NYU CS-GY 6643, Computer Vision
Assignment 4 (Practical Problems)

Name: Amitesh Kumar Sah
NYU ID: N19714360

Topic: OPTICAL FLOW

Objective:

Our goal is to implement an optical flow method that can measure the motion of objects in a series of image data.

Procedure:

1. Choose two consecutive images from the video sequence.
2. Apply smoothing to the images (remember that optical flow assumes smooth object boundaries, i.e. boundaries with larger smoothness than the spatial shift).
3. Calculate the temporal gradient image ∂E ∂t via the difference of the blurred versions of the two consecutive frames, i.e. simply by subtracting the two frames as I(x,t+1)-I(x,t).
4. Estimate the spatial derivatives Ex = ∂E ∂x and Ey = ∂E ∂y by calculating pixel differences I(x+1,y,t)-I(x,y,t) for Ex and I(x,y+1,t)-I(x,y,t) for Ey.
5. Display the original image and the spatial and time gradients (see Fig. 2 for an example).
6. Calculate the normal flow at each pixel.
7. Display the resulting flow vectors as a 2D image. Overlay these vectors and the gray level image.
8. Eventually apply this program to another consecutive image pair of the 6 image video sequence and the result was compared.

FORMULA USED:

Flow calculated over pixel neighborhood:

By choosing 2*2 pixel neighbourhood

$$\nabla E.v + \frac{\partial E}{\partial t} = 0$$

We have 4 measurements from 2x2 pixels to solve for v:

$$Av + b = 0$$

$$v = -(A^T A)^{-1} A^T b$$

The columns of A are the x and y components of the gradient ∇E and b is a column vector of the t gradient component of E, Et

Regularization: Horn and Schunck

$$u_{ij}^{n+1} = \bar{u}_{ij}^{n} - \alpha I_x \qquad \alpha = \frac{I_x \bar{u}_{ij}^{n} + I_y \bar{v}_{ij}^{n} + I_t}{1 + \lambda(I_x^2 + I_y^2)}$$
$$v_{ij}^{n+1} = \bar{v}_{ij}^{n} - \alpha I_y$$

$\bar{u}, \bar{v}$ are the averages of values of neighbors

Summarize the algorithm:

```
% Set initial value of u and v to zero
u = 0;
v = 0;

% Weighted Average kernel
kernel=[1/12 1/6 1/12;1/6 0 1/6;1/12
 1/6 1/12];

%Minimizing Iterations (100 times)
for i=1:100

%Compute local averages of the vectors

 uAvg=conv2(u,kernel,'same');
 vAvg=conv2(v,kernel,'same');

%Compute flow vectors constrained by the local
averages and the optical flow constraints,
where alpha is the smoothing weight

 u = uAvg - ( Ix .* ( ( Ix .* uAvg ) +
( Iy .* vAvg ) + It ) ) ./ ( alpha^2 +
Ix.^2 + Iy.^2);

 v = vAvg - ( Iy .* ( ( Ix .* uAvg ) +
( Iy .* vAvg ) + It ) ) ./ ( alpha^2 +
Ix.^2 + Iy.^2);

end
```

Results and Discussion

For 2 consecutive Images 1 and 2 on raw image

In the figure below, spatio-temporal gradient is shown. For time gradient, we took two successive images and took the difference between them. It can be seen that it show the temporal difference between the images and we can see the edges slightly displaced.
For Horizontal gradient, we took image1 and convolved the image with a high pass filter [-1,1] which detects the vertical discontinuities edges. Its basically taking the difference of successive pixel in horizontal direction. We see more vertical edges.

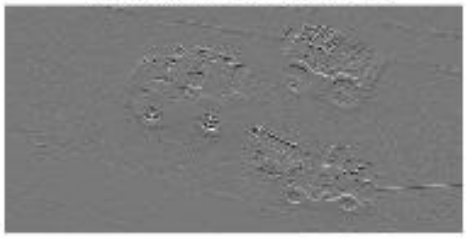
For Vertical gradient, we took image1 and convolved the image with a high pass filter [-1,1] ^Twhich detects the horizontal edges. Its basically taking the difference of successive pixel in vertical direction. We see more horizontal edges.

Original Image

Time gradient

Horizontal spatial gradient

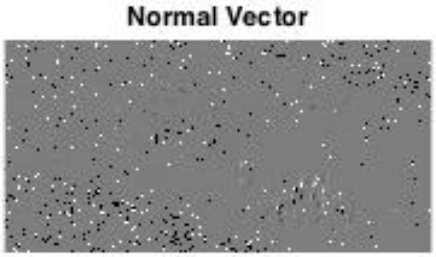
Vertical spatial gradient

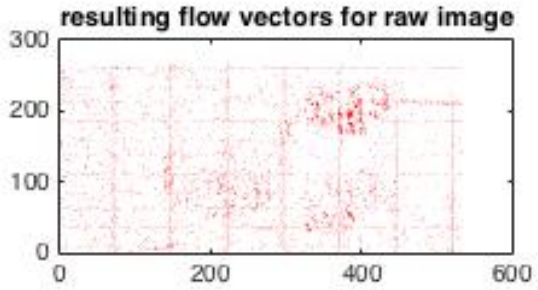


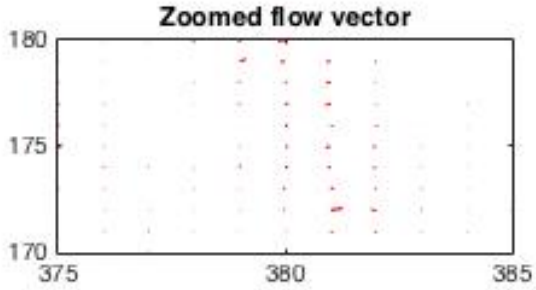
For a normal vector, with one point we can only detect movement perpendicular to the brightness gradient. This motion is normal to the local intensity contour line which adds a second constraint to the motion vector.
Solution is to take a patch of pixels around the pixel of interest. From Normal Vector, we can also conclude the shape and rough estimate position of the image.

