EE417

POST-LAB #6 REPORT

Optical Flow

Optical flow is the pattern of apparent motion of objects in a visual scene caused by the relative motion between an observer and a scene. In this lab we will use some methods to detect optical flow. By detecting of optical flow by use of sequential frames from a scene, we can measure the velocity of an object.

- Translational model

$$I_1(\mathbf{x}_1) = I_2(\mathbf{x}_1 + \Delta\mathbf{x})$$

- Small baseline

$$I(\mathbf{x}(t), t) = I(\mathbf{x}(t) + \mathbf{u}dt, t + dt)$$

- RHS approx. by first two terms of Taylor series

$$\nabla I(\mathbf{x}(t), t)^T \mathbf{u} + I_t(\mathbf{x}(t), t) = 0$$

We say that a point (say x1) will be correspond to x2 in another sequential frame. If they are the same points from an object, their intensity values should be the same. Based on this interpretation, we also say that it is a really small distance and time difference between consecutive frames. Then, by use of Taylor series we can deduce the equation above.

If we integrate around over image patch, it actually turns into an optimization problem which done by taking the first derivative and make it equal to 0.

$$E_b(\mathbf{u}) = \sum_{W(x,y)} [\nabla I^T(x,y,t)\mathbf{u}(x,y) + I_t(x,y,t)]^2$$

- Solve

$$\nabla E_b(\mathbf{u}) = 2 \sum_{W(x,y)} \nabla I(\nabla I^T \mathbf{u} + I_t)$$

$$= 2 \sum_{W(x,y)} \left(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \mathbf{u} + \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix} \right)$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \mathbf{u} + \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} = 0$$

$$G\mathbf{u} + \mathbf{b} = 0$$

$$G = \begin{bmatrix} \sum_{p \in W} I_x^2 & \sum_{p \in W} I_x I_y \\ \sum_{p \in W} I_x I_y & \sum_{p \in W} I_y^2 \end{bmatrix} \quad b = \begin{bmatrix} \sum_{p \in W} I_x I_t \\ \sum_{p \in W} I_y I_t \end{bmatrix}$$

$$u = [Vx; Vy] = -G^{-1} b$$

where; I_x and I_y: spatial gradients in X and Y directions
I_t: the gradient in temporal direction, V_x and V_y are velocity vectors

Procedure

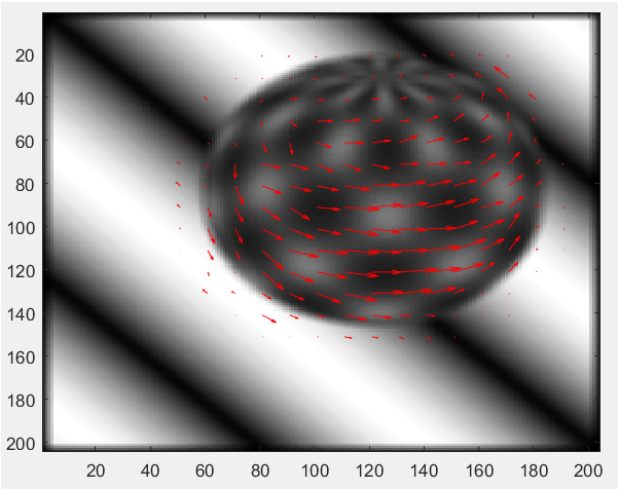
1. We will smooth the input image by an appropriate filter to reduce the noise (Gaussian and Box Filter can be used) for the further processing.
2. We will calculate spatial gradients (I_x, I_y) using Prewitt Filter kernels.

