EE417 Computer Vision Post-Lab Report #6
Hakan Buğra Erentuğ 23637

The task was calculate the optical flow of given sequence and represent the optical flow visually. In this report, we will discuss about how frame size and local filtering effects the accuracy of the optical flow of the image. In this task, for determining the displacements of pixes, Lucas- Kanade method has been used.

Below, you can find the six different result of three different sequence for (k=10, k=20 and k=30) of out experiment with Gaussian and local filtering.

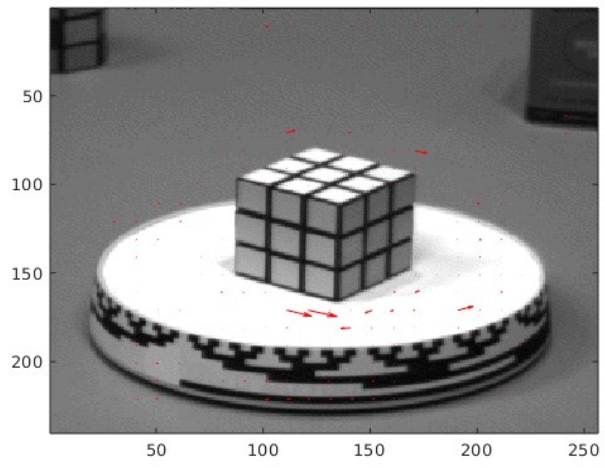


Figure1. Rubic k=10, local filtered

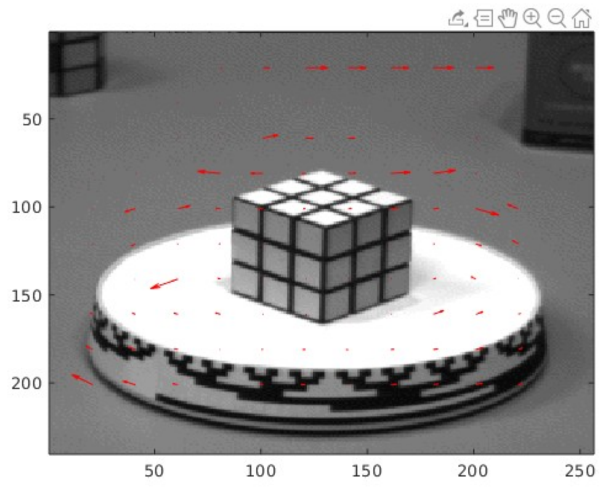


Figure2. Rubic k=20, local filtered

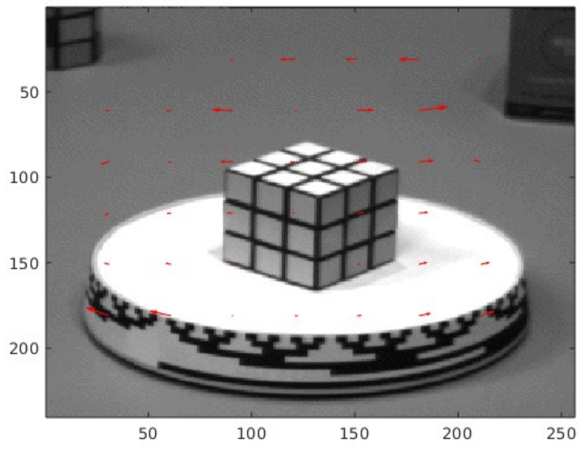


Figure3. Rubic k=30, gauss filtered

EE417 Computer Vision Post-Lab Report #6