The flow vector here was calculated n the raw images without smoothing it. We took 2*2 neighbourhood of the interested pixel we want to calcualate the optical flow direction and got 4 equation for two unknown varibles i.e. u and v . U and V was solved with the formula used above as it was over determined system. Here, we calculated for each and every pixel. We couldn't properly see at the vector from the image, so we scaled the optical flow by 2 times. I also zoomed the optical flow over a small patch to have a better look at the direction of optical flow. It is noticed that the direction is inconsistent and its random(not following a unique direction). It is due to the rough image without the image smoothed and the simple method used to calculate the optical flow as this method does not give the optimal solution.

Later, Optical flow is overlay over the raw grayscale image and we can see for the left car, many vector fields are oriented towards left and for the two right cars, the flow vectors are oriented towards right.







<u>For 2 consecutive Images 1 and 2 after smoothing with Gaussian filter sigma =1</u>

Time gradient : we can see the displacement with respect to time between the two images. It is noticed that most of the pixel is dark , hence giving us only the displacement of the object. The displacement is more clear and distinct and we can easily identify the object that moved. This was possible after smoothing the image.

Horizontal Gradient: Vertical edges are more distinct , We also notice the vertical edges of tiles more clearly in the background

Vertical Gradient: Horizontal edges are more distinct , We also notice the horizontal edges of tiles more clearly in the background

Original Image

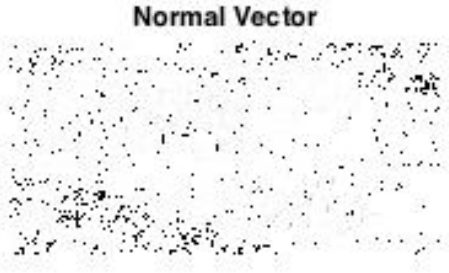
Time gradient

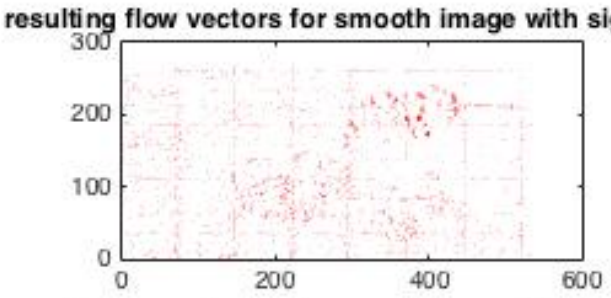
Horizontal spatial gradient

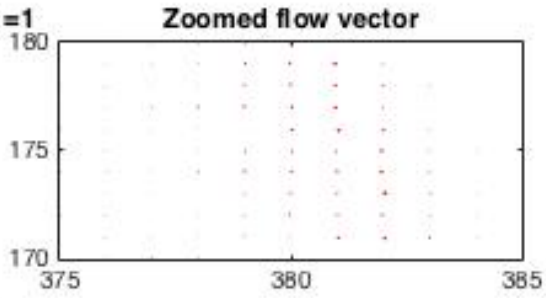
Vertical spatial gradient

Normal Vector: Its sparse and mostly consider the object that has actually undergone motion and does not take into account the background tile motion in detail.

Flow vector: It is well defined for the boundary . For the same texture, it is more directional now rather than random flow vector.
It can be seen that flow vector is more oriented. And after overlaying the flow vector ,we can see the direction of car is more uniform in respective direction.





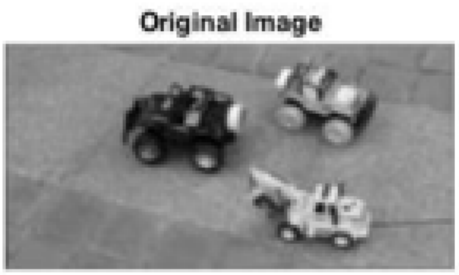


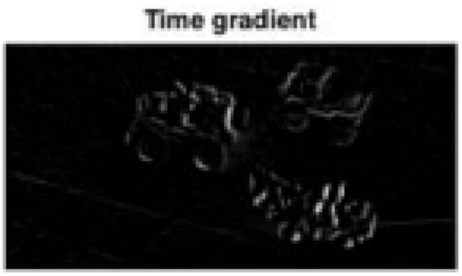
For 2 consecutive Images 1 and 2 after smoothing with Gaussian filter sigma =2

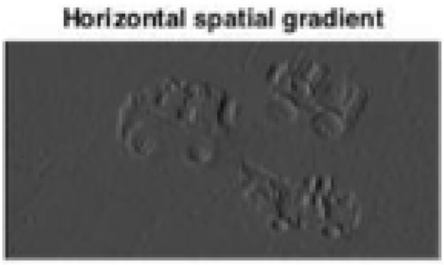
Time gradient : we can see the displacement with respect to time between the two images. It is noticed that most of the pixel is dark , hence giving us only the displacement of the object. The displacement is more clear and distinct and we can easily identify the object that moved. It gives better result compared to sigma =1.