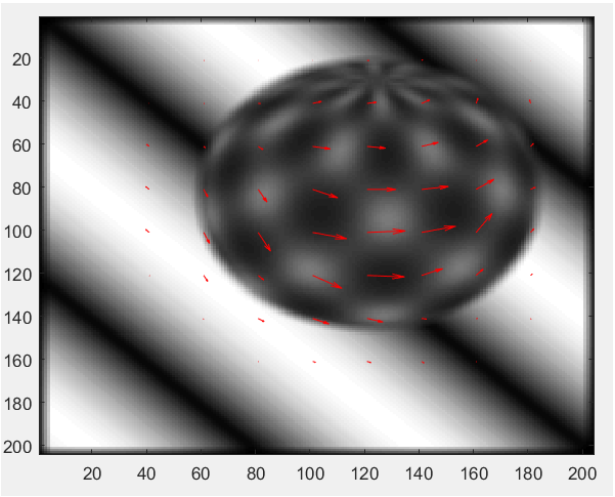
-1	0	1
-1	0	1
-1	0	1

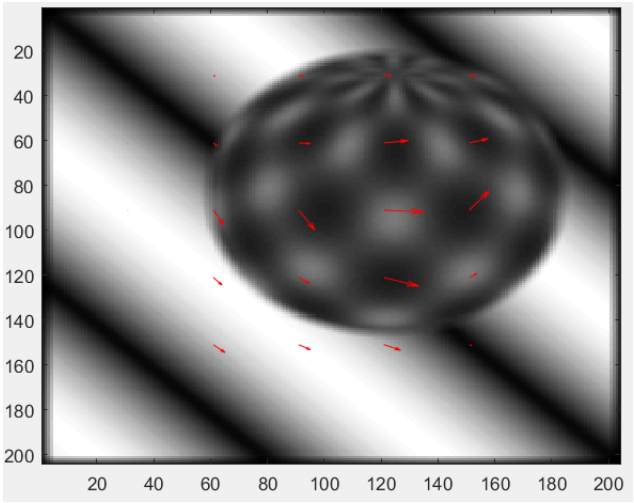
-1	-1	-1
0	0	0
1	1	1

3. Calculate the temporal gradient as ImPrev – ImCurr. That can be thought as taking derivate which shows the difference between two consecutive frames.
4. Compute eigenvalues of G.
5. If smallest eigenvalue of G is bigger than threshold, we will mark this pixel as candidate feature point that will be used to determine the Vx and Vy of the velocity vector u.

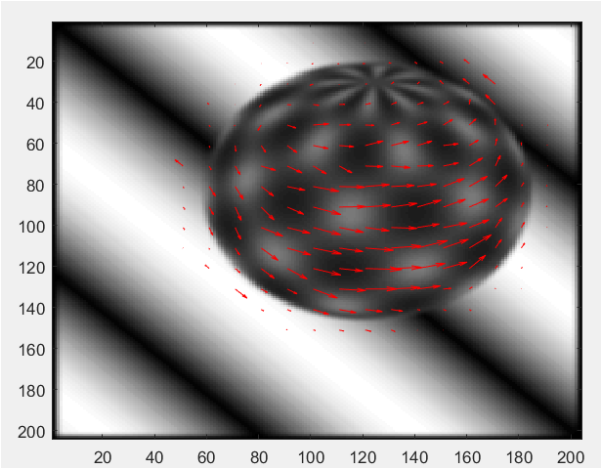
Sphere, Box Filter, K = 10, K = 20, K = 30

