EE417 - Assignment 5 - Post-Lab Report

Baris Sevilmis
Lab Date: 27/11/2018
Due Date: 03/12/2018

Overview

Sixth lab assignment of EE417 focuses on Motion and Optical Flow. In this assignment, velocities of the objects through video frames will be detected given their small baseline. Measurement of velocities will computed by estimated optical flows. Algorithm will be operated on distinctive video frames in order to discuss results more properly.

Part 1: Motion and Optical Flow

Optical Flow is one of the main properties of the Motion Detection and Feature Tracking. It is related to the aperture problem, in which given an diagonal movement of the 3D object aperture, we can only detect normal component of the movement bu not the tangential movement. With the help of this information, optical flow can be specified as the this detected movement by the aperture of camera. In other words, optical flow becomes the distribution of the apparent velocities of objects in an provided video frame. As a conclusion, optical flow aids procedure of feature tracking in videos. Feature tracking algorithm in current Lab Assignment is one of the most basic versions of Motion detection and optical flow estimation such that video frames consist of small baselines, namely changes between images are minimal. Therefore, following formula is provided for small baselines, in which t stands for time frame and x(t) for image at specific time frame:

$$I(x(t), t) = I(x(t) + udt, t + dt)$$

By approximating Right Hand Side(RHS) of the formula by first two terms of Taylor series, we result in the following formula:

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

Formula above is called Brightness Constancy Constraint(BCC)/Brightness Constraint Assumption(BCA) and is crucial for optical flow approximation. $I_t(x(t), t)$ represents the temporal derivative due to time alteration. On the other hand, u represents the motion along y and x axis. However, unknown u has two components of $[u_x, u_y]$ and there exists only one equation for determining u. In order to solve this ill-posed issue, an image patch/window is chosen, in which algorithm proceeds. Following formula is computed by integration around over the given image patches:

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

Derivative of the following formula is taken, as spatial gradients and temporal gradient of the image patch are computed:

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

From the given operation, succeeding operation is formed, in which G and b stands out as the above matrices.

$$Gu + b = 0$$

$$G = \begin{bmatrix} (I_x)^2 I_x I_y \\ I_y I_x (I_y)^2 \end{bmatrix} b = \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix}$$

Stated constraints lead to this equation, and as it is obvious u can be calculated by the inverting above formula:

$$u = -G^{-1}b$$

G ensures good feature detection conceptually given its rank. Moreover, rank of 0 refers to blank wall problem, rank of 1 refers to aperture problem and rank of 2 refers to the good feature candidates such as corners. However, this is conceptual and in reality threshold may involvement changes the situation.

As the optical flow algorithm is provided, it would be reasonable to demonstrate the results of the algorithm and corresponding MatLab code. Listing 2 provides the optical flow function, and contains all the necessary calculations to measure the velocities along the image frames. All the explained steps are computed in Listing 2. On the other hand, Listing 1 ensures the main function of the algorithm in terms of activating Optical Flow function.

As for the results, various window sizes will be tried with $k = 10, 20, 30$, to be more specific different sized image patches. Image will be smoothed out with Box Filter and Gussian Filter in terms of noise reduction before the algorithm proceeds. For testing purposes, threshold will be set to 100, as changing threshold does not result in huge variations such as other parameters in results . For the sake of simplicity, Threshold will be set to a constant:

$$Threshold = 100$$

As an important final detail, same image frames will be given in the results for comparison purposes. On some of the resulting images sequence may be differ a bit, though results are still comparable because of the almost exact velocity changes in both of the frames.

Listing 1: MatLab Code for Main Function

```matlab
clear all; close all; clc;
% Load the files given in SUcourse as Seq variable
load('sphere.mat');
Seq = sphere;
[row,col,num]=size(Seq);
% Define k and Threshold
k = [10 20 30];
Threshold = 100;
for j=2:1:num
    ImPrev = Seq(:,:,j-1);
    ImCurr = Seq(:,:,j);
    lab6OF(ImPrev,ImCurr,k(1),Threshold);
    pause(0.1);
end
```

Listing 2: MatLab Code for Optical Flow function

```
   function lab6OF(ImPrev,ImCurr,k,Threshold)
    % Smooth the input images using a Box filter
    filt = ones(5,5);
    %filt = [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];
    ImPrevS = conv2(ImPrev, filt , 'same');
    ImCurrS = conv2(ImCurr, filt , 'same');
    % 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(ImCurrS, x_filter,'same');
    Iy = conv2(ImCurrS, y_filter,'same');
    % Calculate temporal (It) gradient
    It = ImPrevS - ImCurrS;
    [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
            Ixt = Ix(y-k:y+k, x-k: x+k);
            Iyt = Iy(y-k:y+k, x-k: x+k);
            Itt = It(y-k:y+k, x-k: x+k);
            G = [sum(sum(Ixt.^2)) sum(sum(Ixt.*Iyt));sum(sum(Ixt.*Iyt)) sum(sum(Iyt.^2))];
            b = [sum(sum(Ixt.*Itt)); sum(sum(Iyt.*Itt))];
            if (min(eig(G)) < Threshold)
                Vx(cy,cx)=0;
                Vy(cy,cx)=0;
            else
                % Calculate u
                u = (-1*pinv(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
```

Figure 1 depicts motion detection and optical flow estimation on Sphere image with k = 10. Although real data consists of sequential image frames, one of the best image frames is chosen in terms of demonstrating optical flow approximations. Figure 2 depicts same image with k = 20. It is obvious that there are lines are decreased as the image patches are greater. Lastly, Figure 3 demonstrates same optical flow detection with k = 30. It is clear that there are only a few lines left as the image patch increased. Finally, Figure 4 demonstrates same Optical Detection after Gaussian Smoothing instead of Local Mean Smoothing with k = 10. Although there are no big differences for this image frame, distribution of velocities differ when compared to box filtered image as frequency distribution of the image changes.

