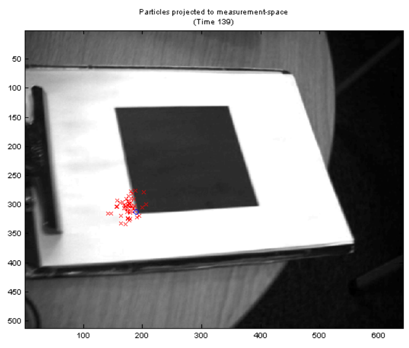
G)

These selected frames show the tracking of one corner (ll) with the moving of the pattern. It can be seen that the particles get together in the corner, yet sometimes there will be some offset due to the error and noise.

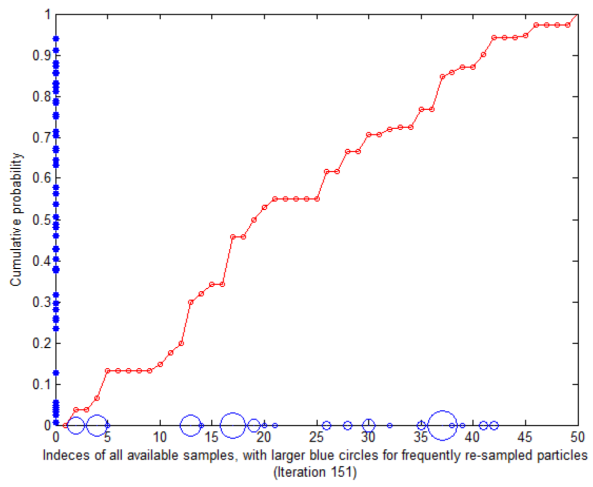








The cumulative sum of the normalized weight is plotted for every frame as well, showing the updated weights.



Document TO DOs

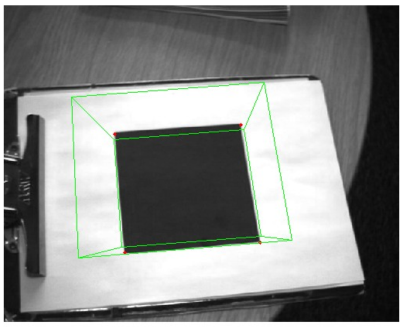
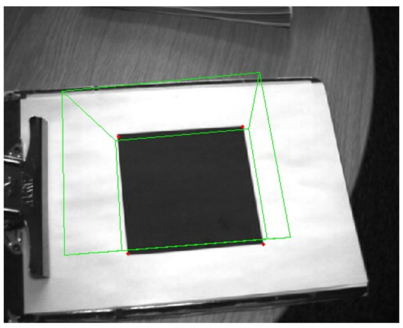
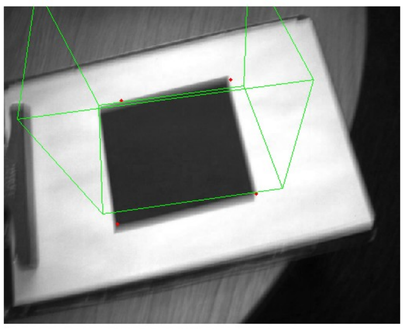
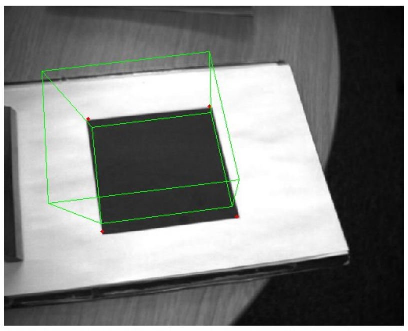
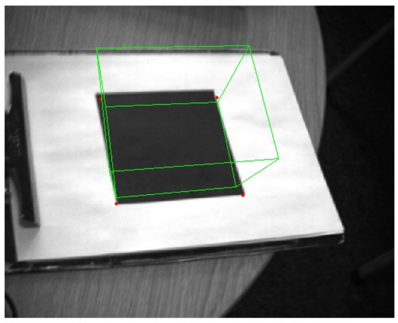
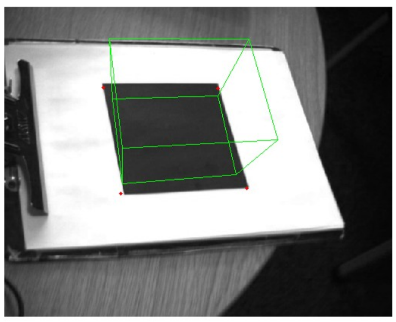
The same as the relevant script, except

```
% TO DO: Compute the coordinate of the "best" (i.e. MAP) location by
% computing the weighted average of all the particles:

    weightedAve = weight_of_samples'*particles_new;
```

Just use the matrix product can get the result.

H)

These selected frames show the results of tracking four corners and the augmented cube in the 2D frame, by computing the relevant homographies. There are some drawbacks (for example sometimes the points do not match the edges), I think it is because of the double errors in both computing the homographies and tracking the corners.s

Document TO DOs

The same as the previous script.

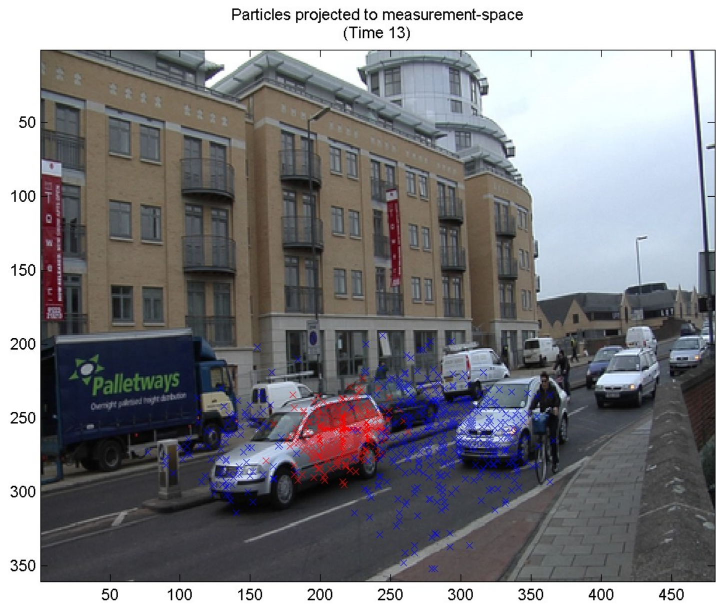
QUESTIONS in script:

Mention at least two actions or changes we could make to improve the results:

1. One action is to consider multiple state w, (for example (w-1) and (w-2)), using Kalman filter or calculating the Chapman-Kolmogorov relation. This method can take the previous posterior into consideration, so that the tracking

point will not have large offset between nearby frames.

2. Tracking four corners at the same time, so that to avoid independent tracking of each corner. May be multiple state can also be used but in one frame (state w1, w2, w3 and w4), just like to track both wheel and person in a bike in my previous attempt(see below). But may be in this way the accuracy cannot be guaranteed.



Extra Credit

Reduce the search space where particles can land by using an edge-detector.
I used my sobel edge detector function. And I tried a few but it seems that the weight cannot be calculated in the output of edge detector.