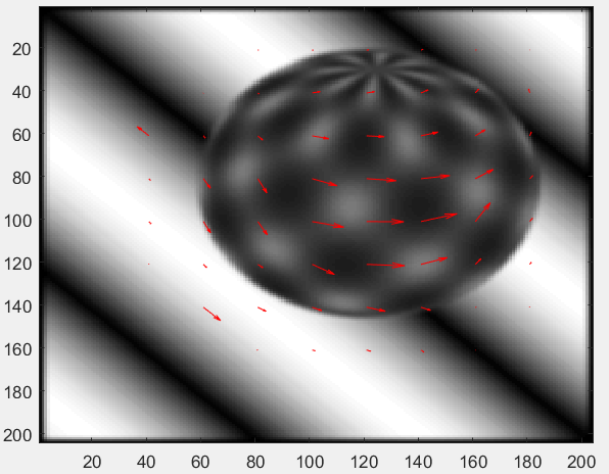


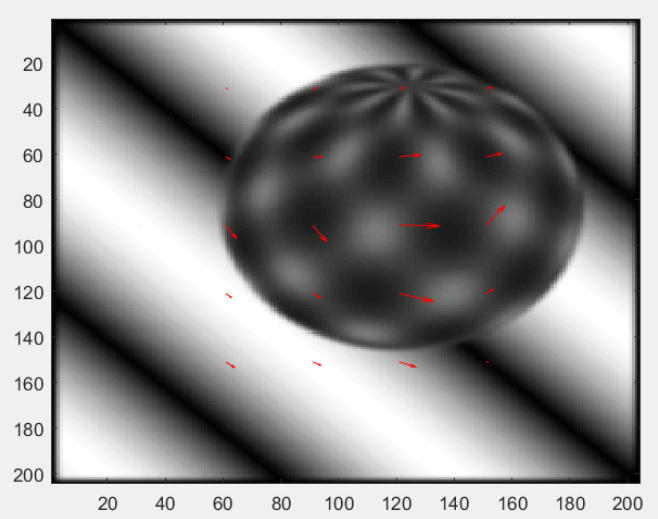




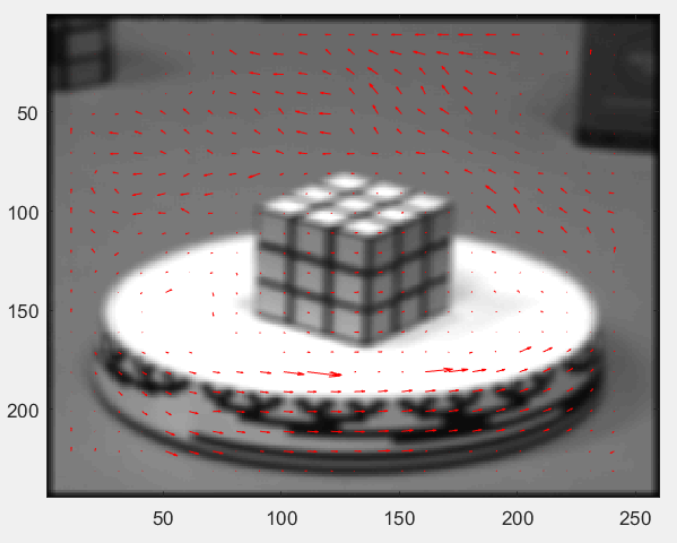
Sphere, Gaussian Filter, K = 10, K = 20, K = 30