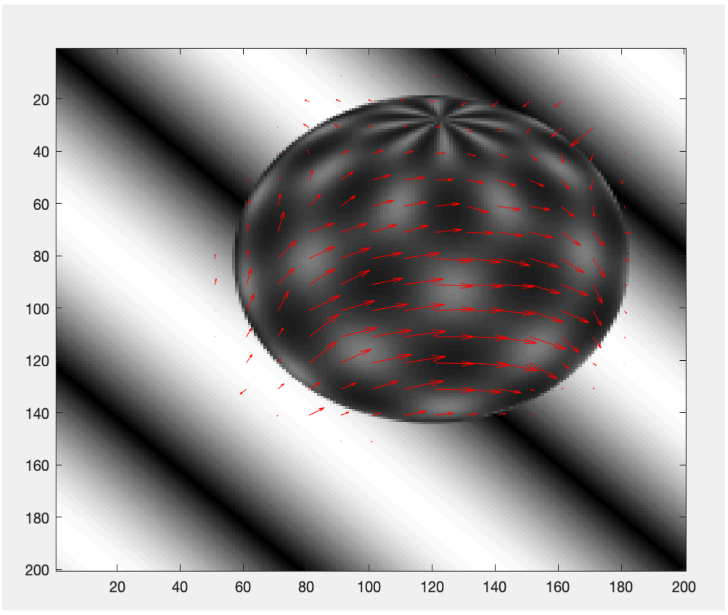

Figure 1: Optical Flow Detection on Sphere with k = 10 with Box Filter

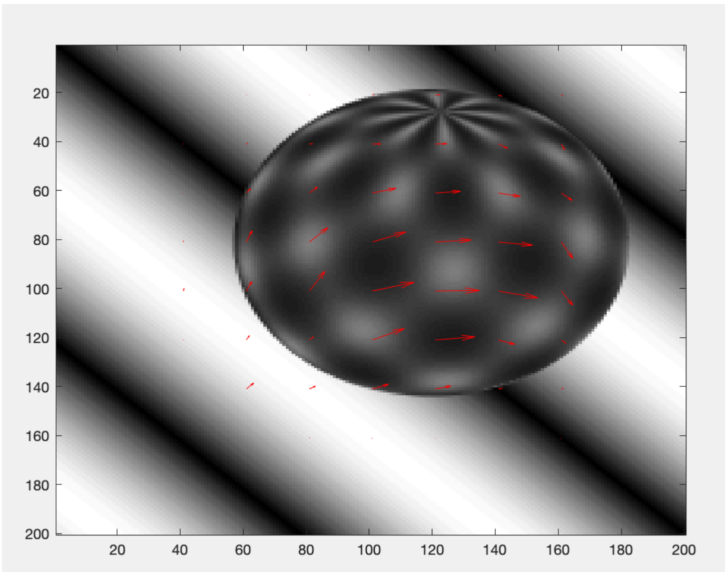

Figure 2: Optical Flow Detection on Sphere with k = 20 with Box Filter

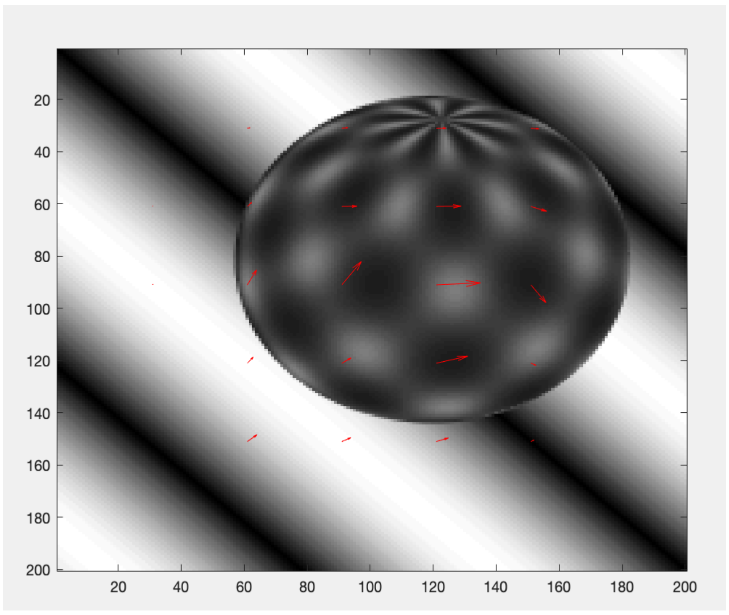

Figure 3: Optical Flow Detection on Sphere with k = 30 with Box Filter

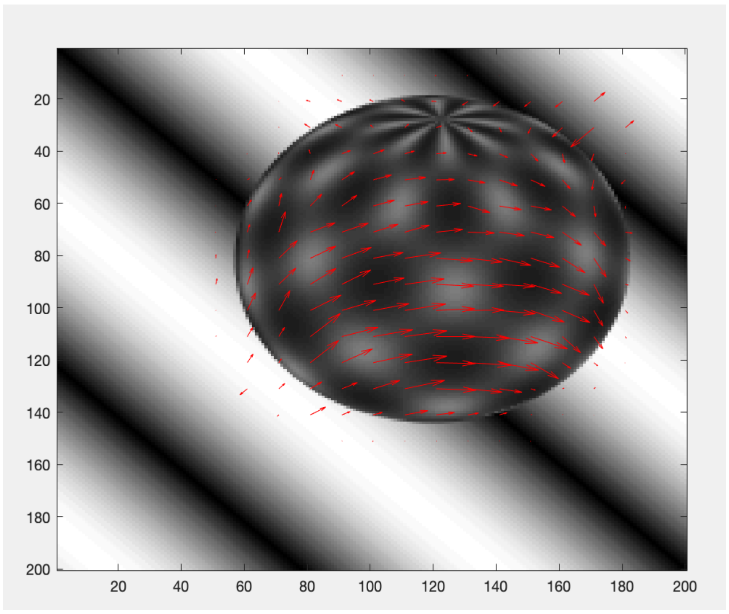

Figure 4: Optical Flow Detection on Sphere with k = 10 with Gaussian Filter

Figure 5, 6, 7 and 8 are the optical flow detection on an rubic image frame sequence. Ordering of the image patch sizes are same as in Figure 1, 2 and 3. Figure 8 will depict optical flow detection after Gaussian smoothing with k = 10 instead of Box Filtering. In order to remind again, same image frames are used as results in provided Figures.

