Aynur Süne - 21041

EE417 Post-Lab #6

lab6OFMain.m

```
clear all; close all; clc;
% Load the files given in SUcourse as Seq variable
load('traffic.mat');
Seq = traffic;

[row,col,num]=size(Seq);

% Define k and Threshold
k=10;
Threshold = 1000;
for j=2:1:num
        ImPrev = Seq(:,:,j-1);
        ImCurr = Seq(:,:,j);
        lab6OF(ImPrev,ImCurr,k,Threshold);
        pause(0.1);
end
```

lab6OF.m

```
function lab6OF(ImPrev,ImCurr,k,Threshold)
% Smooth the input images using a Box filter
 %  ImPrev = lab1locbox(ImPrev,k);
 %  ImCurr = lab1locbox(ImCurr,k);
% Calculate spatial gradients (Ix, Iy) using Prewitt filter
Ix = zeros(size(ImCurr));
Iy = zeros(size(ImCurr));
[r,c,color] = size(ImCurr);

filterX = [-1 0 1;-1 0 1;-1 0 1];
Ix = conv2(ImCurr,filterX,'same');
Iy = conv2(ImCurr,filterX.','same');


% Calculate temporal (It) gradient
It = double(ImPrev) - double(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
       subimg_x = Ix(y-k:y+k,x-k:x+k);
       subimg_y = Iy(y-k:y+k,x-k:x+k);
       subimg_t = It(y-k:y+k,x-k:x+k);
```

```
        g1 = sum(sum(subimg_x.^2));
        g2 = sum(sum(subimg_y.*subimg_x));
        g3 = sum(sum(subimg_y.^2));

        b1 = sum(sum(subimg_x.*subimg_t));
        b2 = sum(sum(subimg_y.*subimg_t));
        G = [g1 g2;g2 g3];
        b = [b1;b2];

        eigen = eigs(G);
       if (min(eigen) < 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
```

Optical Flow is the distribution of apparent velocities of movement of brightness pattern in an image. Velocities of the objects can be estimated with the image gradients of an image with the following equation:

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

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

We consider each frame of '.mat' file one by one. We need to look at the difference of the gradient values between consecutive frames to find the brightness pattern, i.e. optical flow. This gives the movement of the objects. First, we upload necessary .mat file to the workspace. We define k value for windowing operation and threshold for minimum

eigenvalue. We take the current frame and the previous one, then call the function to calculate velocities. This process goes on for all frames of .mat file.

Inside lab6OF function, first we smooth the current and previous frames with box filter and apply Prewitt filter to these frames. Prewitt operation is done by convolution with Prewitt kernel. Gradient along x axis is calculated by its filter kernel, and gradient along y axis is calculated by the transpose of this kernel. We calculate temporal gradient by subtracting current image from previous image. Then we start to consider each frame with the previous frame to calculate the velocity of the objects with the formula given, that based on image gradient of the current frame and temporal gradient (difference between previous and current frame). This calculation is done for each window of the frame during windowing operation. Output is the arrows in the of the velocities, i.e. optical flow, of the objects in each frame, displayed one after another. These directions are found by the eigenvalues of the G matrix given in velocity formula. Minimum eigenvalue is thresholded and if it is above the threshold, it is considered as a movement in that direction.

First, since image gets too blurry after smoothing operation, box filter operation is not applied. With k=10 and threshold=1000, resulting output is as follows:

k=10 & Threshold = 10000

k=10 & Threshold = 100000

k=10 & Threshold = 1000000

k=10 & Threshold = 100

It is observed that when Threshold increases, less arrows are outputted indicating a movement. When Threshold equals to 1000000 which is a very large number, algorithm is able to detect real movement and eliminate noise. When threshold is low, noise is outputted as movement but we can observe that there is no movement in that area.

k=20 & Threshold = 100

k=20 & Threshold = 1000

k=20 & Threshold = 10000

k=20 & Threshold = 100000

k=20 & Threshold = 1000000

We can observe that when k increased to 20; as Threshold increases, still weak flows eliminated nicely and real movement is observed with the arrows. But this time intensity of the arrows is different. With the same threshold, arrows are found in same locations, but bigger k value resulted in less arrows in these locations.