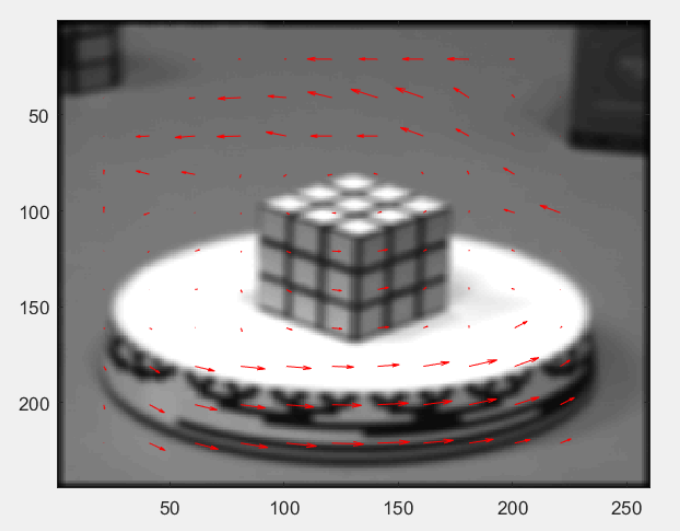


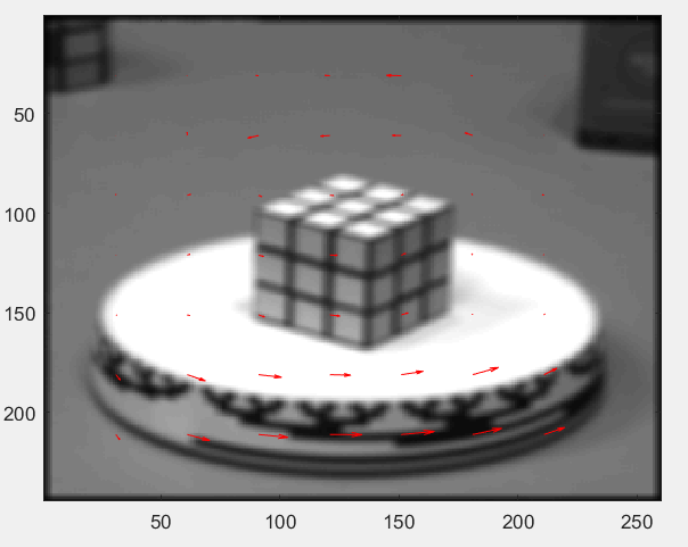




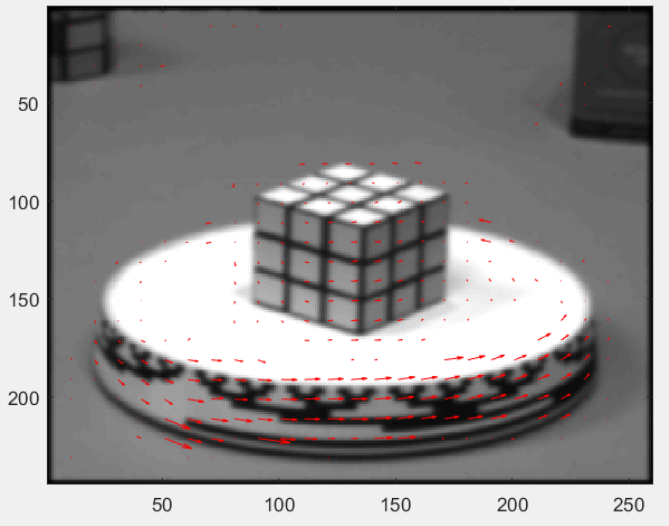
Rubic, Box Filter, K=10, K=20, K=30

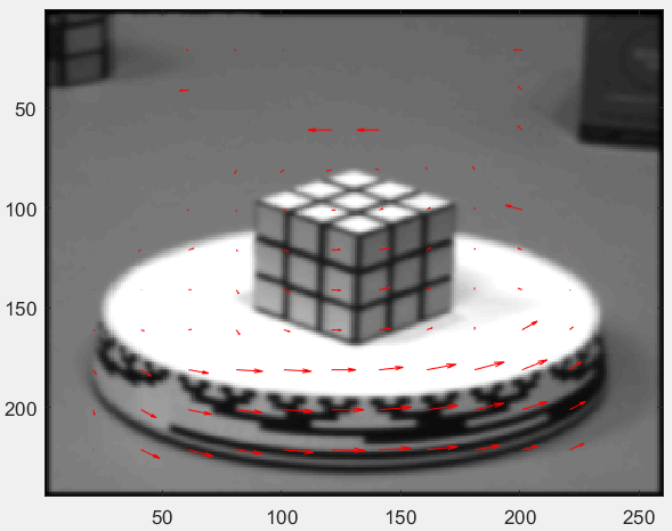


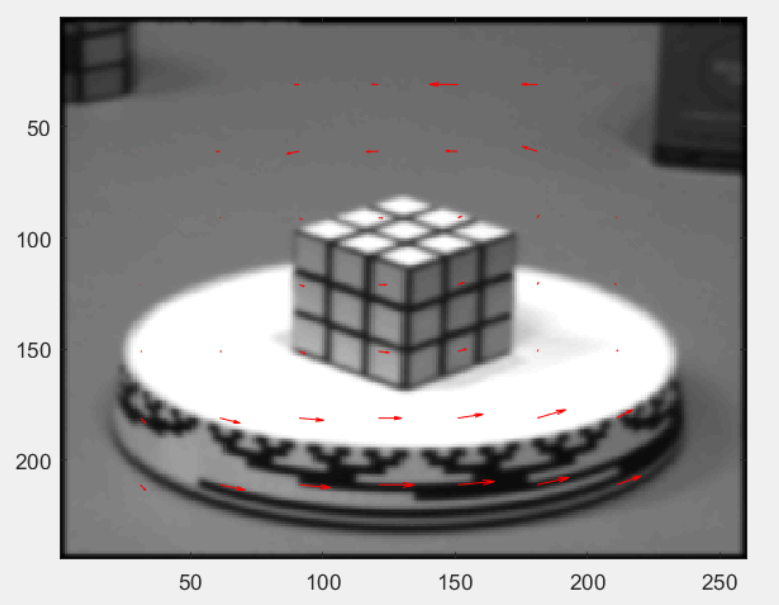




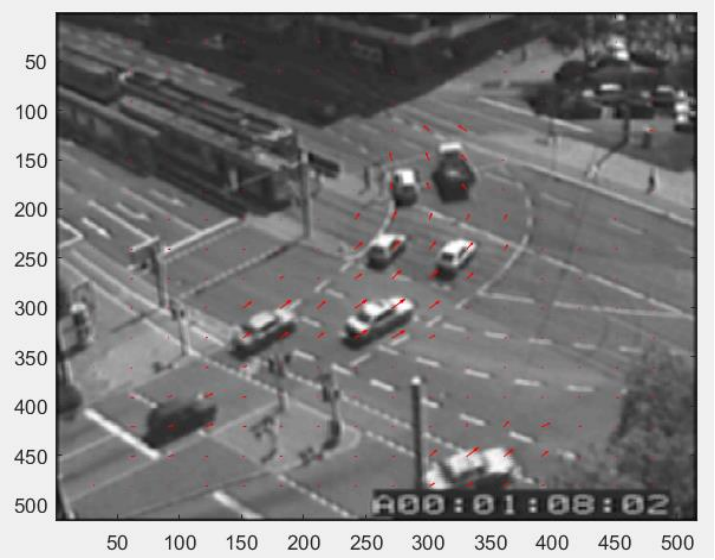
Rubic, Gaussian Filter, K = 10, K = 20, K = 30