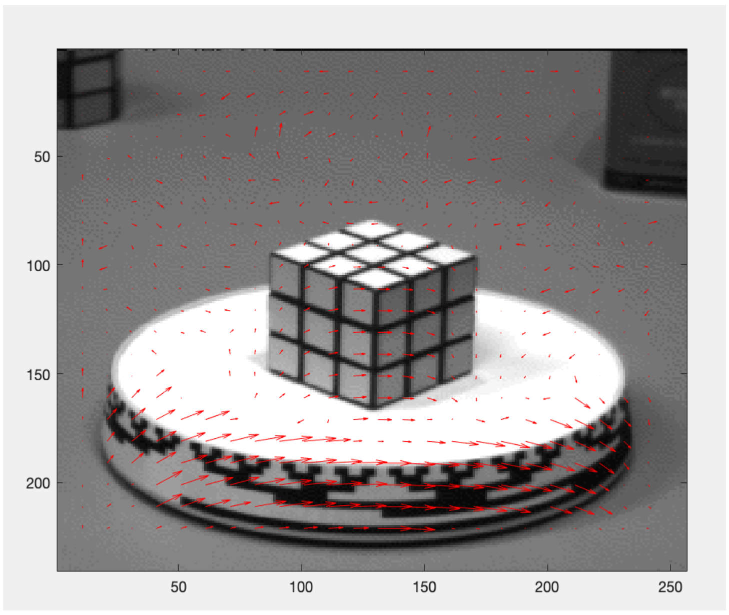

Figure 5: Optical Flow Detection on Rubic with k = 10 with Box Filter

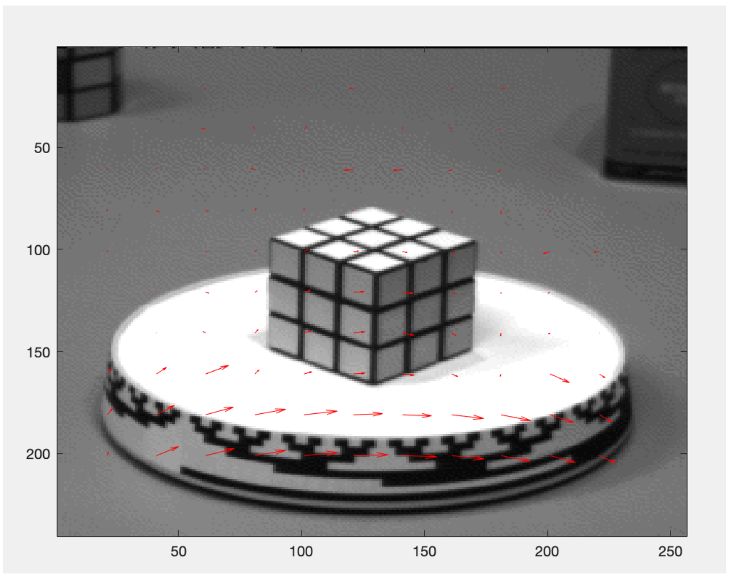

Figure 6: Optical Flow Detection on Rubic with k = 20 with Box Filter

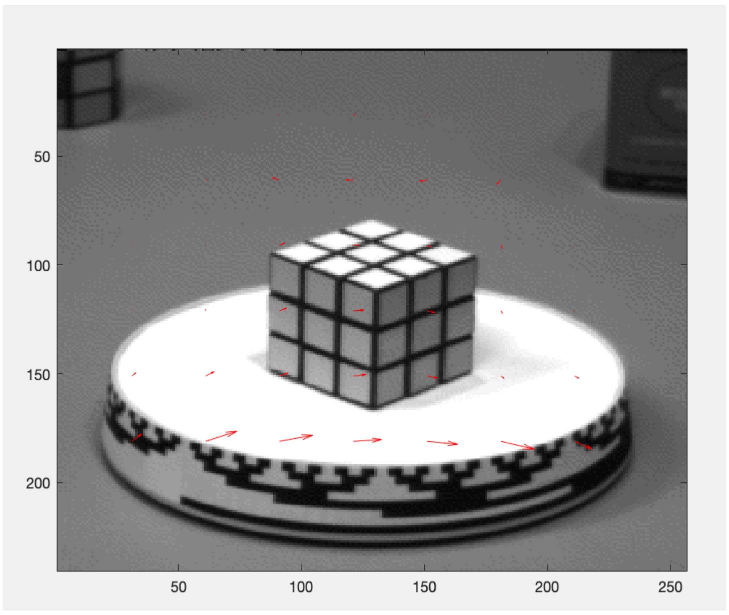

Figure 7: Optical Flow Detection on Rubic with k = 30 with Box Filter

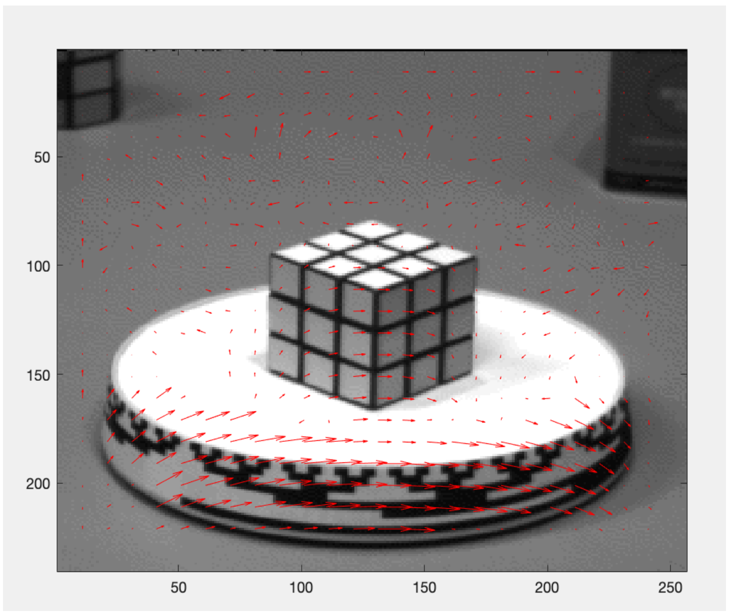

Figure 8: Optical Flow Detection on Rubic with k = 10 with Gaussian Filter

Figure 9, 10, 11 and 12 demonstrate optical flow detection on Cars-I image frame. Image frames are same with each other. Figure 9, 10, and 11 uses box Filter with consecutive k parameters of 10, 20 and 30. Figure 12 depicts image patch with k = 10 and uses Gaussian Smoothing.