Hakan Buğra Erentuğ 23637

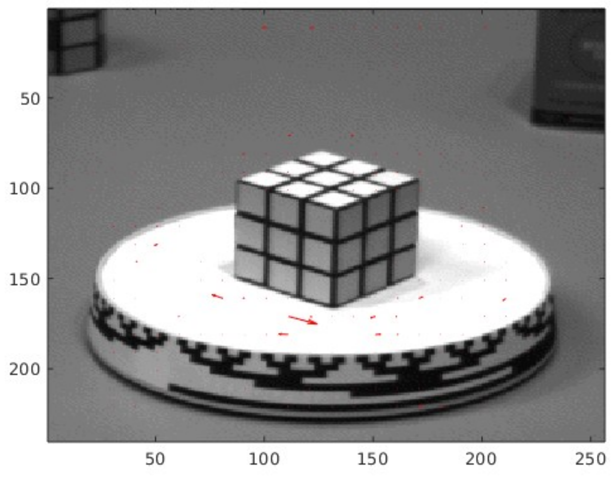


Figure4. Rubic k=10, No local filter

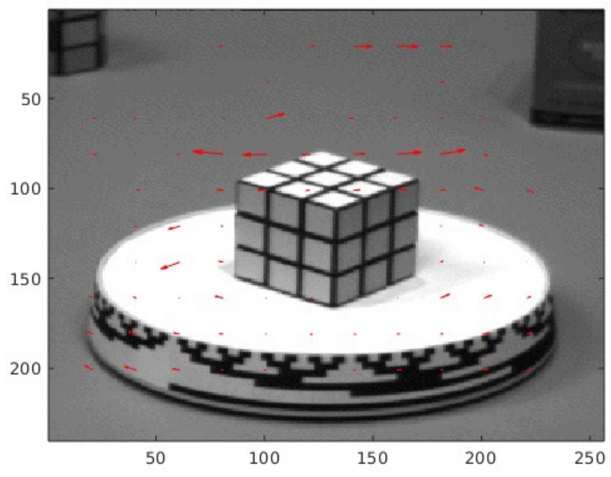


Figure5. Rubic k=20, No local filter

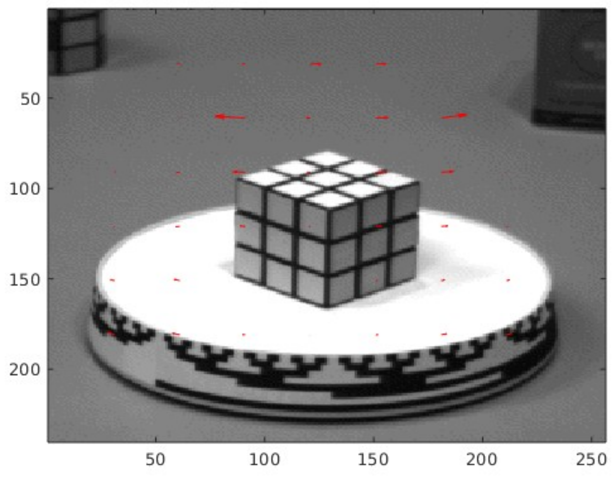


Figure6. Rubic k=30, No local filter

EE417 Computer Vision Post-Lab Report #6
Hakan Buğra Erentuğ 23637

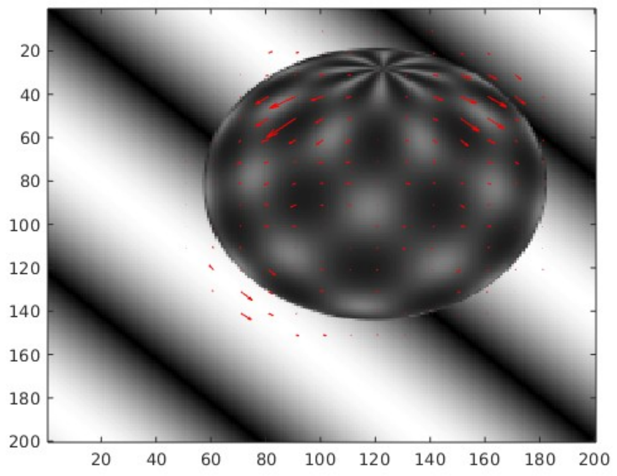


Figure7. Sphere k=10, local filtered