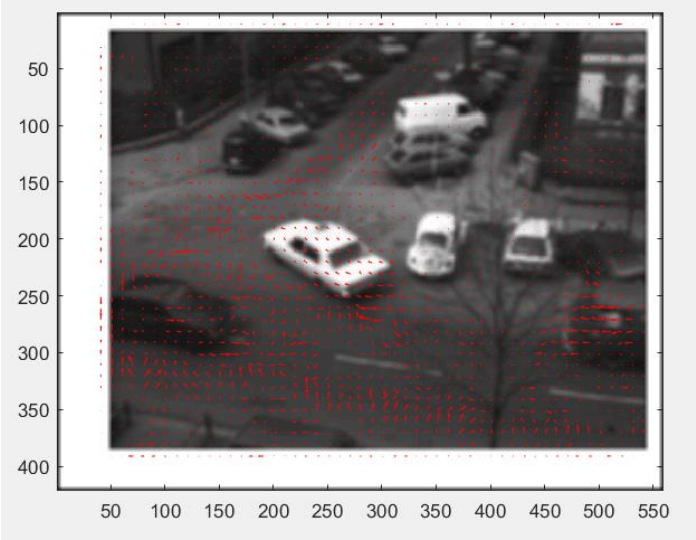


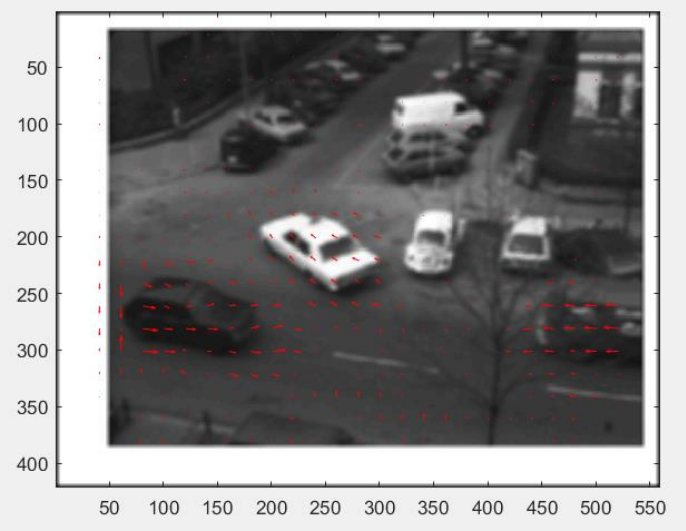
Traffic, Box Filter, K=10, K=20, K=30



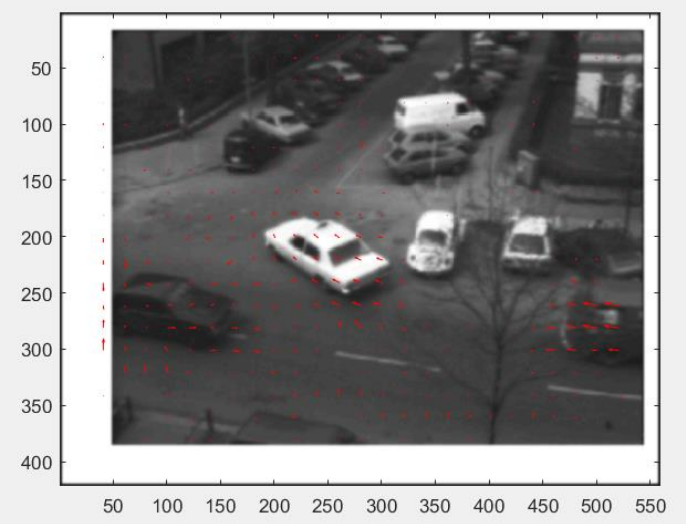
Traffic, Gaussian Filter, K=10, K=20, K=30