k=30 & Threshold = 100

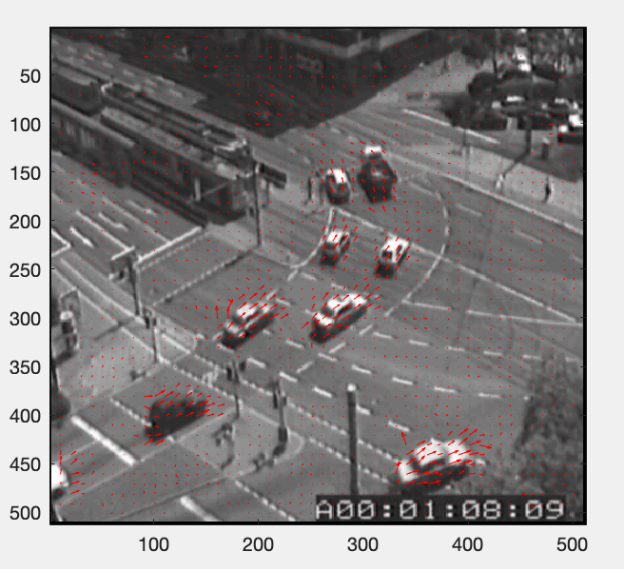
k=30 & Threshold = 10000

k=30 & Threshold = 1000000

With k=30, intensity of the arrows in the movement locations are even less.

Aynur Süne - 21041

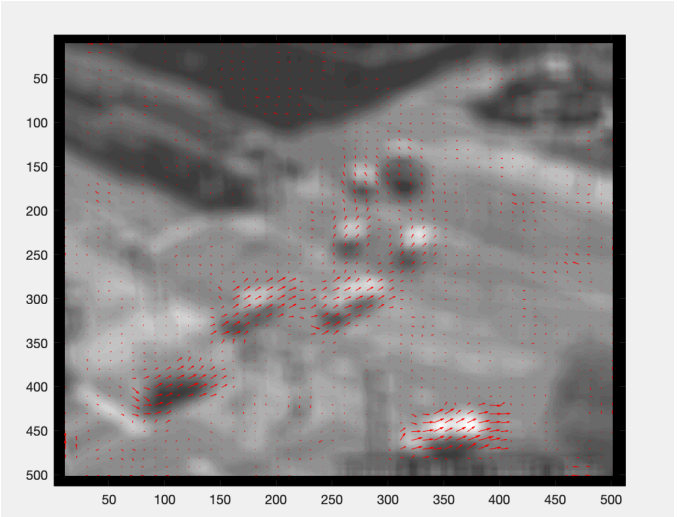
k=10 & Threshold = 100 & Gauss Filter



k=10 & Threshold = 100 Without Gauss Filter

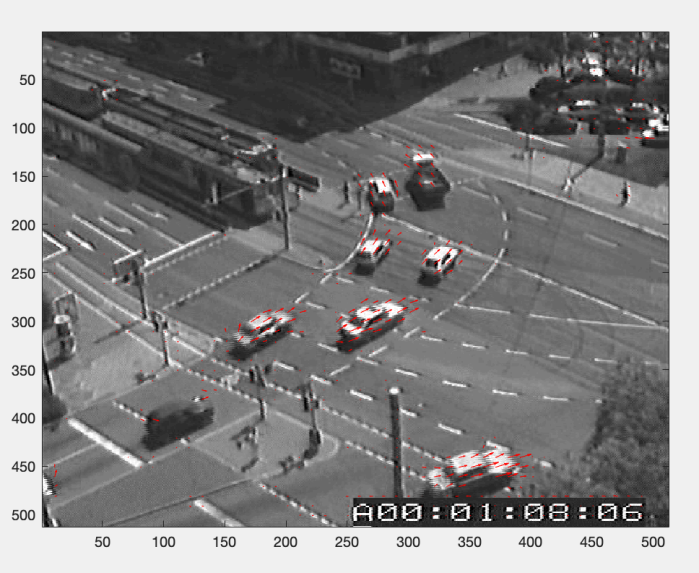
k=10 & Threshold = 100 & Box Filter(k=2)

k=10 & Threshold = 100 & Box Filter(k=10)