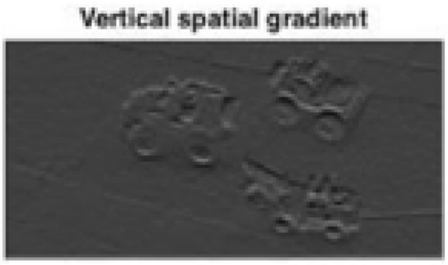
Horizontal Gradient: Vertical edges are more distinct , We also notice the vertical edges of tiles more clearly than sigma=1 in the background

Vertical Gradient: Horizontal edges are more distinct , We also notice the horizontal edges of tiles more clearly than sigma=1 in the background



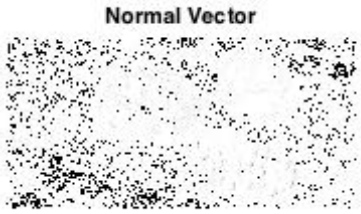


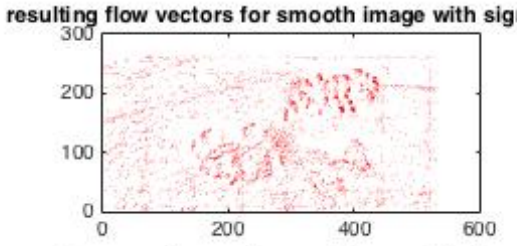


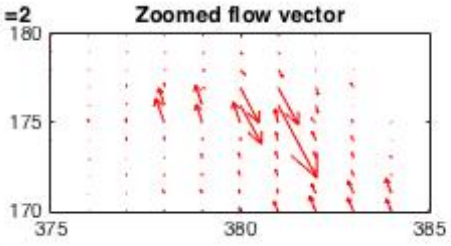


Normal Vector: Its sparse and mostly consider the object that has actually undergone motion .

Flow vector: It is well defined for the boundary . For the same texture, it is more directional now rather than random flow vector.
It can be seen that flow vector is more oriented. And after overlaying the flow vector ,we can see the direction of car is more uniform in respective direction.







<u>For all the 5 consecutive Images 1 and 2 , then 2 and 3, 3 and 4 , 4 and 5 after smoothing with Gaussian filter sigma =2</u>

Below are the 5 consecutive images taken into consideration

Original Image 1

Original Image 2

Original Image 3

Original Image 4

Original Image 5

Normal vector is calculated between consecutive images and hence it shows the direction of motion with repect to previous frame. It can be see that due to the motion of the right car toward right and two left car towards left, the overall boundary expanded, i.e. normal vector is slightly bigger.