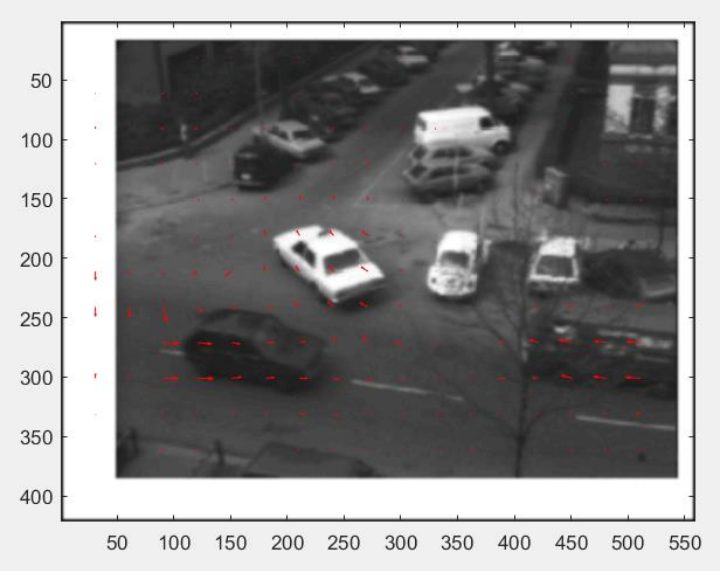
Taxi, Box Filter, K=10, K=20, K=30



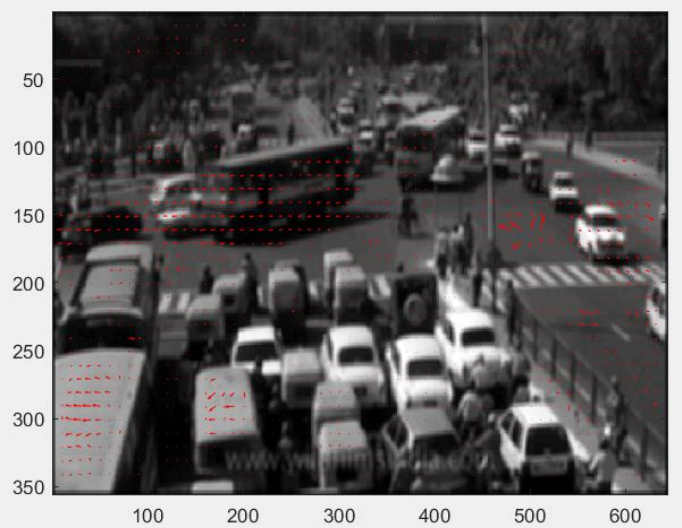


Taxi, Gaussian Filter, K=10, K=20, K=30

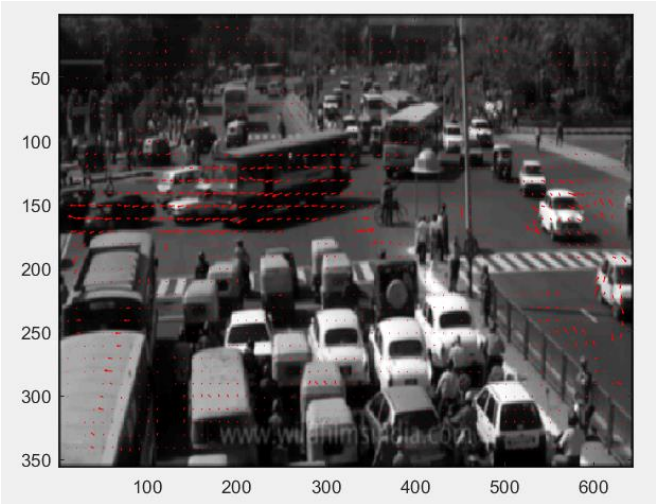


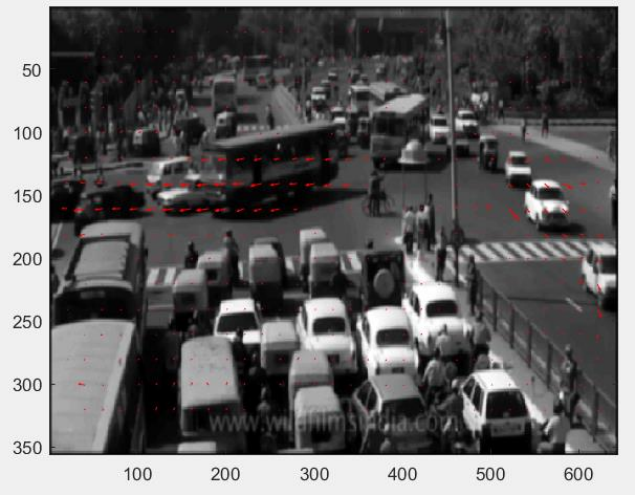


Cars1, Box Filter, K=10, K=20, K=30

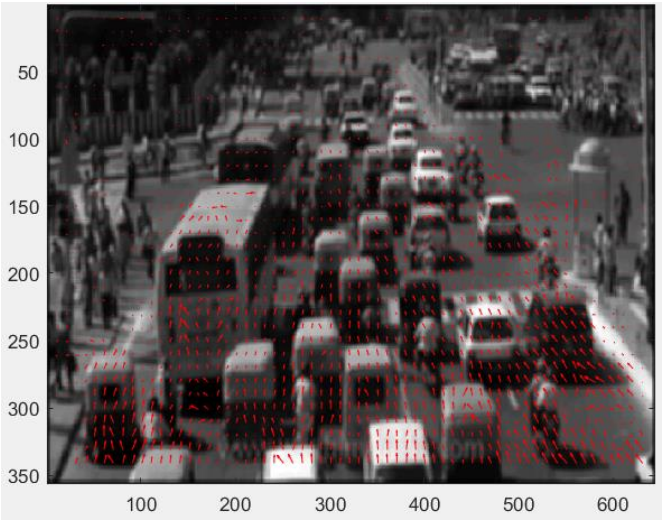


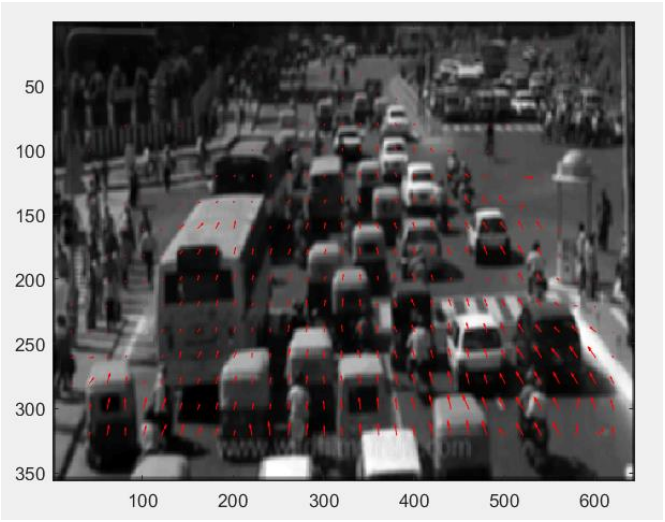
Cars1, Gaussian Filter, K=10, K=20, K=30