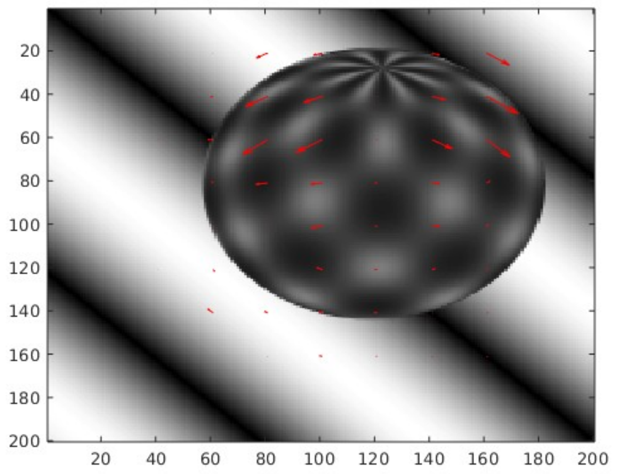


Figure8. Sphere k=20, local filtered

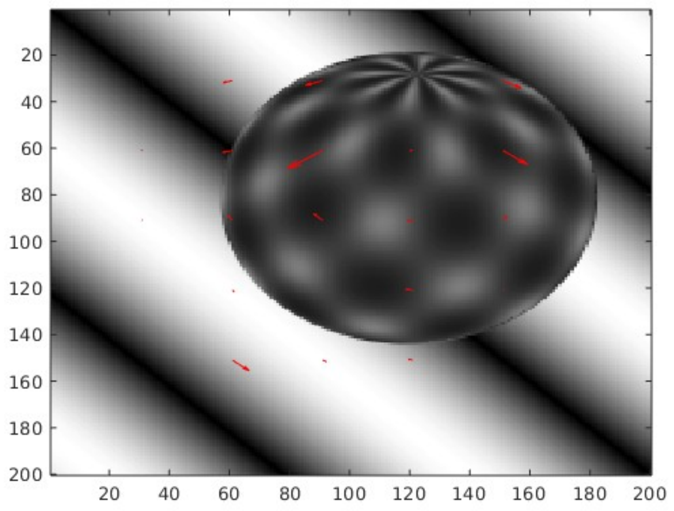


Figure9. Sphere k=30, gauss filtered

EE417 Computer Vision Post-Lab Report #6
Hakan Buğra Erentuğ 23637

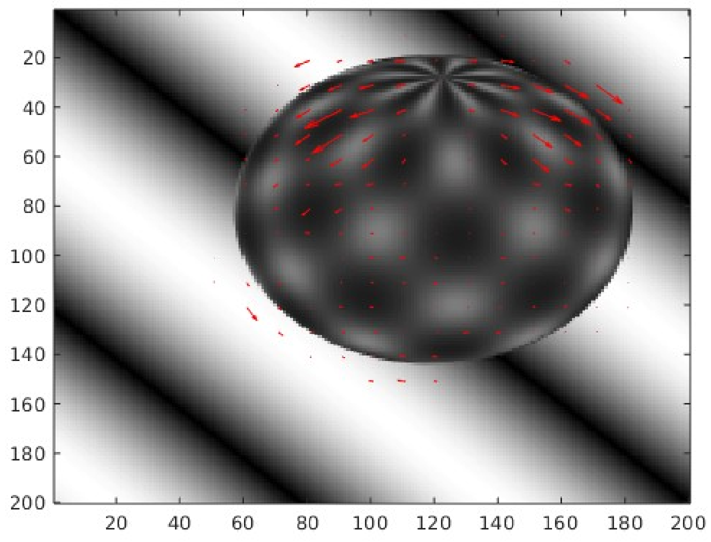


Figure10. Sphere k=10, No local filter

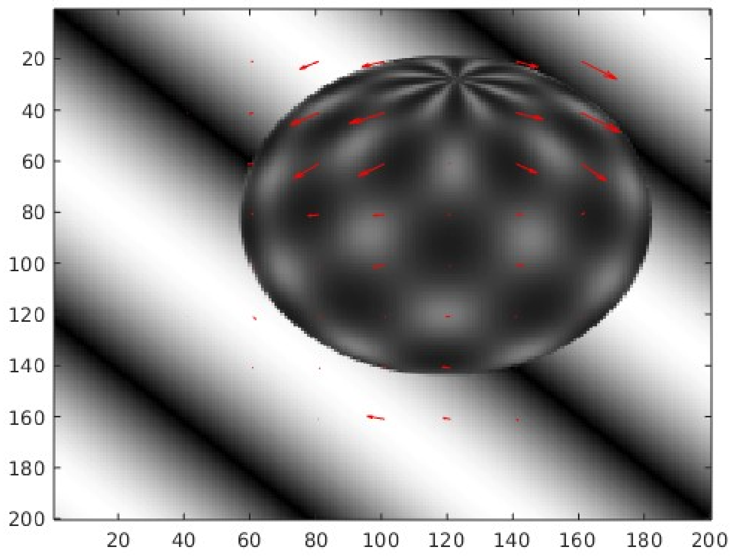


Figure11. Sphere k=20, No local filter