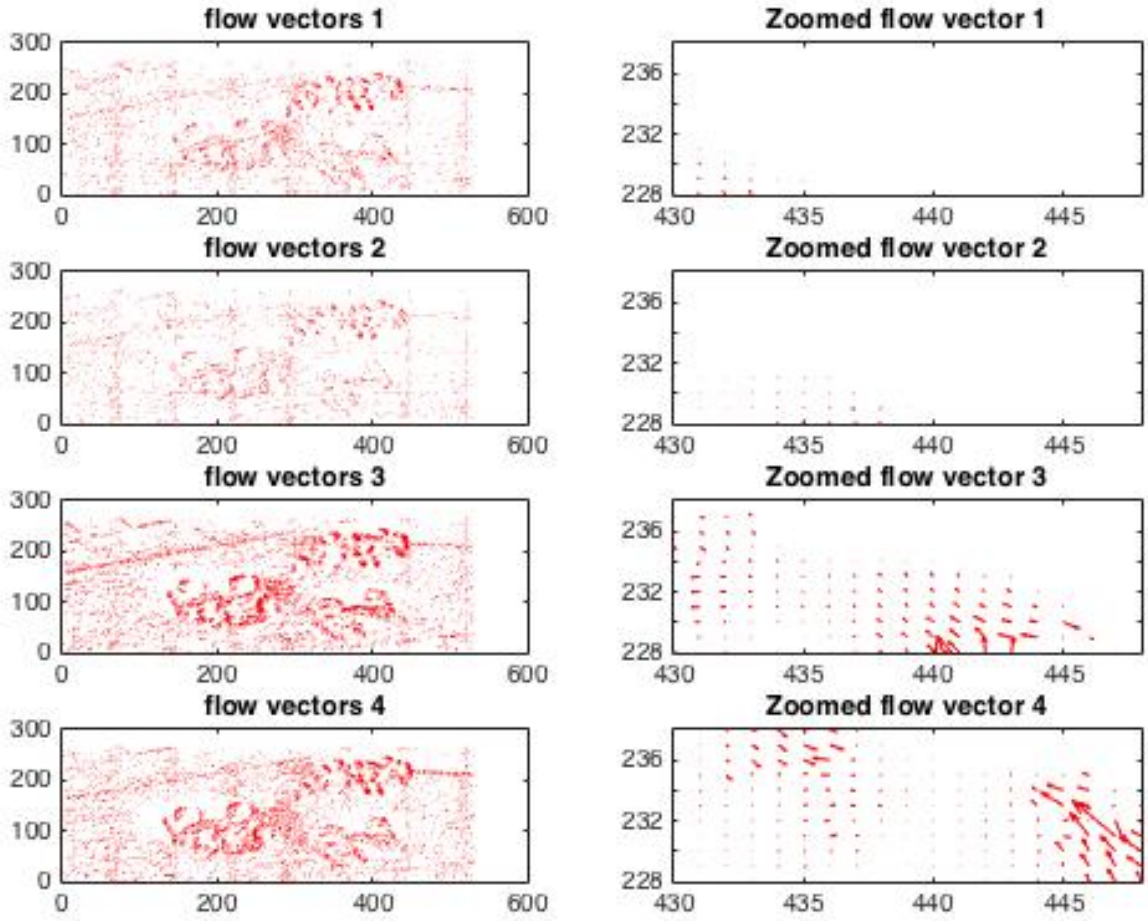
Normal Vector 1and2

Normal Vector 2and3

Normal Vector 3and4

Normal Vector 4and5

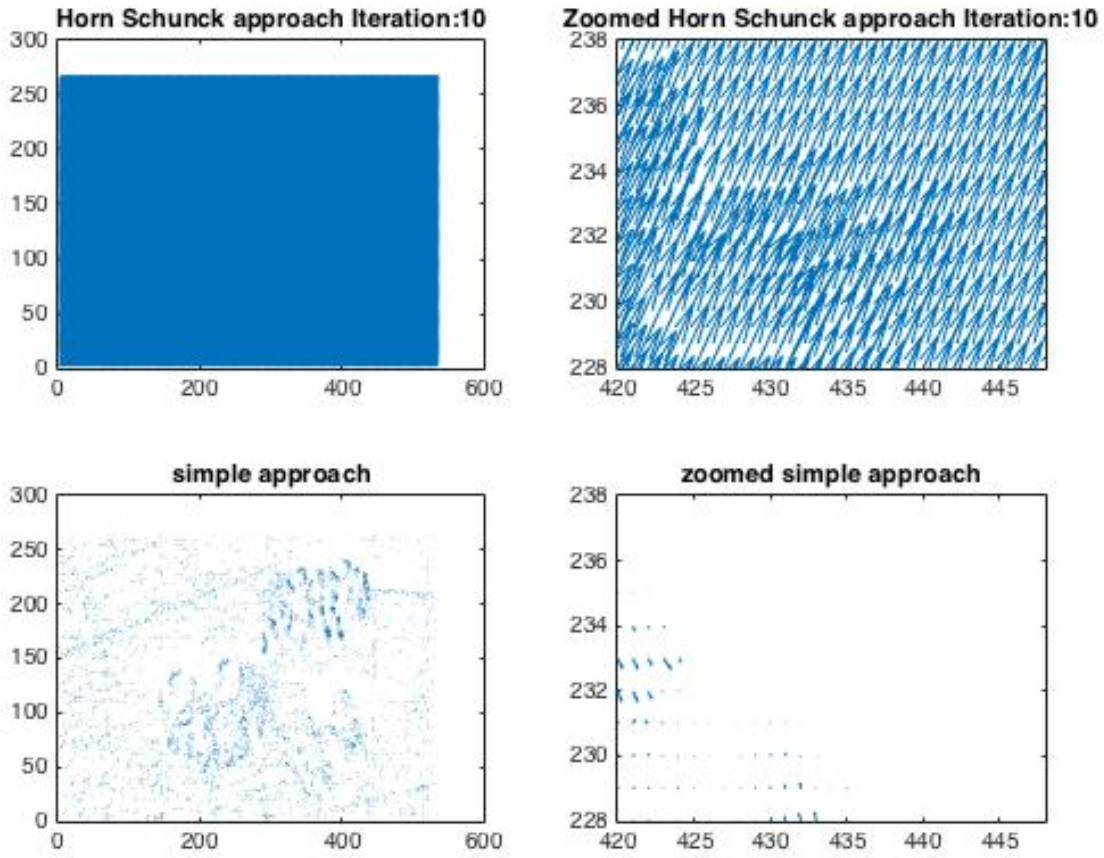
Due to the movement of the car, the flow vector from one to four, e flow vector is getting denser where there is car motion. From zoomed flow vetor of top right car, In zoomed flow vector1, there is rarely any optical flow, whereas as we go from 1 to 4, the number of optical flow vector increases, hence proving that car is moving towards right and slightly in upward direction



As we overlay the vector field on image 1, the optical flow field becomes denser as there is relatively more motion in image 5 from image 1 , than in image from 2 to image 1. Hence, showing us that our result is correct.

Applying Horn Schunck Method and Comparing this method with Simple approach

Horn Schunck Method gives smooth flow, global information, accurate time derivatives, It is also possible to use more then two frames. It is unsharp at boundaries. However due to its iterative method , it is slow . But the simple approach was fast and not so accurate.



.

My Own Video

In this setup, there is a small ball rotating from left to right with checkerboard as the background. I Below is the result

Original Image 1

Original Image 2

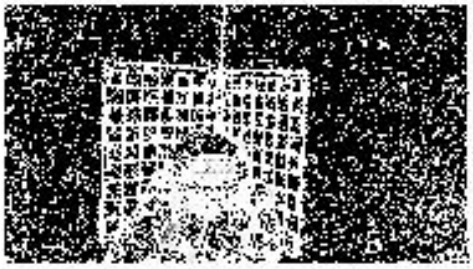
Original Image 3

Original Image 4

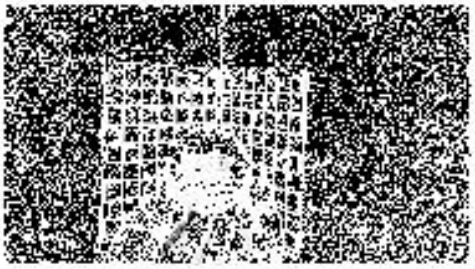
Original Image 5

Below is the Normal flow direction and the result is not accurate .

