EE 417

LAB#6 REPORT

OPTICAL FLOW

GOKTUG KORKULU

INTRODUCTION

This week's laboratory assignment's goal is to work and process over images (as always), but this time, over time interval. Images change due to the the relative motion between the scene and the camera. This *relative* motion can emerge by three different combination;

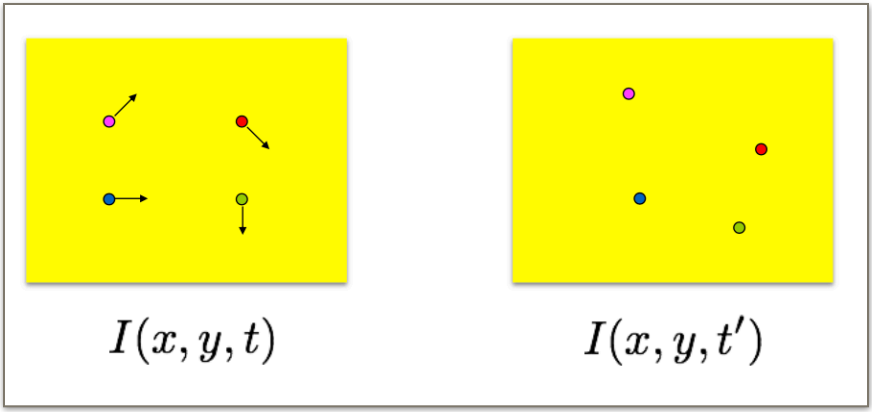
1) Camera still, scene moving
2) Camera moving, scene still
3) Both camera and the scene moving

We will work on relative motion with the help of *Optical Flow* notion.

Optical flow or **optic flow** is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. Optical flow can also be defined as the distribution of apparent velocities of *movement of brightness pattern* in an image.

The main reason why we try estimating the optical flow between two consecutive images (or, two consecutive video frames) is that we aim to understand and measure the velocities of objects in those consecutive images; in other words, in the video.

In order to do that, we need to make some simple assumptions. First, we will assume that the apparent brightness of the moving objects remain constant. The other assumption is that, we will need the location change between two consecutive images will be significantly small. In other words, pixels will move only a little bit.



For instance; above, the color (for the notion of brightness) are same for both of the images [Assumption 1]. Also notice that, objects make really small moves [Assumption 2]. Note that t' - t is the time between those two image frames.

By further investigation of these key assumptions, we will come up with the equations below of estimating the optical flow of the objects.

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

Brightness Constancy Equation

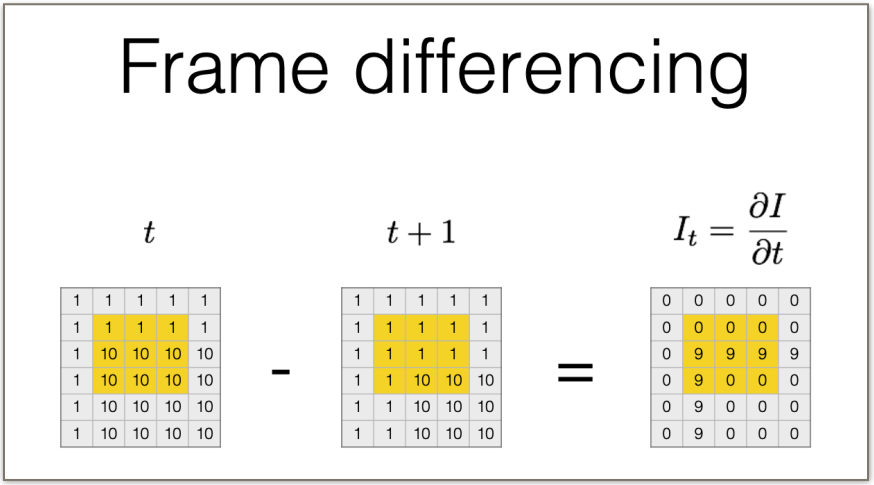
$$I_x u + I_y v + I_t = 0$$

(x-flow) (y-flow)