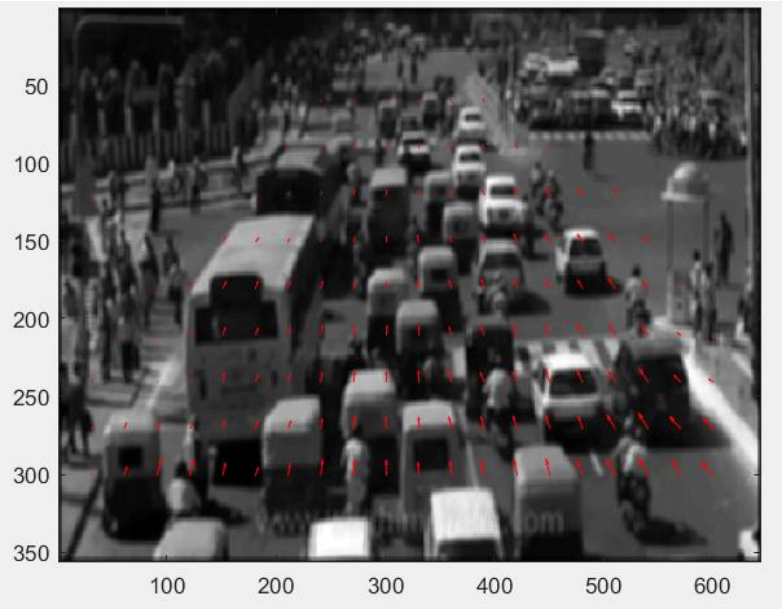




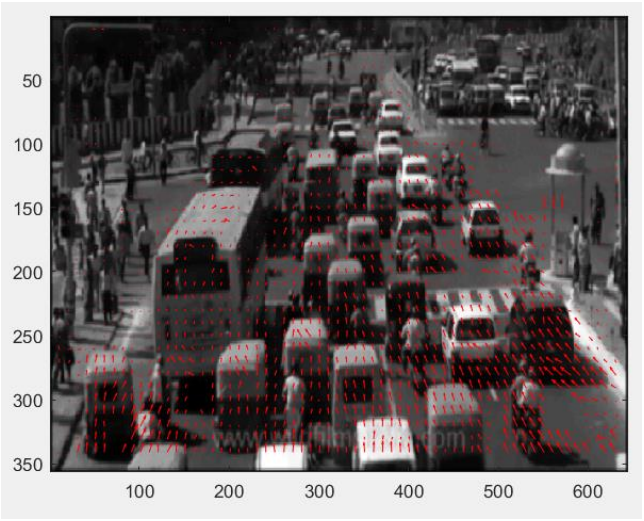
Cars2, Box Filter, K=10, K=20, K=30

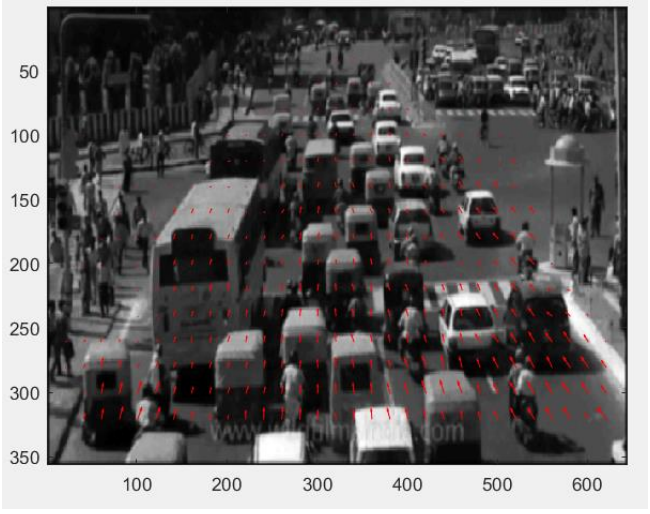


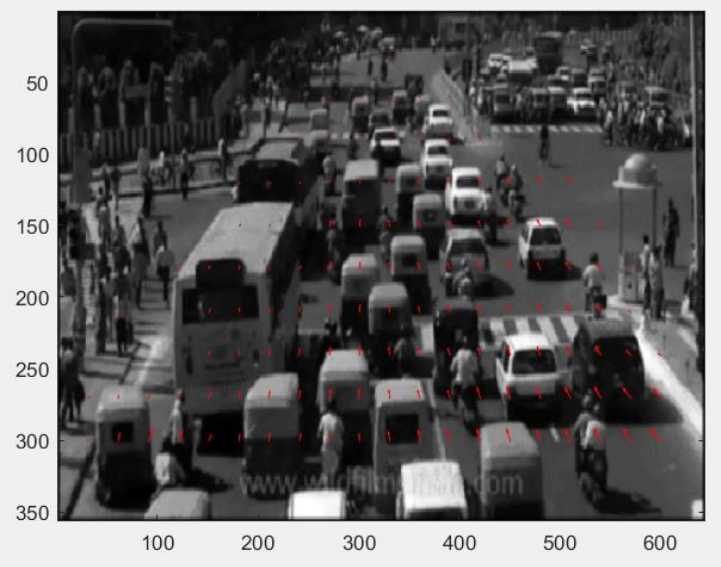




Cars2, Gaussian Filter, K=10, K=20, K=30







DISCUSSION

Affect of Window Size

If we compare the results, we obtain that increasing of K (window size) makes our detection less vulnerable to the little changes. To illustrate, look the Rubic cube images with K=10 and K=30. When it has a bigger window size, its velocity gradients are distributed more sparsely. As another example, taxi scene shows a clear demonstration of the given phenomenon. In K=10 case there are lots of spots that is detected as motion wrongly. On the other hand, in K=30 only real motion parts of the scene are detected. This is true for all of other sample images. To sum up, higher the window size, less the vulnerability for detection of motion.

Affect of Filter Type

We applied box filter and gaussian filter separately and we get the following result. When the image is noisier, due to high blurry characteristic of Gaussian filter, it gives more robust velocity values these type of images. Let us look at Rubic cube again, if we compare Box Filtered and Gaussian Filtered images with K=10, we can see the difference clearly. In the first image (box filter one), due to noise causing the intensity changes in the pixels it has many spots which are detected as a motion. On the other hand, the second image which is obtained by Gaussian filter does not detect these spots as motion. Because these spots are already eliminated that prevents the wrong detections.

CODES

lab6OF.m

```
function lab6OF(ImPrev,ImCurr,k,Threshold)
% Smooth the input images using a Box filter or Gaussian