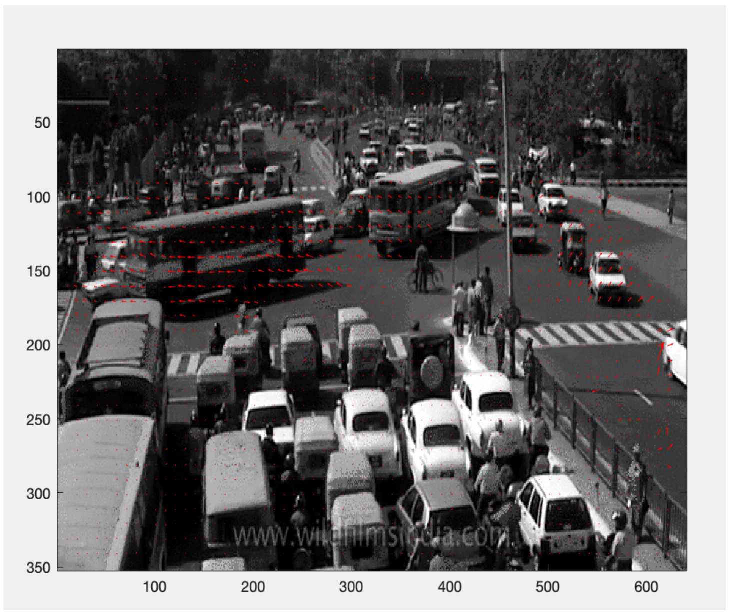

Figure 9: Optical Flow Detection on Cars-I with k = 10 with Box Filter

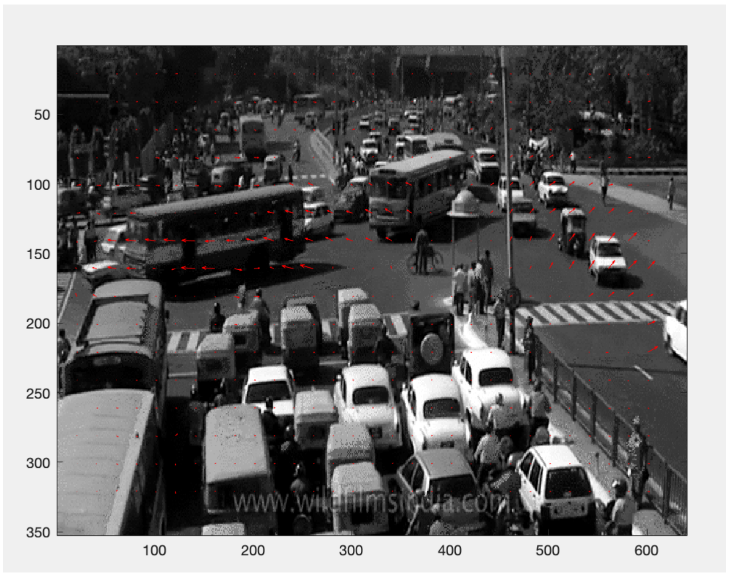

Figure 10: Optical Flow Detection on Cars-I with k = 20 with Box Filter

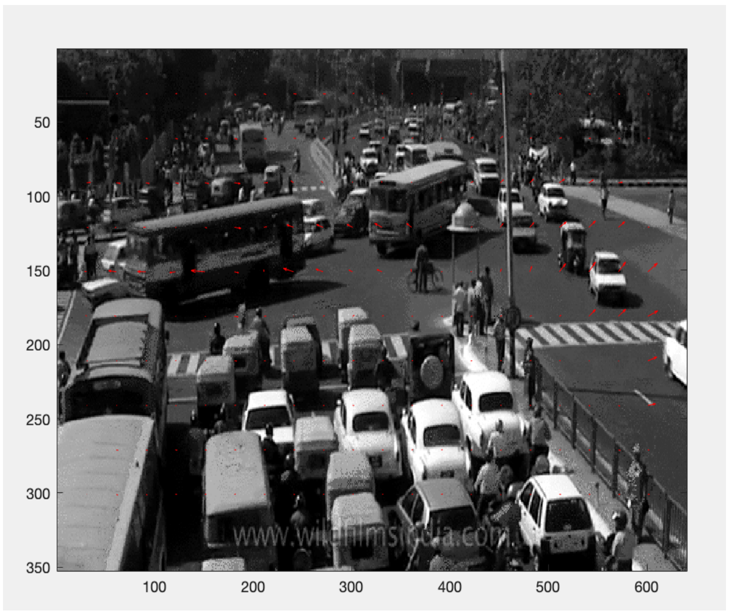

Figure 11: Optical Flow Detection on Cars-I with k = 30 with Box Filter

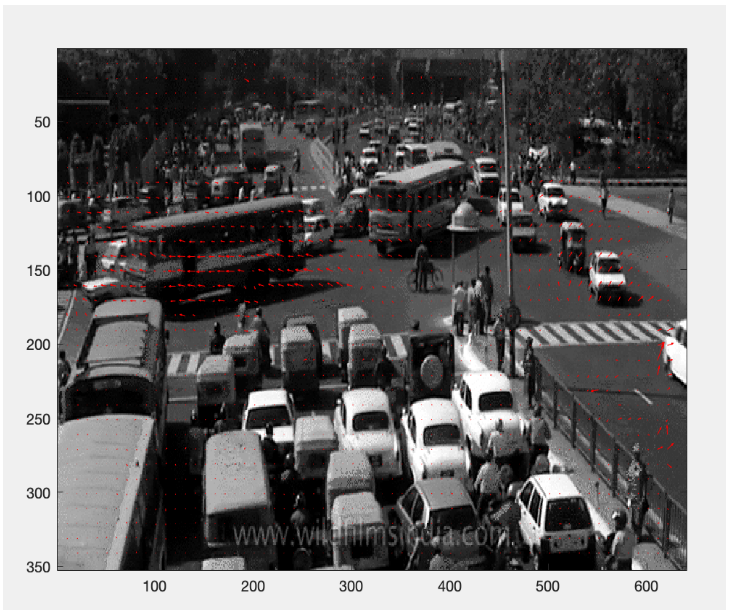

Figure 12: Optical Flow Detection on Cars-I with k = 10 with Gaussian Filter

Figure 13, 14, 15 and 16 are same set of operations on Cars-II image. Operation ordering and sorting of parameters is same as previous consecutive Figures as well as the image frames that are used. Although velocity lines are more tough to see because of the image itself, it is clear that results are same as before. As k increases, optical flow amount decreases and different smoothing techniques cause variations in optical flow depictions as the frequency rates change.