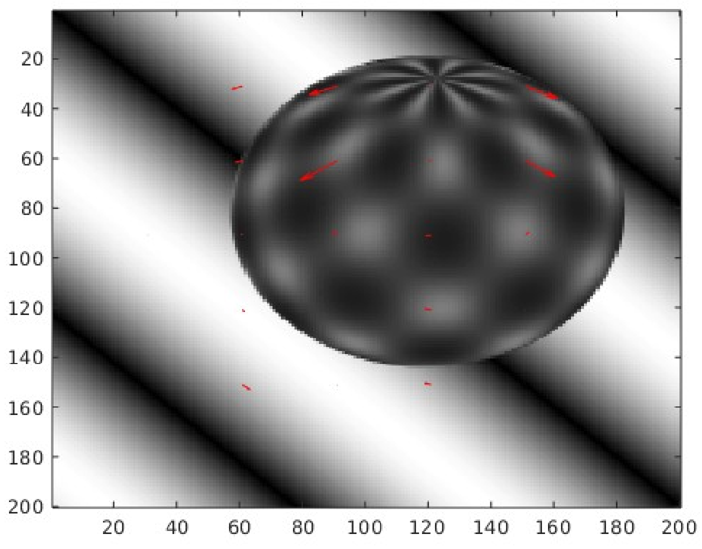


Figure12. Sphere k=30, No local filter

EE417 Computer Vision Post-Lab Report #6
Hakan Buğra Erentuğ 23637

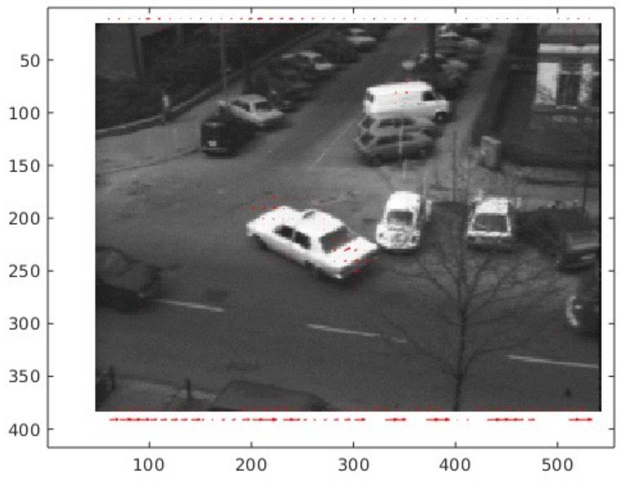


Figure11. Taxi k=10, local filtered

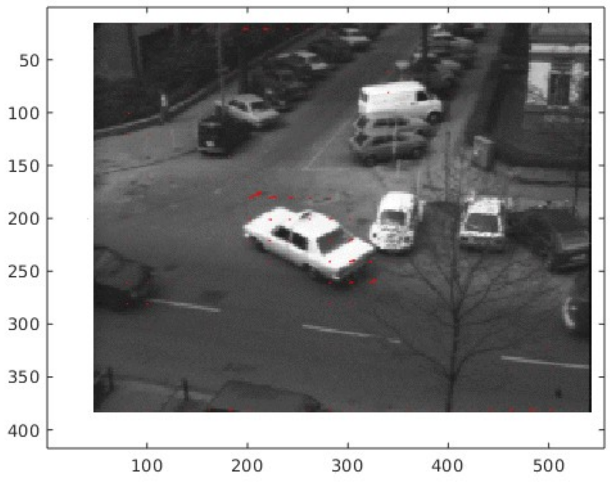


Figure11. Taxi k=20, local filtered

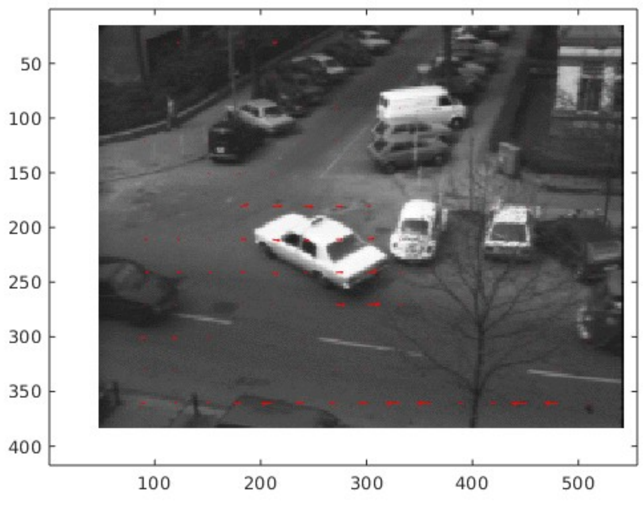


Figure12. Taxi k=30, gauss filtered