Normal Vector 1and2

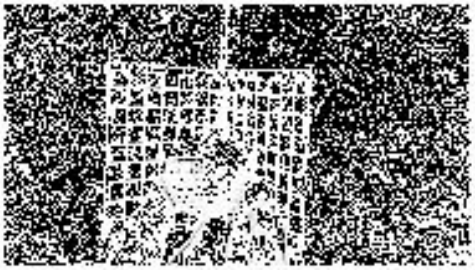


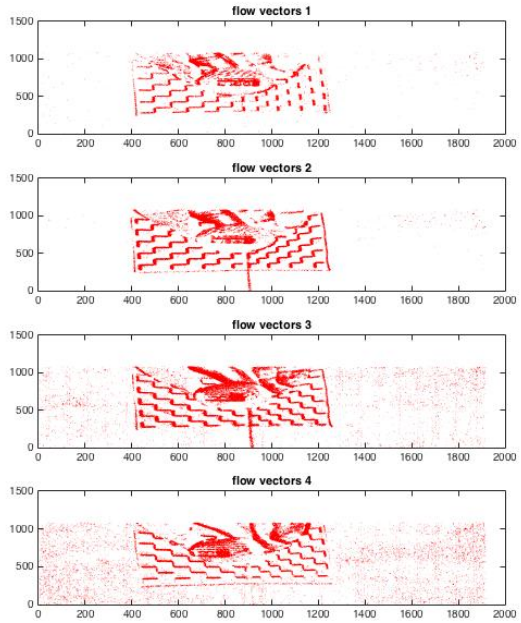
Normal Vector 2and3

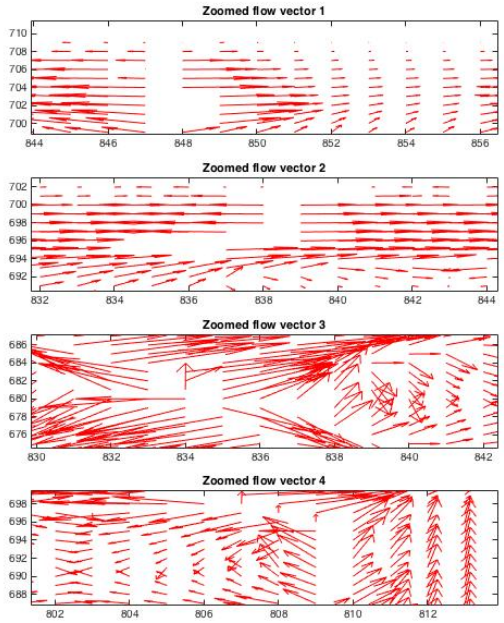


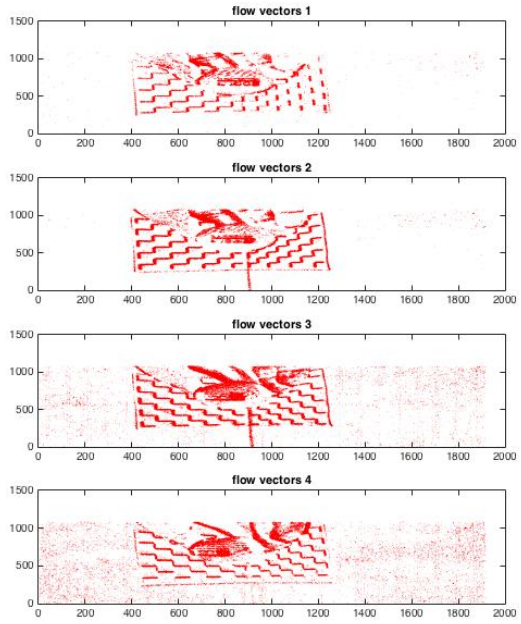
Normal Vector 3and4

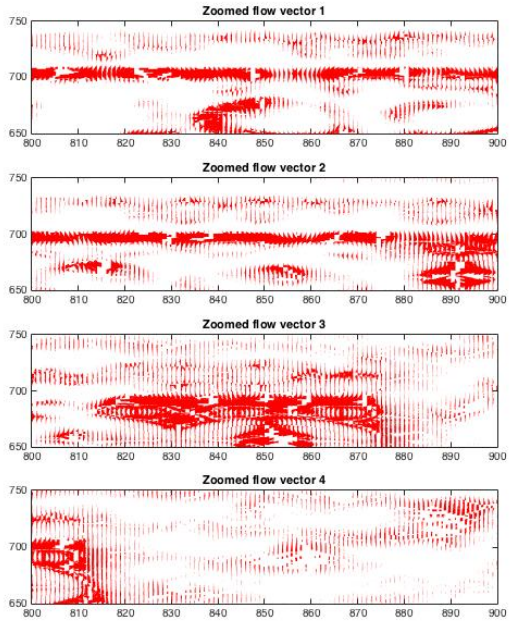
Normal Vector 4and5











The optical flow direction is towards both end. It may be due to the hand shaking initially. However , later optical flow motion shows that object is moving from left to right. At the end there optical flow direction bends upward showing the rotational direction and concluding the shape that it is a sphere.

As the optical flow is overlayed on image, we can see the displacement that image has undergone from its initial image is from left to right .Later the optical flow vector is apparent in the background showing the shaking of the camera when the video was shot.

Conclusion

Here we used optical flow to determine the displacement of moving objects from the sequence of images (discrete video frame with time). In this assignment I used to method to calculate optical flow. One was by using 2*2 pixel neighbourhood and evaluating the overdetermined linear system equation. Another method was by using Horn Schunck Method which uses a iterative technique to minimise the energy of the function and also uses a smoothing constraint i.e. l1 norm regularisation. We performed the experiment for different smoothing of images. For the raw image, optical flow was random and when gaussian filter was used , we obtained more accurate and oriented flow vector.