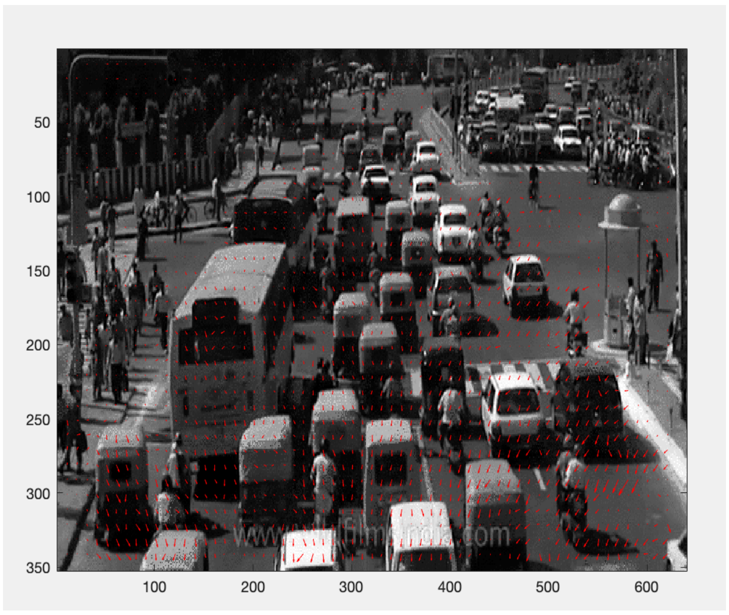

Figure 13: Optical Flow Detection on Cars-II with k = 10 with Box Filter

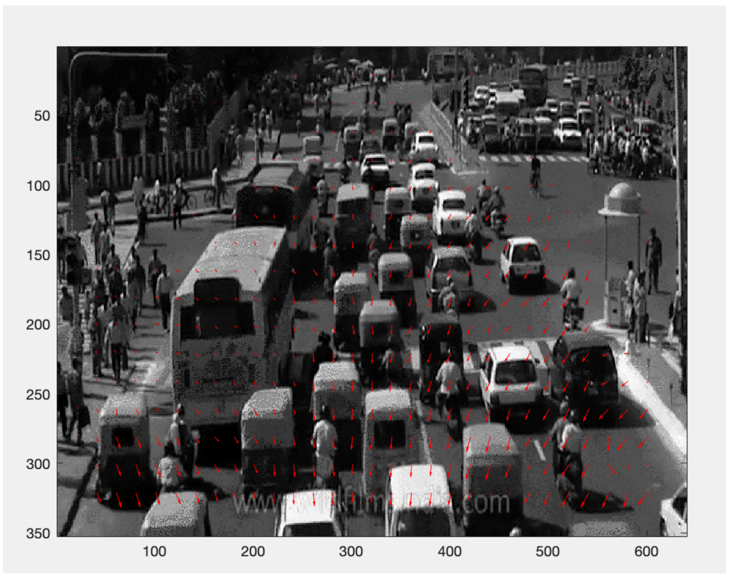

Figure 14: Optical Flow Detection on Cars-II with k = 20 with Box Filter

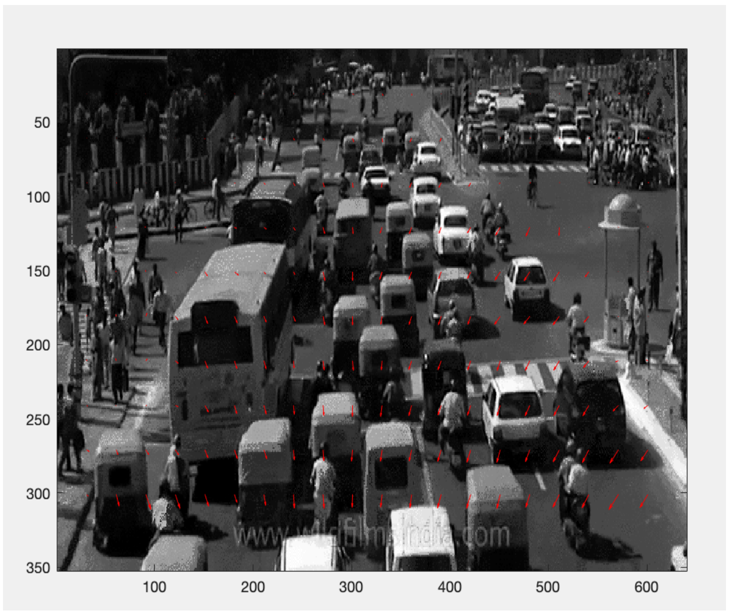

Figure 15: Optical Flow Detection on Cars-II with k = 30 with Box Filter

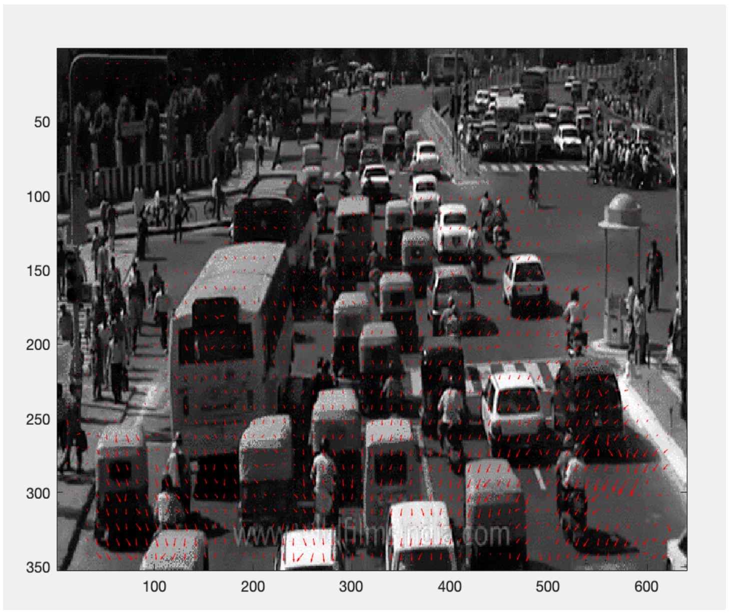

Figure 16: Optical Flow Detection on Cars-II with k = 10 with Gaussian Filter

Figure 17, 18, 19 and 20 show the same Optical Flow results on Taxi Image. As the operation order and sorting of parameters are same, same set of image frames are again used for comparison. In comparison to Cars-II, image frames of Taxi show more accurate results, in which optical flow detection is much more clearly shown.