EE417 Computer Vision Post-Lab Report #6
Hakan Buğra Erentuğ 23637

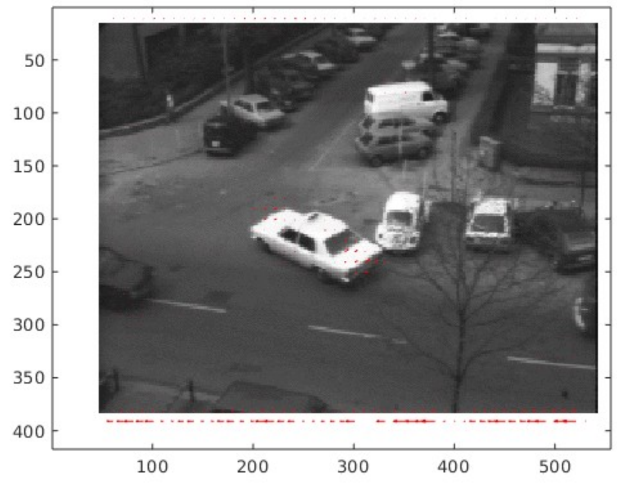


Figure13. Taxi k=10, No local filter

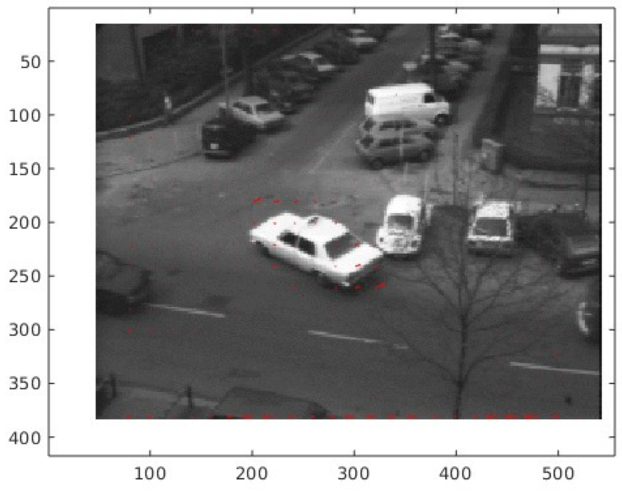


Figure14. Taxi k=20, No local filter

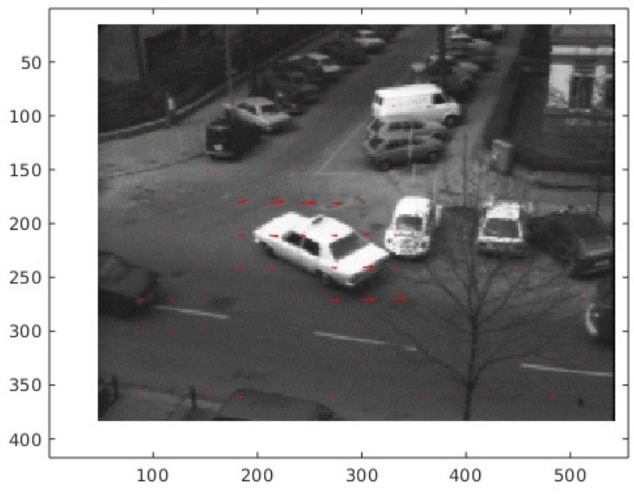


Figure15. Taxi k=30, No local filter

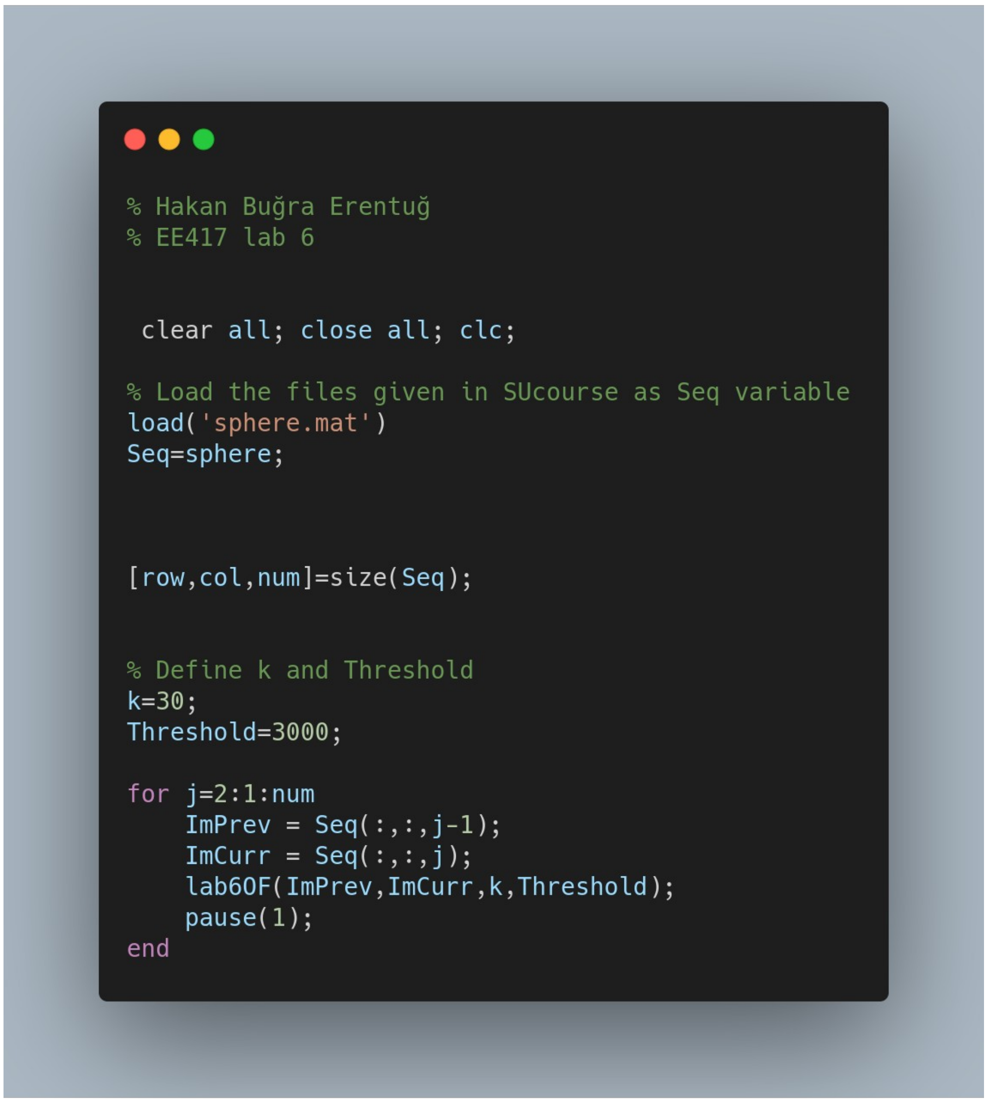
EE417 Computer Vision Post-Lab Report #6
Hakan Buğra Erentuğ 23637