Then flow calculation of 2*2 pixel neighbourhood was used for series of images from 1 to 5 and estimated the flow vector. Through flow vector , we saw the displacement of the object from the initial position.
I compared the result with Horn Schunck Method and it can be concluded that Horn Schunck Method gives more accurate and well oriented flow vector than the Pixel neighbourhood method.

I applied the pixel neighbourhood to my own video sequence and I am pleased with the optical flow result I got as it gave me close to approximate direction which was from left to right.

Appendix A

```
clc;clear all;
close all;

% Choose two consecutive images from the video sequence
I1=imread('toy-car-images-bw/toy_formatted2.png');
I2=imread('toy-car-images-bw/toy_formatted3.png');

I1=double(I1);
I2=double(I2);

% Apply this flow calculation to a) the raw images,
[uv_raw,N_raw]=optical_flow(I1,I2);
figure,
subplot(3,2,1),imshow(I1,[]);title('Original Image 1');
subplot(3,2,2),imshow(N_raw,[]),title('Normal Vector');
subplot(3,2,3),quiver(uv_raw(:,:,1),uv_raw(:,:,2),3,'-r');
                title('resulting flow vectors for raw image');
subplot(3,2,4),quiver(uv_raw(:,:,1),uv_raw(:,:,2),3,'-
r');title('Zoomed flow vector');
ylim([170 180]);
xlim([375 385]);
subplot(3,2,5),imshow(I1,[]);
hold on
quiver(uv_raw(:,:,1),uv_raw(:,:,2),3,'-r');
title('vectors over the grayimage  Raw');
hold off


%%
% b) images filtered by a Gaussian smoothing of sigma=1,
gauss_filtered_image1 = gaussian_filter(I1, 1.0);
gauss_filtered_image2 = gaussian_filter(I2, 1.0);

[uv_sig1,N_sig1]=optical_flow(gauss_filtered_image1,gauss_filtered
_image2);
```

```
figure,
subplot(3,2,1),imshow(I1,[]);title('Original Image 1');
subplot(3,2,2),imshow(N_sig1,[]),title('Normal Vector');
subplot(3,2,3),quiver(uv_sig1(:,:,1),uv_sig1(:,:,2),3,'-r');
                 title('resulting flow vectors for smooth image
with sigma =1');
subplot(3,2,4),quiver(uv_sig1(:,:,1),uv_sig1(:,:,2),3,'-
r');title('Zoomed flow vector');
ylim([170 180]);
xlim([375 385]);
subplot(3,2,5),imshow(I1,[]);
hold on
quiver(uv_sig1(:,:,1),uv_sig1(:,:,2),3,'-r');
title('vectors over the grayimage with smooth, sigma=2');
hold off


%%
% c) images filtered by a Gaussian smoothing of sigma=2.
gauss_filtered_image11 = gaussian_filter(I1, 2.0);
gauss_filtered_image12 = gaussian_filter(I2, 2.0);

[uv_sig2,N_sig2]=optical_flow(gauss_filtered_image11,gauss_filtere
d_image12);

figure,
subplot(3,2,1),imshow(I1,[]);title('Original Image 1');
subplot(3,2,2),imshow(N_sig2,[]),title('Normal Vector');
subplot(3,2,3),quiver(uv_sig2(:,:,1),uv_sig2(:,:,2),3,'-r');
                 title('resulting flow vectors for smooth image
with sigma =2');
subplot(3,2,4),quiver(uv_sig2(:,:,1),uv_sig2(:,:,2),3,'-
r');title('Zoomed flow vector');
ylim([170 180]);
xlim([375 385]);
subplot(3,2,5),imshow(I1,[]);
hold on
quiver(uv_sig2(:,:,1),uv_sig2(:,:,2),3,'-r');
title('vectors over the grayimage with smooth, sigma=2');
hold off

%% For all the 5 images
I1=double(imread('toy-car-images-bw/toy_formatted2.png'));
I2=double(imread('toy-car-images-bw/toy_formatted3.png'));
I3=double(imread('toy-car-images-bw/toy_formatted4.png'));
I4=double(imread('toy-car-images-bw/toy_formatted5.png'));
I5=double(imread('toy-car-images-bw/toy_formatted6.png'));

% Show all the 5 image
```

```
figure,
subplot(3,2,1),imshow(I1,[]);title('Original Image 1');
subplot(3,2,2),imshow(I2,[]);title('Original Image 2');
subplot(3,2,3),imshow(I3,[]);title('Original Image 3');
subplot(3,2,4),imshow(I4,[]);title('Original Image 4');
subplot(3,2,5),imshow(I5,[]);title('Original Image 5');

% images filtered by a Gaussian smoothing of sigma=1,
gauss_filtered_image1 = gaussian_filter(I1, 2.0);
gauss_filtered_image2 = gaussian_filter(I2, 2.0);
gauss_filtered_image3 = gaussian_filter(I3, 2.0);
gauss_filtered_image4 = gaussian_filter(I4, 2.0);
gauss_filtered_image5 = gaussian_filter(I5, 2.0);

% Apply this flow calculation
[uv1,N1]=optical_flow(gauss_filtered_image1,gauss_filtered_image2)
;
[uv2,N2]=optical_flow(gauss_filtered_image1,gauss_filtered_image3)
;
[uv3,N3]=optical_flow(gauss_filtered_image1,gauss_filtered_image4)
;
[uv4,N4]=optical_flow(gauss_filtered_image1,gauss_filtered_image5)
;

% Plot Normal
figure,
subplot(2,2,1),imshow(N1,[]),title('Normal Vector 1and2');
subplot(2,2,2),imshow(N2,[]),title('Normal Vector 1and3');
subplot(2,2,3),imshow(N3,[]),title('Normal Vector 1and4');
subplot(2,2,4),imshow(N4,[]),title('Normal Vector 1and5');

%Plot optical flow
figure,
subplot(4,2,1),quiver(uv1(:,:,1),uv1(:,:,2),3,'-r');title('flow
vectors 1');
subplot(4,2,2),quiver(uv1(:,:,1),uv1(:,:,2),3,'-r');title('Zoomed
flow vector 1');
ylim([170 180]);xlim([375 385]);
subplot(4,2,3),quiver(uv2(:,:,1),uv2(:,:,2),3,'-r');title('flow
vectors 2');
subplot(4,2,4),quiver(uv2(:,:,1),uv2(:,:,2),3,'-r');title('Zoomed
flow vector 2');
ylim([170 180]);xlim([375 385]);
subplot(4,2,5),quiver(uv3(:,:,1),uv3(:,:,2),3,'-r');title('flow
vectors 3');
subplot(4,2,6),quiver(uv3(:,:,1),uv3(:,:,2),3,'-r');title('Zoomed
flow vector 3');
ylim([170 180]);xlim([375 385]);
subplot(4,2,7),quiver(uv4(:,:,1),uv4(:,:,2),3,'-r');title('flow
```

```
vectors 4');
subplot(4,2,8),quiver(uv4(:,:,1),uv4(:,:,2),3,'-r');title('Zoomed
flow vector 4');
ylim([170 180]);xlim([375 385]);

% Plot overlay vector
figure,
subplot(2,2,1),imshow(I1,[]);hold on
quiver(uv1(:,:,1),uv1(:,:,2),3,'-r');
title('vectors over the grayimage 1');
hold off

subplot(2,2,2),imshow(I1,[]);hold on
quiver(uv2(:,:,1),uv2(:,:,2),3,'-r');
title('vectors over the grayimage 2');
hold off

subplot(2,2,3),imshow(I1,[]);hold on
quiver(uv3(:,:,1),uv3(:,:,2),3,'-r');
title('vectors over the grayimage 3');
hold off

subplot(2,2,4),imshow(I1,[]);hold on
quiver(uv4(:,:,1),uv4(:,:,2),3,'-r');
title('vectors over the grayimage 4');
hold off

%% Horn Schunck Method
figure,
j=1;
for i=1:5:40
[u,v]=horn_schunck(gauss_filtered_image1,gauss_filtered_image2,i);
subplot(4,2,j),quiver(u,v),title(['Iteration:',num2str(i)]);
 xlim([400 410]);
 ylim([58 64]);
j=j+1;
end

%%
% Comparing the flow fields with the ones obtained with the
simplified approach
[u,v]=horn_schunck(gauss_filtered_image1,gauss_filtered_image2,10)
;
[uv_simple,N_simple]=optical_flow(gauss_filtered_image1,gauss_filt
ered_image2);
figure,
subplot(1,2,1),quiver(u,v),title('Horn Schunck approach
Iteration:10');
```

```
subplot(1,2,2),quiver(uv_simple(:,:,1),uv_simple(:,:,2),3);title('
simple approach');
```

