INTRODUCTION

Optical flow throughout a video is calculated during the lab.

Optical flow is corresponding to use of apparent motion of brightness patterns observed in an image sequence (video). One important assumption in here is that the apparent brightness of moving objects remains constant as we discussed in the lecture.

Both in-lab and post-lab results will be discussed at the end of the report in discussion part.

ALGORITHM STEPS FOR BOTH IN-LAB & POST-LAB

1. Applying smoothing to images
2. Calculating spatial gradients (I_x and I_y) using Prewitt Filter
3. Calculating temporal gradient (I_t)
4. For each pixel in the centre of window, creating G and b matrices
5. If minimum eigenvalue of G is larger than threshold
    - Calculating u (velocity vectors matrix)

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

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

6. At the end, show these velocity vectors on the image

Ege Erdogan 25331

IN-LAB & POST-LAB RESULTS

Traffic Video Sequence with Different Window Size (k) Values

K = 10, Threshold = 100000, Smoothing = Box Filter:

K = 20, Threshold = 100000, Smoothing = Box Filter:

K = 30, Threshold = 100000, Smoothing = Box Filter:

Ege Erdogan 25331

Taxi Video Sequence with Different Window Size (k) Values

K = 10, Threshold = 100000, Smoothing = Box Filter:

K = 20, Threshold = 100000, Smoothing = Box Filter:

K = 30, Threshold = 100000, Smoothing = Box Filter:

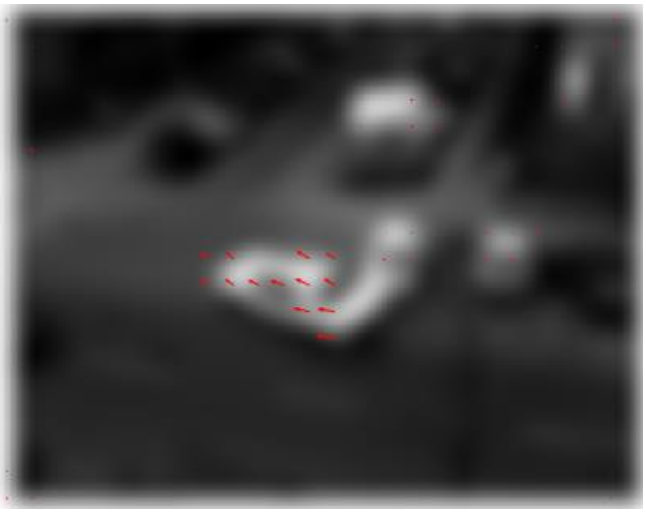
Ege Erdogan 25331

Taxi Video Sequence with Different Filters and Kernel Sizes

K = 20, Threshold = 100000, Smoothing = Box Filter:

K = 20, Threshold = 100000, Smoothing = Gaussian Filter with 3x3:

K = 20, Threshold = 100000, Smoothing = Gaussian Filter with 9x9:

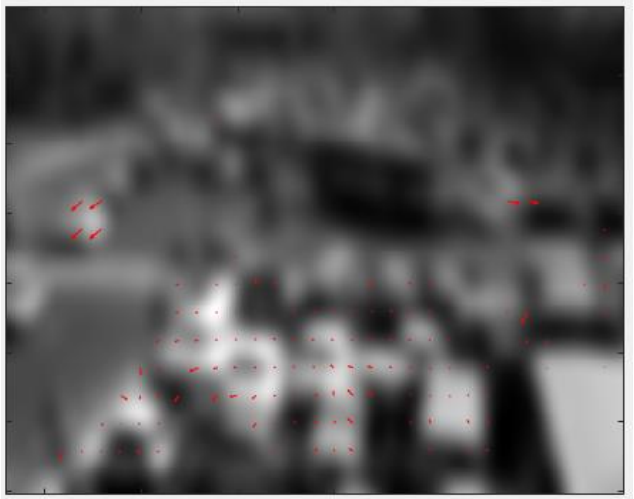


Cars1 Video Sequence with Different Filters and Sizes

K = 20, Threshold = 100000, Smoothing = Box Filter:

K = 20, Threshold = 100000, Smoothing = Gaussian Filter with 3x3:

K = 20, Threshold = 100000, Smoothing = Gaussian Filter with 9x9:



Ege Erdogan 25331

Traffic Video Sequence with Different Threshold Values

K = 30, Threshold = 1000, Smoothing = Box Filter:

K = 30, Threshold = 100000, Smoothing = Box Filter:

K = 30, Threshold = 500000, Smoothing = Box Filter:

Ege Erdogan 25331

Rubic Video Sequence with Different Threshold Values

K = 30, Threshold = 1000, Smoothing = Box Filter:

K = 30, Threshold = 100000, Smoothing = Box Filter:

K = 30, Threshold = 500000, Smoothing = Box Filter:

DISCUSSION

We can observe that it is possible to detect optical flows in the video sequences with these algorithmic steps. Different parameters have various effects to detection of optical flow.