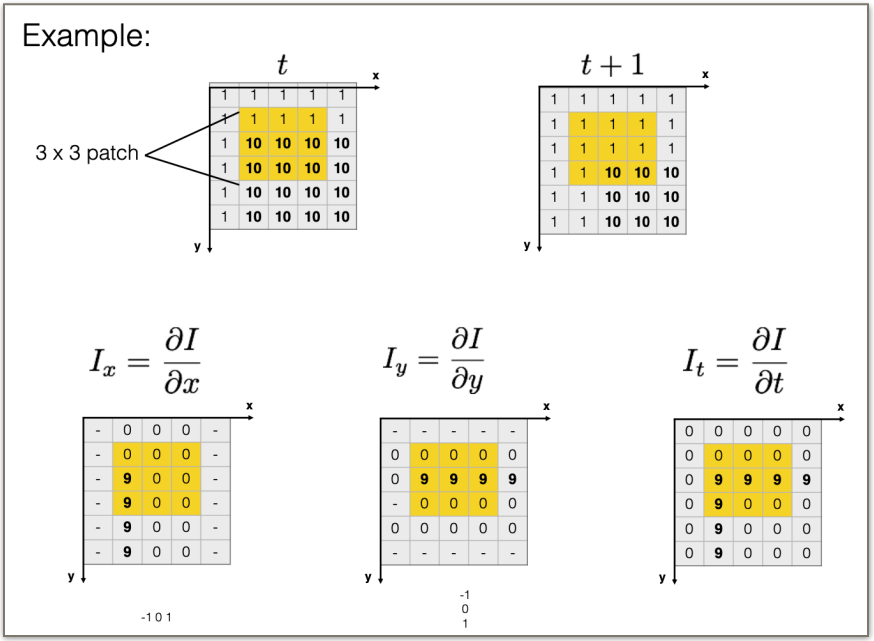
shorthand notation

Note that ***Ix*** and ***Iy*** are image gradients with respect to c and y coordinates respectively. ***It*** is temporal gradient which

can be measured by frame differencing, as shown below. ***u*** and ***v*** are optical flow velocities with respect to x and y axis.



Below, there is an example of how image gradients (Ix and Iy) and temporal gradient(It) is measured for his specific image frames.



In order to move further on estimating the optical flow of the objects, more specifically ***u*** and ***v*** values; we need to proceed 1 more step and make another assumption. The assumption title that we should make is 'constant flow' and we should assume that the surrounding patch (3x3, 5x5, 7x7, …) has that 'constant flow'. Which deeply means that neighboring pixels have same displacement and flow is locally smooth. Therefore, simply;

$$\underset{A^\top A}{\begin{bmatrix} \sum_{p\in P} I_x I_x & \sum_{p\in P} I_x I_y \\ \sum_{p\in P} I_y I_x & \sum_{p\in P} I_y I_y \end{bmatrix}} \underset{\hat{x}}{\begin{bmatrix} u \\ v \end{bmatrix}} = - \underset{A^\top b}{\begin{bmatrix} \sum_{p\in P} I_x I_t \\ \sum_{p\in P} I_y I_t \end{bmatrix}}$$

In order to solve this for ***u*** and ***v*** values; we need to modify the matrix equation and obtain below equation as a final step.

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

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

Note that, ATA is replaced by G and ATb by b in above equations. Consider them as equals for both equations presented above.

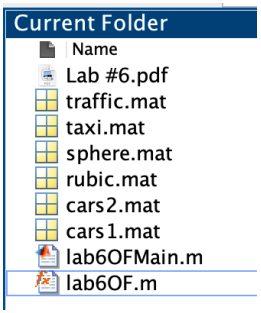
METHOD

There will be two MATLAB files that was given to us and we will modify some missing integral parts in order to make those files to run properly. The files "***lab6OFMain.m***" and "***lab6OF.m***" shown below consecutively;

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

```
function lab6OF(ImPrev,ImCurr,k,Threshold)
% Smooth the input images using a Box filter
% Calculate spatial gradients (Ix, Iy) using Prewitt filter
% Calculate temporal (It) gradient
[ydim,xdim] = size(ImCurr);
u = zeros(ydim,xdim);
v = 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
    if (min(eigenvalues of G) < Threshold)
      u(cy,cx)=0;
      v(cy,cx)=0;
    else
      % Calculate (u,v)
      u(cy,cx)=OF(1);
      v(cy,cx)=OF(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,u,v,10,'r');
colormap gray;
end
```

lab6OF.m is a function to calculate the optic flow of the input images as follows that were called inside the ***lab6OFMain.m.***



Before proceeding any further, let me show you the content of the file that was given to us for this lab study. Note that, .mat files are consecutive image sequence matrixes which will be used to measure their optical flow. For example, traffic.mat is 512x512x49 sized matrix which implies each image frames are 512x512 pixels and there are 49 consecutive image frames for this specific example.