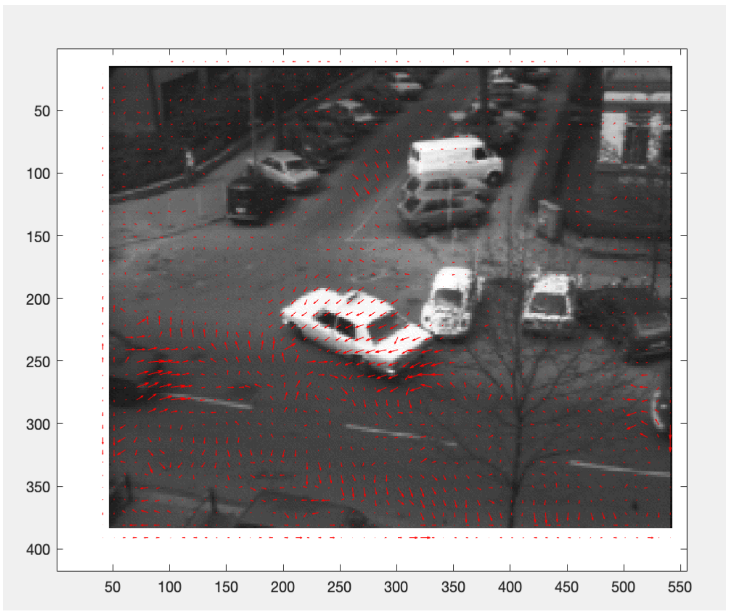

Figure 17: Optical Flow Detection on Taxi with k = 10 with Box Filter

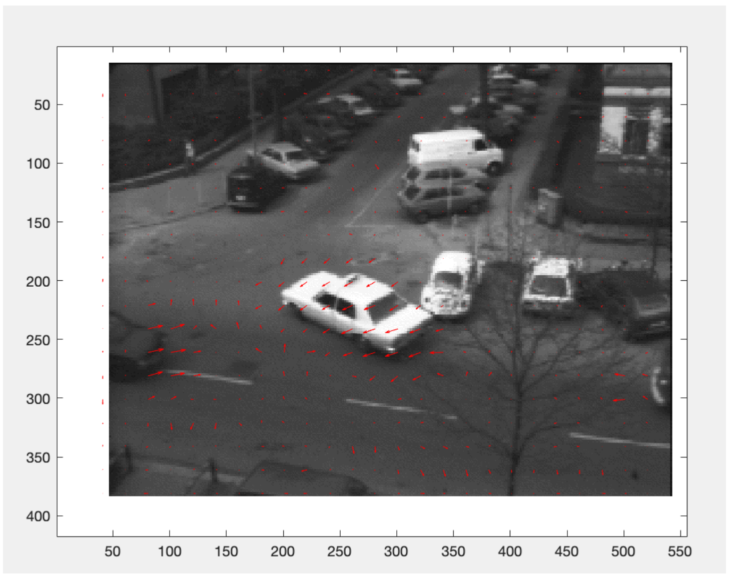

Figure 18: Optical Flow Detection on Taxi with k = 20 with Box Filter

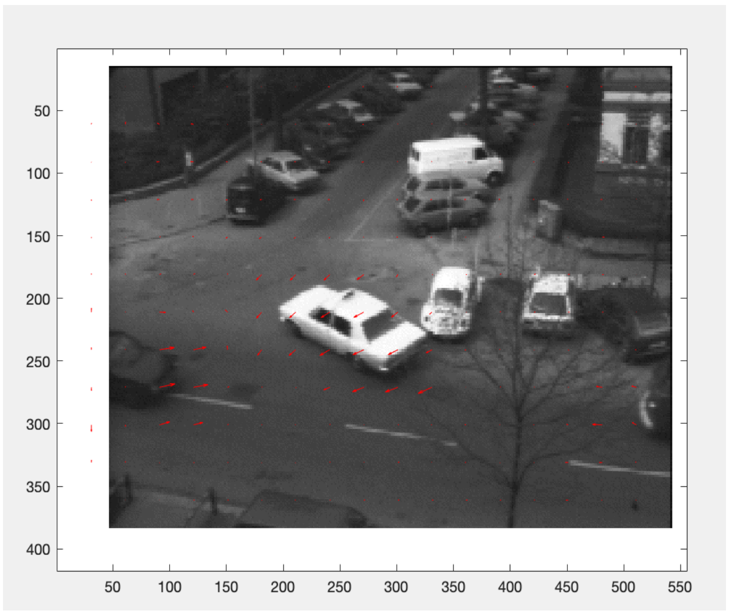

Figure 19: Optical Flow Detection on Taxi with k = 30 with Box Filter

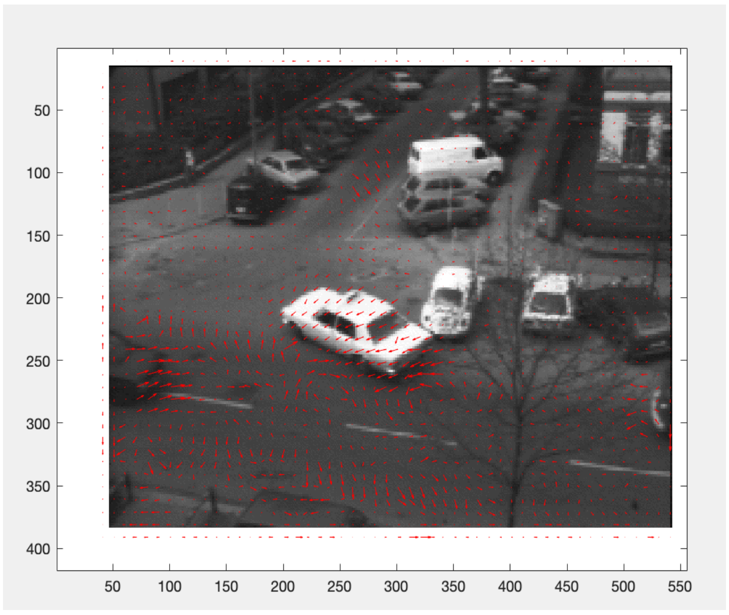

Figure 20: Optical Flow Detection on Taxi with k = 10 with Gaussian Filter

To conclude, Figure 21, 22, 23 and 24 will be last set of images to demonstrate the results of our basic Optical Flow algorithm on image Traffic. Sorting of parameters and operation ordering are identical, therefore the results of the images will be similar to previous Figure sequences. With increasing k, image patches increase. With Gaussian Smoothing frequency distribution of the image frames change. Therefore, resulting, image frames vary in their optical flow detection.