Firstly, we can observe that window size is affecting the number of velocity vectors in the output images. When we increase the window size, the number of velocity vectors is decreasing. For instance, if we look for traffic video sequence, k = 30 is a better choice than k = 10 as we have nicer and smoother number of velocity vectors in the output image. Therefore, window size should be chosen carefully.

Secondly, we can observe that filter type and its kernel size is affecting the output optical flow. When we use Gaussian Filter instead of Box Filter, we are ending up with blurrier output image in the Gaussian Filtered output image. This reduces the output quality. For instance, we can see the difference between 3x3 Box Filtered and 3x3 Gaussian Filtered taxi video sequence. I think Box filter is more useful for detecting optical flow as it does not add too much blurring to the output. However, we should also observe that 3x3 Gaussian Filtered output's optical flow vectors is more accurate than Box Filtered version. Thus, there is a trade-off between output quality and optical flow accuracy.

Lastly, we can observe that threshold value is affecting the output optical flow. However, this effect is not a dramatic change. One effect of threshold can be that if it is set too low, the number of non-accurate motion vectors may increase. On the other hand, if it is set too high, we may miss some motion vectors. Therefore, it should be chosen carefully but it is not too important as the window size or filter type.

CODES

Lab7OFMain:

```
% Load the files given in SUcourse as Seq variable
load("traffic.mat");
load("cars1.mat");
load("cars2.mat");
load("rubic.mat");
load("sphere.mat");
load("taxi.mat");

Seq = cars2;
[row,col,num]=size(Seq);
% Define k and Threshold
% k = 10;
% k = 20;
k = 30;
% Threshold = 1000;
Threshold = 100000;
% Threshold = 500000;
for j=2:1:num
ImPrev = Seq(:,:,j-1);
ImCurr = Seq(:,:,j);
lab7OF(ImPrev,ImCurr,k,Threshold);
pause(0.1);
end
```

Lab7OF:

```
function lab7OF(ImPrev,ImCurr,k,Threshold)
% Smooth the input images using a Box filter
box_filter = [1,1,1;1,1,1;1,1,1];
ImPrev = conv2(ImPrev, box_filter, 'same');
ImCurr = conv2(ImCurr, box_filter, 'same');

% % Smooth the input image using a Gaussian Filter 3x3
% ImPrev = imgaussfilt(double(ImPrev), [3, 3]);
% ImCurr = imgaussfilt(double(ImCurr), [3, 3]);

% % Smooth the input image using a Gaussian Filter 9x9
% ImPrev = imgaussfilt(double(ImPrev), [9, 9]);
% ImCurr = imgaussfilt(double(ImCurr), [9, 9]);

% Calculate spatial gradients (Ix, Iy) using Prewitt filter
x_filter = [-1,0,1; -1,0,1; -1,0,1];
y_filter = [-1,-1,-1; 0,0,0; 1,1,1];
Ix = conv2(ImCurr, x_filter, 'same');
Iy = conv2(ImCurr, y_filter, 'same');
% Calculate temporal (It) gradient
It = 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
    my_window_x = Ix((y-k):(y+k), (x-k):(x+k));
    my_window_y = Iy((y-k):(y+k), (x-k):(x+k));
    my_window_t = It((y-k):(y+k), (x-k):(x+k));
    G(1,1) = sum(sum(my_window_x.*my_window_x));
    G(1,2) = sum(sum(my_window_x.*my_window_y));
    G(2,1) = G(1,2);
    G(2,2) = sum(sum(my_window_y.*my_window_y));
    b(1,1) = sum(sum(my_window_x.*my_window_t));
    b(2,1) = sum(sum(my_window_y.*my_window_t));
% Calculate the elements of G and b
if (min(eig(G)) < Threshold)
Vx(cy,cx)=0;
Vy(cy,cx)=0;
else
% Calculate u
u = (inv(G)*b)*-1;
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
```