STEP 1: First, let's start with ***lab6OFMain.m.*** Here, we need to write codes corresponding to the comment lines in the given bunch of codes. As a beginning, I will do the first instruction given to me as *"% Load the files given in SUCourse as Seq variable"*:

lab6OFMain.m * | lab6OF.m

```
clear all; close all; clc;

%% Load the files given in SUCourse as Seq variable
load('rubic.mat');
Seq = rubic;

%load('traffic.mat');
%Seq = traffic;

%load('taxi.mat');
%Seq = taxi;

%load('sphere.mat');
%Seq = sphere;

%load('cars1.mat');
%Seq = cars1;

%load('cars2.mat');
%Seq = cars2;

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

```

Note that, since there are 6 different image frame sequences given to us for the lab, I needed to load each of them one by one with the order that I will measure its optical flow and initialized the ***Seq*** variable according to their name. At the end of this step, I have stored the pixel amount of both x and y direction as ***row*** and ***col***, and the amount of consecutive images as ***num*** in that specific .mat variable. For instance, 'rubic.mat' is 240x256x20 sized matrix and therefore ***row = 240, col = 256, num = 20***.

STEP 2: In the next step, I am asked to define k and Threshold values. It is hard to define these values before trying several different combinations of them for each distinct image sequences. That's why, I will set their values beforehand when I finish the remaining codes. But, for the sake of completeness, I will add a screenshot of first two k and threshold values for rubic.mat and traffic.mat since I have already decided which values are the most proper ones. Later, I will mention about other .mat files' k and threshold values as well.

```
%% Define k and Threshold
%for rubic
k=15;
Threshold = 3000;
%

%for traffic
% k=20;
% Threshold = 10000;
%
```

Note that, for rubic.mat image sequence's proper optical flow visualization, I have set k=15 and Threshold=3000; for traffic.mat image sequence's proper optical flow visualization, I have set k=20 and Threshold = 10000.

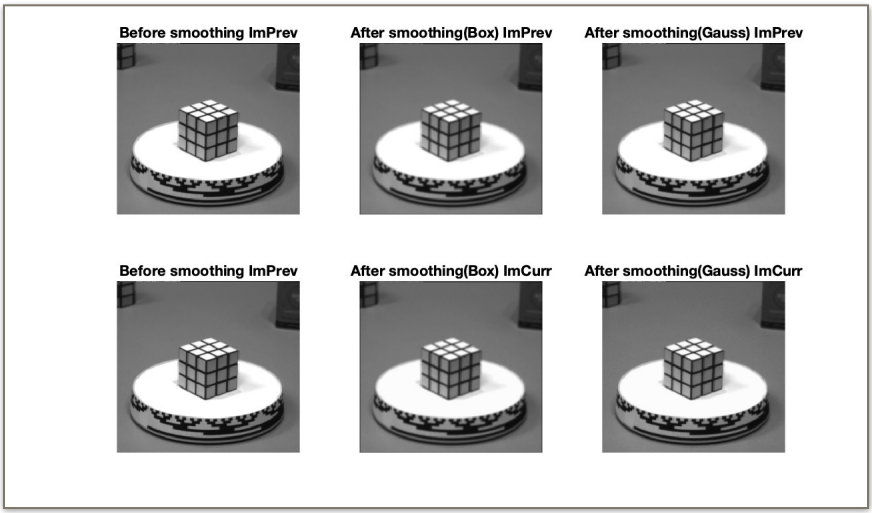
STEP 3: Now, let's fill the missing codes in the ***lab6OF.m*** function. First of all, we are asked to smooth the images using a Box filter. However, in the postlab part, we are also asked to use Gaussian filter and compare the results of them. Now, I will start with using Box filter.

lab6OFMain.m | lab6OF.m *

```
function lab6OF(ImPrev,ImCurr,k,Threshold)

%% Smooth the input images using a Box filter (+ Gaussian filter for postlab)
ImPrev = double(ImPrev);
ImCurr = double(ImCurr);

kernel = 1/9 * [1 1 1; 1 1 1; 1 1 1];
conv2(ImPrev, kernel, 'same');
conv2(ImCurr, kernel, 'same');

% ImPrev = imgaussfilt(ImPrev);
% ImCurr = imgaussfilt(ImCurr);

```

Notice here, first, I have converted the integer pixel values to double so that when I apply smoothing filtering, I will not lose any decimal values.