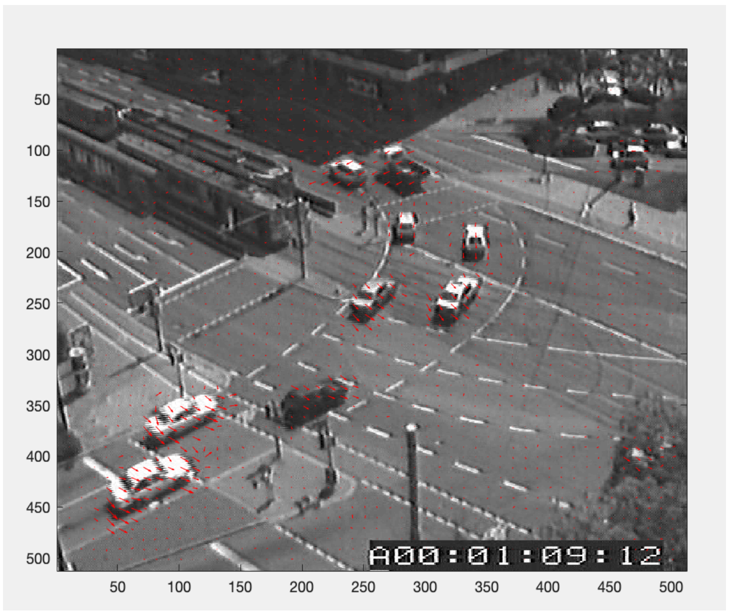

Figure 21: Optical Flow Detection on Traffic with k = 10 with Box Filter

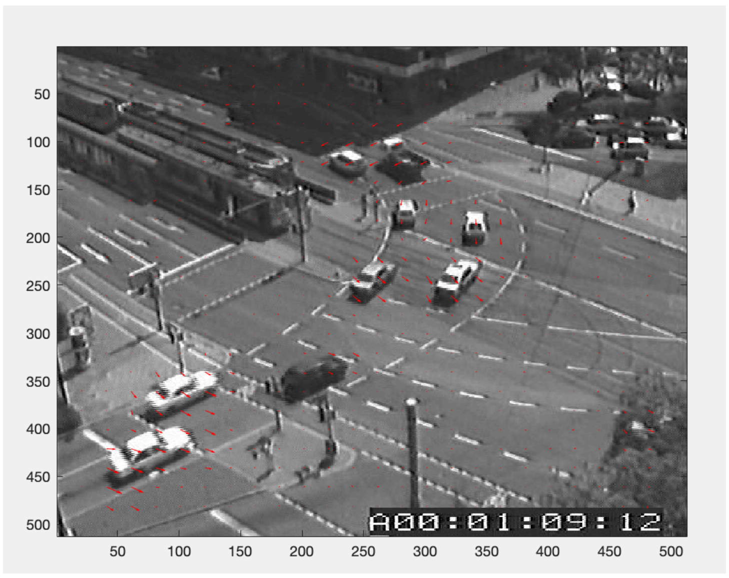

Figure 22: Optical Flow Detection on Traffic with k = 20 with Box Filter

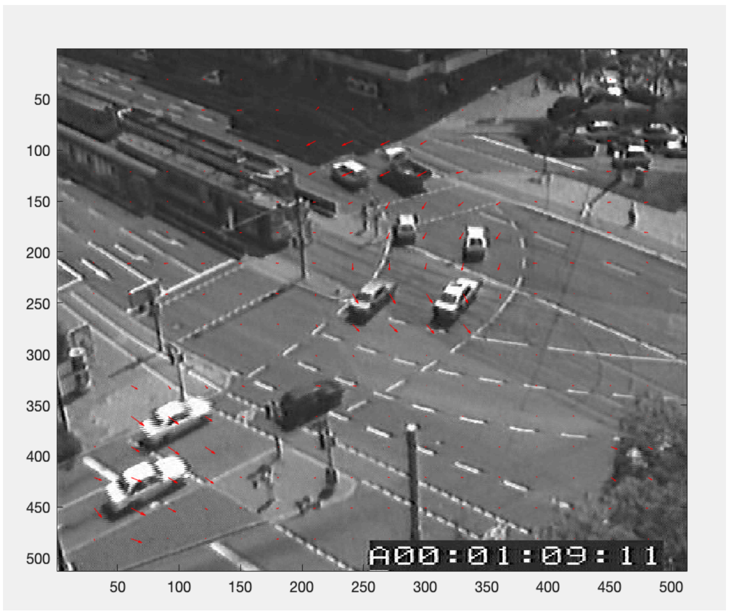

Figure 23: Optical Flow Detection on Traffic with k = 30 with Box Filter

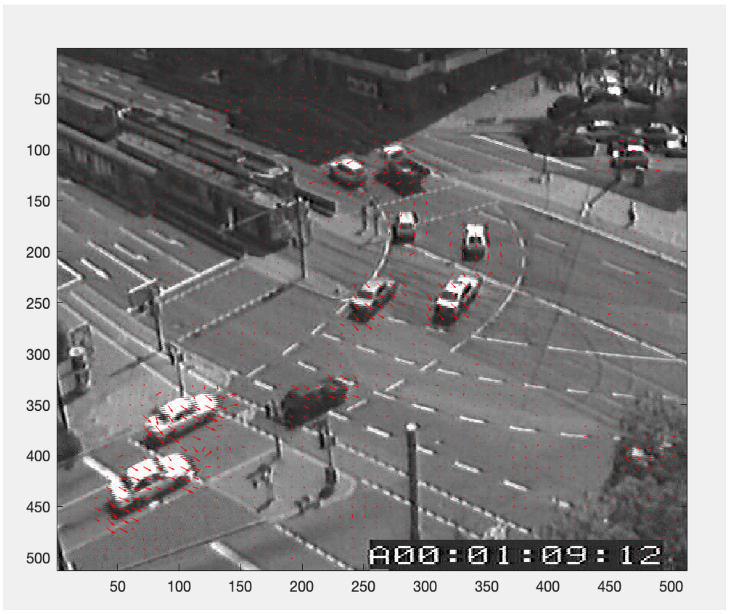

Figure 24: Optical Flow Detection on Traffic with k = 10 with Gaussian Filter

Finally, demonstrating a difference of threshold change might be useful also, though changing threshold does not give always the same result. For this reason, Figure 25 demonstrates k = 10 with Threshold = 100 and Figure 26 demonstrates k = 10 with Threshold = 10000000. Traffic image is used for the depiction and image frames are same.Moreover, both of the images are filtered with box filter. It is clear that velocity arrows are much less in shaded areas of Figure 25, since it has much higher threshold.

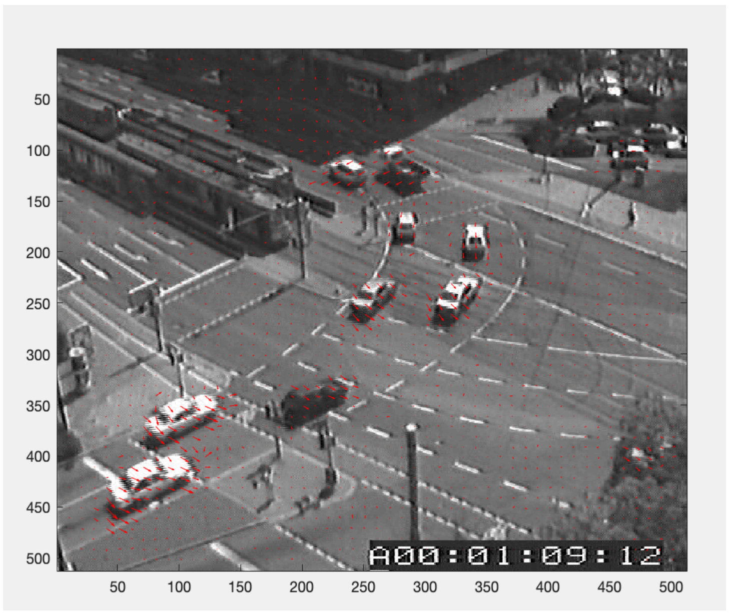