Filter
boxfilter = ones(5,5);

gaussfilter = 1/273 *   [1 4 7 4 1;
                         4 16 26 16 4;
                         7 26 41 26 7;
                         4 16 26 16 4;
                         1 4 7 4 1];

ImCurr = conv2(ImCurr, boxfilter);
ImPrev = conv2(ImPrev, boxfilter);

% Calculate spatial gradients (Ix, Iy) using Prewitt filter
 prewitt_xkernel = [-1 0 1;
                    -1 0 1;
                    -1 0 1];

 prewitt_ykernel = [-1 -1 -1;
                     0 0 0;
                     1 1 1];

 IxG = conv2(ImCurr, prewitt_xkernel);
 IyG = conv2(ImCurr, prewitt_ykernel);

% Calculate temporal (It) gradient
 ItG = ImPrev - ImCurr;

[ydim,xdim] = size(ImCurr);
Vx = zeros(ydim,xdim);
Vy = zeros(ydim,xdim);
G = zeros(2,2);
b = zeros(2,1);
cx=k+1;
for x=k+1:k:xdim-k-1
cy=k+1;
for y=k+1:k:ydim-k-1
% Calculate the elements of G and b
 Ix = IxG(y-k:y+k,x-k:x+k);
 Iy = IyG(y-k:y+k,x-k:x+k);
 It = ItG(y-k:y+k,x-k:x+k);

 G11 = sum(sum(Ix.*Ix));
 G12 = sum(sum(Ix.*Iy));
 G21 = G12;
 G22 = sum(sum(Iy.*Iy));
```

```
 G = [G11 G12; G21 G22];
 b1 = sum(sum(Ix.*It));
 b2 = sum(sum(Iy.*It));
 b = [b1; b2];

if (min(eigs(G)) < Threshold)
Vx(cy,cx)=0;
Vy(cy,cx)=0;
else
% Calculate u
u = -inv(G)*b;
Vx(cy,cx)=u(1);
Vy(cy,cx)=u(2);
end
cy=cy+k;
end
cx=cx+k;
end
cla reset;
imagesc(ImPrev); hold on;
[xramp,yramp] = meshgrid(1:1:xdim,1:1:ydim);
quiver(xramp,yramp,Vx,Vy,10,'r');
colormap gray;
end
```

lab6OFMain.m

```
clear all; close all; clc;
load('sphere.mat');
Seq = sphere;
% Load the files given in SUcourse as Seq variable
[row,col,num]=size(Seq);
% Define k and Threshold
k = 10;
Threshold = 5000;
for j=2:1:num
ImPrev = Seq(:,:,j-1)
ImCurr = Seq(:,:,j)
lab6OF(ImPrev,ImCurr,k,Threshold);
pause(0.1);
end
```