Then, I have created 3x3 Box filter kernel as the variable name ***kernel.*** Finally, I have convolved both previous and current images which were taken from function call as an input as ***ImPrev*** and ***ImCurr***.



Here is an example of smoothing the previous and current images using both Box and Gauss filter. You may notice that box filter blurs the details more than the Gauss in this specific example. But in order to understand which one is better in terms of getting rid of the noises so that the ultimate output will be effected by noises lesser; we need to analyze the outputs with optical flow visualization at the end. Later, we will talk about it but for now, I will proceed with Box filter.

STEP 4&5: Now, lets continue and calculate spatial gradients (Ix and Iy) and temporal gradient(It).

Notice the fact that I will be using **[Ix, Iy] = *imgradientxy(I, method)*** built-in MATLAB function which returns the directional gradients *Ix* and *Iy* of the grayscale or binary image *I* using the specified *method*. Here is the code for it:

```
23    %% Calculate spatial gradients (Ix, Iy) using Prewitt filter
24    [Ix, Iy] = imgradientxy(ImPrev,'Prewitt');
25    Ix = -Ix;
26    Iy = -Iy;
27    %% Calculate temporal (It) gradient
28    It = ImPrev - ImCurr;
29
```

Notice, we will only use spatial gradients of the image before the time change, i.e. ImPrev in our case since we only need that values in order to measure the optical flow. We are not going to use ImCurr's gradient values on calculating the optical flow.

IMPORTANT!! In the introduction part of this lab report, I have given an example of calculating the Ix, Iy and It values for a specific image frames. Notice that the image gradient calculation vectors were [-1 0 1] and [-1;0;1] for Ix and Iy respectively. However, since the default gradient calculation

vectors are [1 0 -1] and [1;0;-1] in ***imgradientxy*** function, we need to change the sign of Ix and Iy in order to have prover directional informations. That's why the signs of Ix and Iy are set exactly opposite.

Also, calculating temporal gradient matrix is as easy as subtracting the current image matrix from previous image frame.

STEP 6: Inside two nested for loop that was given to us already, we are requested to calculate the elements of G and b and then calculate the eigenvalues of G to use it at one code below.

```
%% Calculate the elements of G and b
G = [sum(sum(Ix(y-k:y+k, x-k:x+k).^2)) sum(sum(Ix(y-k:y+k, x-k:x+k).*Iy(y-k:y+k, x-k:x+k)));
    sum(sum(Ix(y-k:y+k, x-k:x+k).*Iy(y-k:y+k, x-k:x+k))) sum(sum(Iy(y-k:y+k, x-k:x+k).^2))];

b = [sum(sum(Ix(y-k:y+k, x-k:x+k).*It(y-k:y+k, x-k:x+k))); sum(sum(Iy(y-k:y+k, x-k:x+k).*It(y-k:y+k, x-k:x+k)))];

eigenvalues = eigs(G);
```

Here, I have implemented the code according to exactly what ehe equation of G and b was given in the introduction part as ;

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

This is a direct implementation, nothing to have further comments about.

Step 7: As the final missing part of the codes, we are asked to calculate the ***u*** and ***v*** components optical flow vector.

```
%% Calculate (u,v)
OF = (-1)*(G^(-1))*b;
u(cy,cx)=OF(1);
v(cy,cx)=OF(2);
```

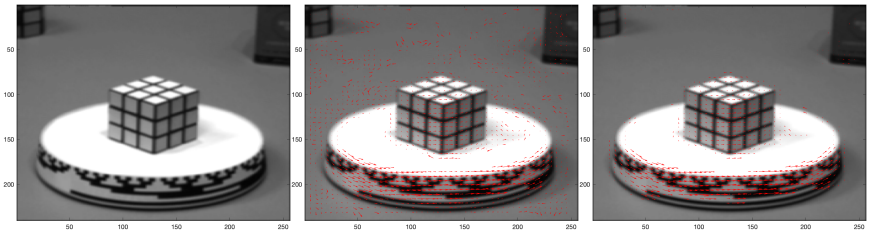
Again, the coding was easy since the equation of u and v are already known and all I needed to do was just type the equation to MATLAB with respect to the syntax. Below is the equation of u and v.