Conclusion

Above, we share the results of all combination of parameters local filtering and k frame size to investigate how these metrics effects our accuracy. We can say we expected to see frequent and smaller arrows with lower k values. The main reason behind this expectation, is the working mechanism of the algorithm. K value is stands for frame size and in this algorithm we analyze the consecutive two image frame by frame. The bigger k value means less separation and more area to analyze. Therefore it makes sense to expect more arrows for k=10 value with less detailed manner. When we look at the results, we can say results support our expectations.

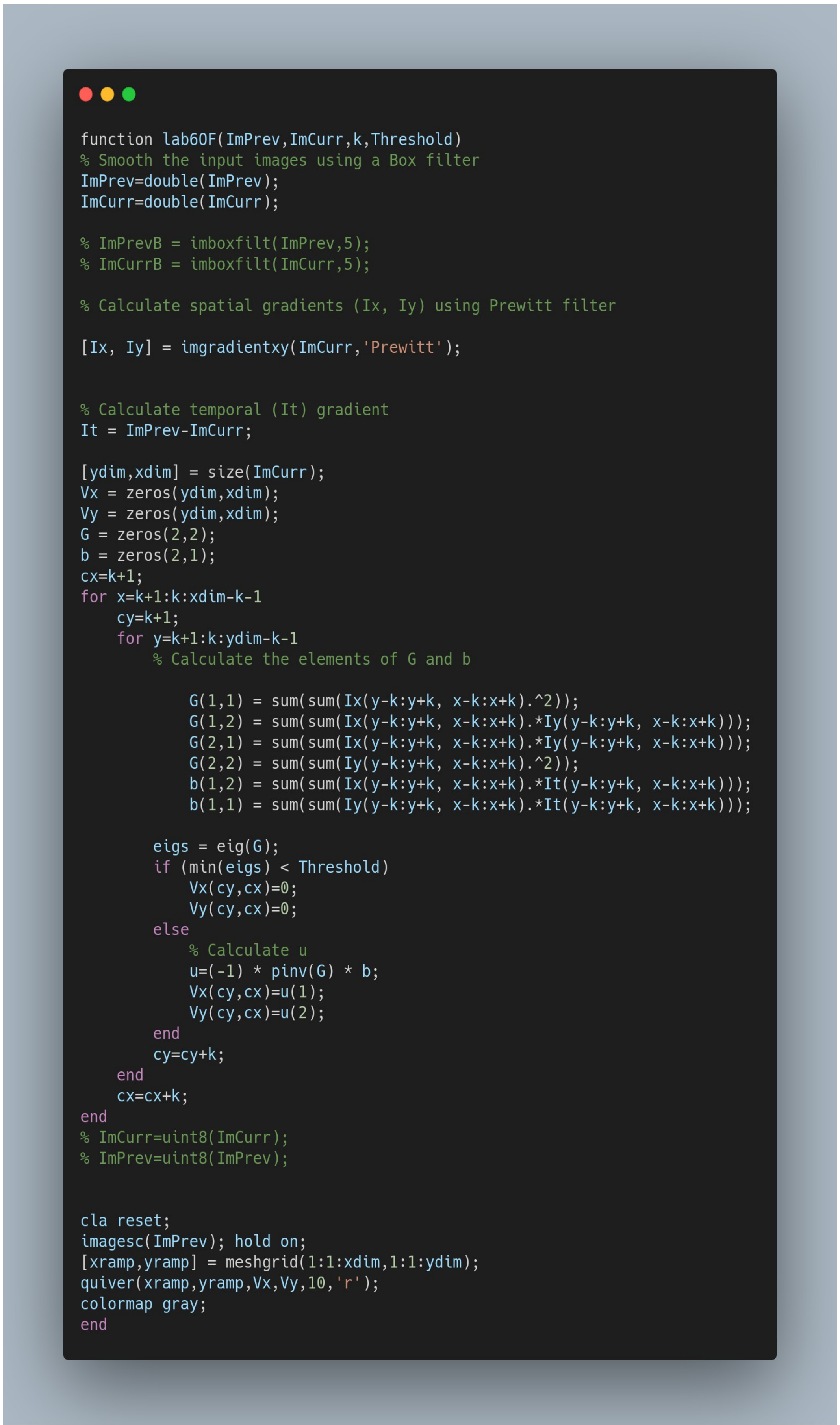
Above, we also compare the results with and without filtering. In Lucas-Kannada algoritm, two smoothing filter has been used; local box filtering and Gaussian filtering. All filtering operations is made by MATLAB's built-in functions. By looking the results, we can easily say, smoothing operation is essential to increase accuracy of optical flow. Also, we can say, gauss filtering give more proper results than box filtering. All the performance metrics is based on accuracy and frequency of displaying of arrows. Any time performance model has not been followed.