Appendix B

Function for the optical flow calculation

```
function [uv,N]=optical_flow(Image1,Image2)
% Calculate the temporal gradient image ?E/?t via the difference
of the blurred versions of the
% two consecutive frames.
Et=double(Image2-Image1);

% Calculate the spatial derivatives Ex =?E/?x and Ey =?E/?y ,
simply by
% subtracting the two frames as I(x,t+1)-I(x,t).
    Ex=conv2(Image1,0.5*[-1,1],'same');
    Ey=conv2(Image1,0.5*[-1;1],'same');

% Display the original image and the spatial and time gradients
 figure,
subplot(2,2,1),imshow(Image1,[]),title('Original Image');
subplot(2,2,2),imshow(Et,[]),title('Time gradient');
subplot(2,2,3),imshow(Ex,[]),title('Horizontal spatial gradient');
subplot(2,2,4),imshow(Ey,[]),title('Vertical spatial gradient');

% Calculate the normal flow at each pixel N=-Et/|grad E|
[r,c]=size(Image1);
N=zeros(r,c);
for i=1:r
    for j=1:c
        N(i,j)=-Et(i,j)/sqrt(Ex(i,j)^2+Ey(i,j)^2);
    end
end

% Display the resulting flow vectors as a 2D image. Overlay these
vectors and the gray level image.
uv=zeros(r,c,2);
for i=1:r-1
    for j=1:c-1;

A=[Ex(i,j),Ex(i,j+1),Ex(i+1,j),Ex(i+1,j+1);Ey(i,j),Ey(i,j+1),Ey(i+
1,j),Ey(i+1,j+1)];
        A=A';
%         Ax = A(:,1);
%         Ay = A(:,2);
        b=-[Et(i,j);Et(i,j+1);Et(i+1,j);Et(i+1,j+1)];
        uv(i,j,:)=-pinv(A'*A)*A'*b;
%         %uv(i,j,:) = - pinv(A)*b;
```

```
%          uv(i,j,1) = (Ax'*Ax)\(Ax'*b);
%          uv(i,j,2) = (Ay'*Ay)\(Ay'*b);

    end
end
```