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

OUTCOMES AND CONCLUSION

In this chapter, I will go over each .mat extended image frame sequences and apply the code I have written just now. I will separate the distinct image frame sequences and there will be 6 subtitle in this chapter.

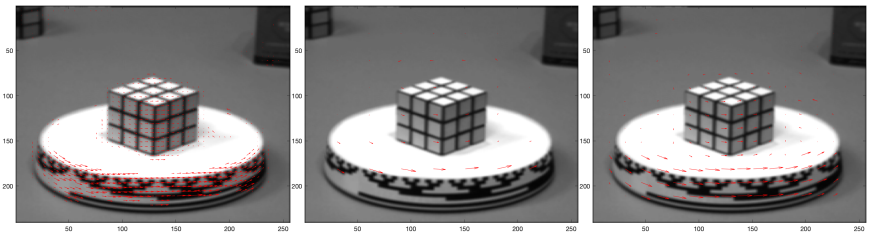
1) rubic.mat



k(5) Threshold(3,000,000) k(5) Threshold(300) k(5) Threshold(3000)

I have set k(5) and Th(3,000,000) for the first try and didn't see any arrow in the beginning. I thought that Th value might be too high, then tried to lover it to 300; and now I see too much arrows that is giving me wrong results just because Th value is too low now. Then, as a third guess, I have set Th to 3000 and this one is good enough.

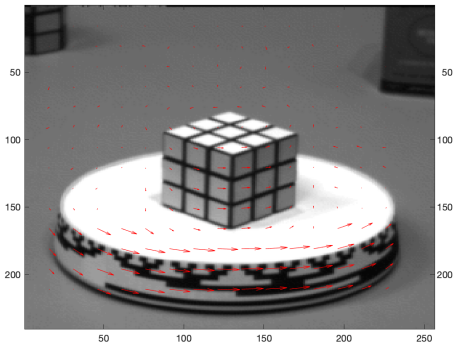
Then, I needed to see the arrows and the direction more clear, so changed the k value by keeping the Th value still.



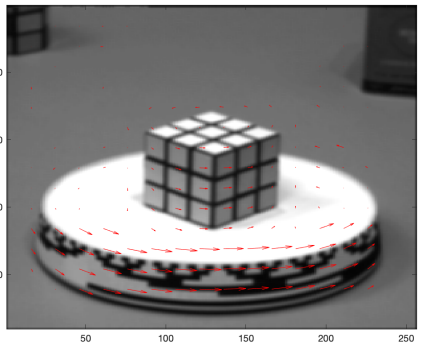
k(5) Threshold(3000) k(30) Threshold(3000) k(15) Threshold(3000)

Now, it seems like k=5 when Th = 3000 is too small and k=30 seems too large. However, k=15 and Threshold=3000 gives the best measurement of the flow and the arrows are the best represented in this values. For that reason, my optimal k and Threshold values are 15 and 3000 respectively.

As a one last comparison, I have changed the filter from Box filter to Gaussian Filter and try to compare the outcomes. Here are the Box filtered and Gaussian filtered images:



k(15) Threshold(3000)
Gaussian Filtered



k(15) Threshold(3000)
Box Filtered

Notice that, Optical Flow representation on Gaussian Filtered image sequences give more errors on this k and Threshold values. Although the image is less blurred and better for visualization, I'd choose Box filter in order to filter my images beforehand.

2) traffic.mat

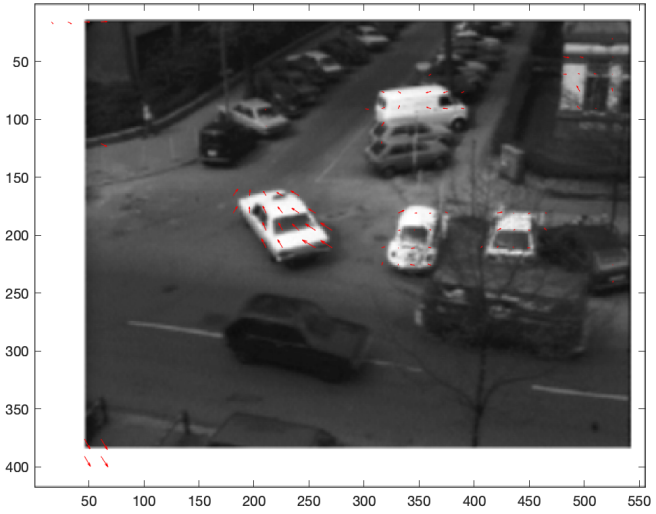
k(10) Threshold(1,000,000) k(20) Threshold(1,000,000) k(30) Threshold(1,000,000)