Figure 25: Optical Flow Detection on Traffic with k = 10 with Box Filter, Threshold = 10

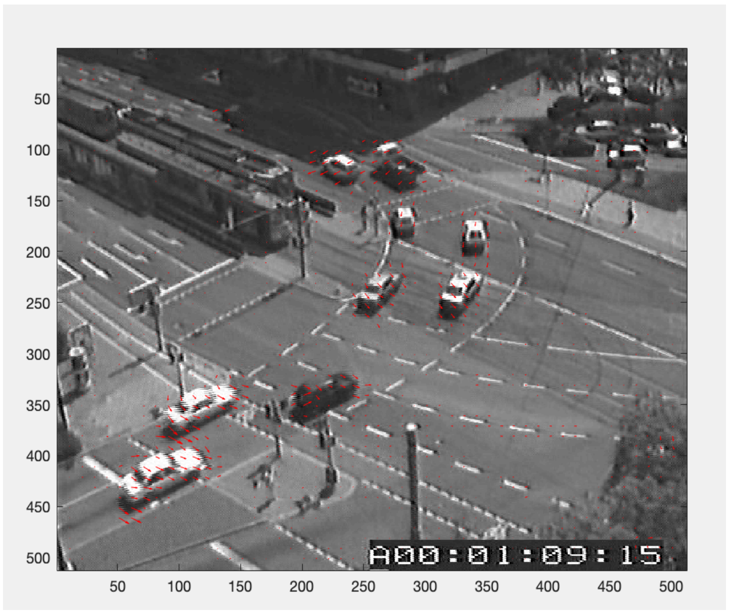

Figure 26: Optical Flow Detection on Traffic with k = 10 with Box Filter, Threshold = 10000000

As a consequence of all the distinct paramaters, it would be reasonable to imply that image patch/window is the most significant parameter to be tuned. Variation of optical flow directly changes as k changes. In addition, different smoothing on image frames with the Box Filter and Gaussian Filter does result in frequency change in intensity values. For this reason, optical flow may be detected with slight changes. Although threshold parameter was fixed, threshold can be changed additionally. However, changes in Threshold does not alter the resulting images as the other parameters in case of our provided image sequences. Therefore, fixing threshold to a certain value ensures abstraction within the algorithm in terms of comparing other parameter changes.