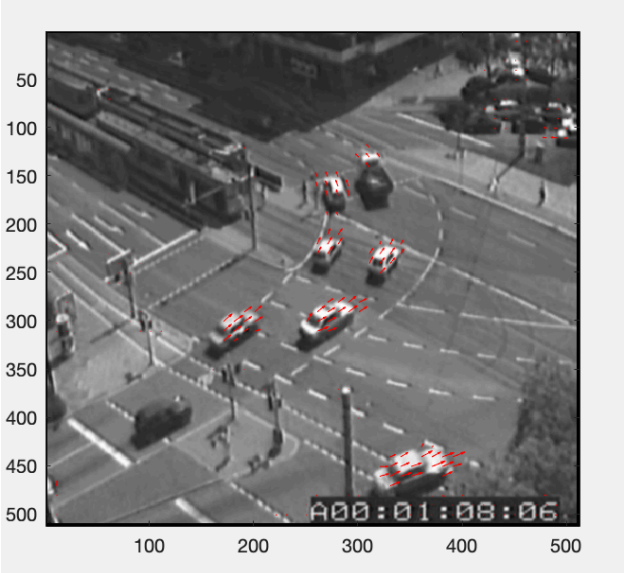


When we set k value of the Box filter to 2 as in default Gauss Filter, output is blurry than Gauss Filter but arrow numbers and locations are similar in all. No significant change is observed about the detection of the optical flow with different smoothing operations. From setting two k values the same, I understand that this not the way to do. From now on, I will always set k to 2 when operating with Box Filter.

k=10 & Threshold = 1000000 Without Gauss Filter



k=10 & Threshold = 1000000 & Gauss Filter



k=10 & Threshold = 1000000 & Box Filter

We can easily observe that in the areas with moving objects, there are less arrows found with Gauss Filter than unfiltered image. But Box filtered image has even lesser arrow than Gaussian filtered image. Still unfiltered image is the best with high threshold, but Gaussian is better than Box Filter with high threshold. Most of the noise is eliminated with high threshold anyways. But with threshold equals to 10000, no significant change is observed between the two smoothing operations.

Aynur Süne - 21041

k=30 & Threshold = 100 Without Filter

k=30 & Threshold = 100 & Gauss Filter

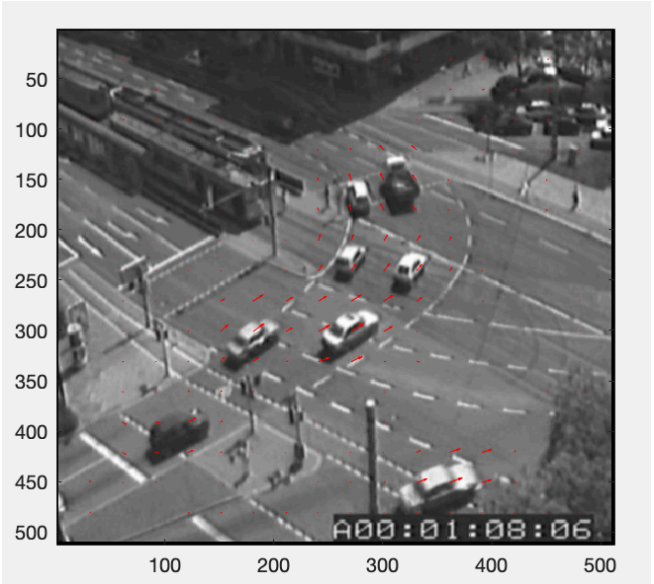
k=30 & Threshold = 100 & Box Filter

There is no significant difference between unfiltered and filtered images with low threshold when k value is high.

Aynur Süne - 21041

k=30 & Threshold = 1000000 Without Filter

k=30 & Threshold = 1000000 & Gauss Filter



k=30 & Threshold = 1000000 & Box Filter

There is no significant difference between unfiltered and filtered images with high threshold when k value is also high.