Appendices:



```
% Hakan Buğra Erentuğ
% EE417 lab 6


 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=30;
Threshold=3000;

for j=2:1:num
    ImPrev = Seq(:,:,j-1);
    ImCurr = Seq(:,:,j);
    lab6OF(ImPrev,ImCurr,k,Threshold);
    pause(1);
end
```

EE417 Computer Vision Post-Lab Report #6
Hakan Buğra Erentuğ 23637



```
function lab6OF(ImPrev,ImCurr,k,Threshold)
% Smooth the input images using a Box filter
ImPrev=double(ImPrev);
ImCurr=double(ImCurr);

% ImPrevB = imboxfilt(ImPrev,5);
% ImCurrB = imboxfilt(ImCurr,5);

% Calculate spatial gradients (Ix, Iy) using Prewitt filter

[Ix, Iy] = imgradientxy(ImCurr,'Prewitt');


% 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
        % Calculate the elements of G and b

            G(1,1) = sum(sum(Ix(y-k:y+k, x-k:x+k).^2));
            G(1,2) = sum(sum(Ix(y-k:y+k, x-k:x+k).*Iy(y-k:y+k, x-k:x+k)));
            G(2,1) = sum(sum(Ix(y-k:y+k, x-k:x+k).*Iy(y-k:y+k, x-k:x+k)));
            G(2,2) = sum(sum(Iy(y-k:y+k, x-k:x+k).^2));
            b(1,2) = sum(sum(Ix(y-k:y+k, x-k:x+k).*It(y-k:y+k, x-k:x+k)));
            b(1,1) = sum(sum(Iy(y-k:y+k, x-k:x+k).*It(y-k:y+k, x-k:x+k)));

        eigs = eig(G);
        if (min(eigs) < 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
% ImCurr=uint8(ImCurr);
% ImPrev=uint8(ImPrev);


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
```