Appendix C

```
% Regularization:
% 1. Horn and Schunck

function [u,v]=horn_schunck(I1,I2,nit)

Ix=conv2(I1,0.25*[-1,1;-1,1],'same')+conv2(I2,0.25*[-1,1;-
1,1],'same');
Iy=conv2(I1,0.25*[-1,-1;1,1],'same')+conv2(I2,0.25*[-1,-
1;1,1],'same');
It=conv2(I1,0.25*ones(2),'same')+conv2(I2,-0.25*ones(2),'same');

alpha=10;
[r,c]=size(I1);
u=zeros(r,c);
v=zeros(r,c);

for n=1:nit
    for i=2:r-1
        for j=2:c-1
            umean=0.25*(I1(i-1,j)+I1(i+1,j)+I1(i,j-1)+I1(i,j+1));
            vmean=0.25*(I2(i-1,j)+I2(i+1,j)+I2(i,j-1)+I2(i,j+1));


p=(Ix(i,j)*umean+Iy(i,j)*vmean+It(i,j))/(alpha^2+Ix(i,j)^2+Iy(i,j)
^2);
            u(i,j)=umean-Ix(i,j)*p;
            v(i,j)=vmean-Iy(i,j)*p;
        end
    end

end
end
```

Appendix D

Code for own video

```
clc;clear all;
close all;
```

```
vidObj = VideoReader('video_opt2.mp4');
% Determine the height and width of the frames.
vidHeight = vidObj.Height;
vidWidth = vidObj.Width;

% Create a MATLABÆ movie structure array, s.
s = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'),...
    'colormap',[]);
% Read one frame at a time using readFrame until the end of the file is reached.
% Append data from each video frame to the structure array.
k = 1;
while hasFrame(vidObj)
    s(k).cdata = readFrame(vidObj);
    k = k+1;
end
%%
I1=s(3).cdata;
I2=s(15).cdata;
I3=s(30).cdata;
I4=s(45).cdata;
I5=s(60).cdata;
%% RGB to gray
I1=double(rgb2gray(I1));
I2=double(rgb2gray(I2));
I3=double(rgb2gray(I3));
I4=double(rgb2gray(I4));
I5=double(rgb2gray(I5));
%%
% Show all the 5 image
figure,
subplot(3,2,1),imshow(I1,[]);title('Original Image 1');
subplot(3,2,2),imshow(I2,[]);title('Original Image 2');
subplot(3,2,3),imshow(I3,[]);title('Original Image 3');
subplot(3,2,4),imshow(I4,[]);title('Original Image 4');
subplot(3,2,5),imshow(I5,[]);title('Original Image 5');
%%
% images filtered by a Gaussian smoothing of sigma=1,
gauss_filtered_image1 = gaussian_filter(I1, 2.0);
gauss_filtered_image2 = gaussian_filter(I2, 2.0);
gauss_filtered_image3 = gaussian_filter(I3, 2.0);
gauss_filtered_image4 = gaussian_filter(I4, 2.0);
gauss_filtered_image5 = gaussian_filter(I5, 2.0);

% Apply this flow calculation
[uv1,N1]=optical_flow(gauss_filtered_image1,gauss_filtered_image2);
[uv2,N2]=optical_flow(gauss_filtered_image2,gauss_filtered_image3);
[uv3,N3]=optical_flow(gauss_filtered_image3,gauss_filtered_image4);
[uv4,N4]=optical_flow(gauss_filtered_image4,gauss_filtered_image5);

% Plot Normal
figure,
subplot(2,2,1),imshow(N1,[]),title('Normal Vector 1and2');
subplot(2,2,2),imshow(N2,[]),title('Normal Vector 2and3');
subplot(2,2,3),imshow(N3,[]),title('Normal Vector 3and4');
subplot(2,2,4),imshow(N4,[]),title('Normal Vector 4and5');
%%
%Plot optical flow
figure,
subplot(4,2,1),quiver(uv1(:,:,1),uv1(:,:,2),3,'-r');title('flow vectors 1');
subplot(4,2,2),quiver(uv1(:,:,1),uv1(:,:,2),3,'-r');title('Zoomed flow vector 1');
ylim([650 750]);xlim([800 900]);
subplot(4,2,3),quiver(uv2(:,:,1),uv2(:,:,2),3,'-r');title('flow vectors 2');
subplot(4,2,4),quiver(uv2(:,:,1),uv2(:,:,2),3,'-r');title('Zoomed flow vector 2');
```

```
ylim([650 750]);xlim([800 900]);
subplot(4,2,5),quiver(uv3(:,:,1),uv3(:,:,2),3,'-r');title('flow vectors 3');
subplot(4,2,6),quiver(uv3(:,:,1),uv3(:,:,2),3,'-r');title('Zoomed flow vector 3');
ylim([650 750]);xlim([800 900]);
subplot(4,2,7),quiver(uv4(:,:,1),uv4(:,:,2),3,'-r');title('flow vectors 4');
subplot(4,2,8),quiver(uv4(:,:,1),uv4(:,:,2),3,'-r');title('Zoomed flow vector 4');
ylim([650 750]);xlim([800 900]);
%%
% Plot overlay vector
figure,
subplot(2,2,1),imshow(I1,[]);hold on
quiver(uv1(:,:,1),uv1(:,:,2),3,'-r');
title('vectors over the grayimage 1');
hold off

subplot(2,2,2),imshow(I1,[]);hold on
quiver(uv2(:,:,1),uv2(:,:,2),3,'-r');
title('vectors over the grayimage 2');
hold off

subplot(2,2,3),imshow(I1,[]);hold on
quiver(uv3(:,:,1),uv3(:,:,2),3,'-r');
title('vectors over the grayimage 3');
hold off

subplot(2,2,4),imshow(I1,[]);hold on
quiver(uv4(:,:,1),uv4(:,:,2),3,'-r');
title('vectors over the grayimage 4');
hold off
```