I would choose k value as 10 since the arrows corresponding to moving cars are the most proper one. k=20 and k=30 is not that detailed. In other words, although only cars are moving, k=20 and k=30 ones show like the neighbor areas are moving as well, which is not we want. To sum up, in order to detect the moving objects within the given frames, I'd choose k=10 in this specific example.

k(10) Threshold(10,000) k(10) Threshold(10,000,000) k(10) Threshold(1,000,000)

When threshold value is 10000, it is too small; when it is 10000000, it is too large. It seems that the optimal value is k=10 and Threshold = 1,000,000

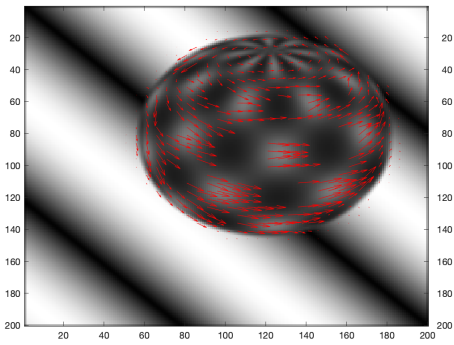
3) taxi.mat



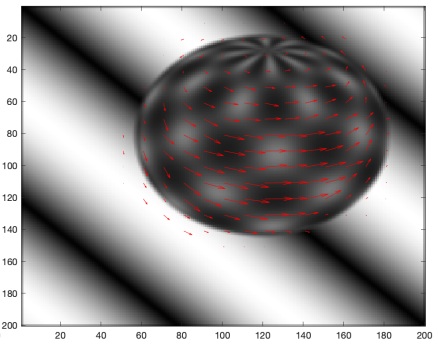
k(15) Threshold(1,000,000)

I have investigates several cases as I did to previous image frames as well and came up with the idea that this is the best result.

4) sphere.mat



k(5) threshold(10,000)



k(10) Threshold(10,000)

I will choose the right one with k=10 and threshold=10000 since the left one doesn't give the optical flow properly for the black regions.

5) cars1.mat

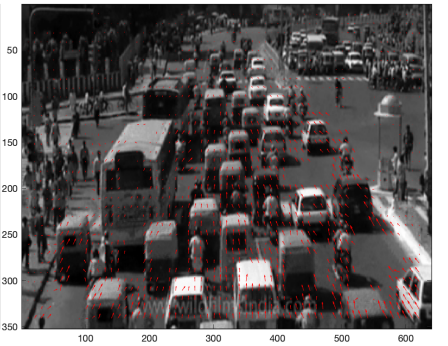
k(5) Threshold(10000)

k(10) Threshold(10000)

On the right, there are a lot of false positive optical flow rays where the left one is way better to detect the flows. Therefore, k=5 and Threshold=10,000 is my optimal values.

6) cars2.mat

k(15) Threshold(1,000,000)



k(10) Threshold(100,000)

Depends on one's preference, both k and Threshold values might be chosen. For this specific example, my preference would be on the left one since the left one having less false positive optical flow rays.

In this week's lab, I have introduced the notion of the optical flow from the very beginning of it. Then, gave some reasonings for the equations that are being used in order to measure the optical flow parameters.

Then, step by step, I have analyzed and showed my own codes that are being inserted into the missing parts of the given raw codes. One by one, I have explained what and why I wrote in those gaps. Sometimes, I showed the different options for the same operation such as filtering operation; and compared the results of those options within that step. Time to time, I have referred to the introduction part in order to relate the code that I have written and the equations/ reasonings.

Finally, I have used the codes on the given 6 distinct sequences of image frames in order to visualize the results. Thus, I have mentioned my reasonings why and how I have made my choices on choosing the k and Threshold values.