Computer Vision I

Motion Detection Using Simple Image Filterin

Taoran Liu and Jianpeng Chen

Department of EECE, Northeastern University

February 21, 2023

Abstract

This project aims to detect motion in image sequences captured with a stationary camera using simple image filtering techniques. We will implement a program that reads in grayscale image frames, applies a temporal derivative filter, and creates a binary mask of the moving objects. We will explore different filters and design a strategy to select appropriate thresholds. The report will describe the algorithms, experiments, parameter values, and conclusions. The project will enhance our understanding of image filtering techniques and their applications in surveillance systems and traffic monitoring.

Keywords
description of algorithms, experiments, values of parameters used, observations and conclusions.

1 Introduction

The objective of this project is to explore motion detection techniques using simple image filtering algorithms. The project is particularly focused on detecting moving objects in image sequences captured with a stationary camera, where most pixels belong to a stationary background and relatively small moving objects pass in front of the camera. In this case, the intensity values observed at a pixel over time is a constant or slowly varying signal, except when a moving object begins to pass through that pixel, in which case the intensity of the background is replaced by the intensity of the foreground object. Thus, we can detect a moving object by looking at large gradients in the temporal evolution of the pixel values.

The project involves implementing a program that reads in a sequence of image frames, converts them to grayscale, and applies a one-dimensional differential operator to each pixel to compute the temporal derivative. The absolute values of the derivatives are then thresholded to create a binary mask of the moving objects, which is combined with the original frame to display the results. We will also explore various temporal derivative filters and spatial smoothing filters, and design a strategy to select appropriate thresholds.

The project report will describe the algorithms used, experiments conducted, parameter values chosen, observations made, and conclusions drawn. We will also include representative images and outputs showing both good and bad results. The project will help us gain a better understanding of manipulating images and image filtering techniques, and how they can be applied in real-world scenarios such as surveillance systems and traffic monitoring.

2 Project Content

2.1 Use temporal derivative filter directly

We first read in a sequence of image frames and make them grayscale. And for a image to be applied with a 1-D differential operator at each pixel to compute a temporal derivative, we must have enough number of pictures. For example, for a simple 0.5[-1, 0, 1] filter, we need three images to this calculation: the image itself, the previous frame and the next frame. After do this calculation to every point in this image, we can get the result like figure 1.

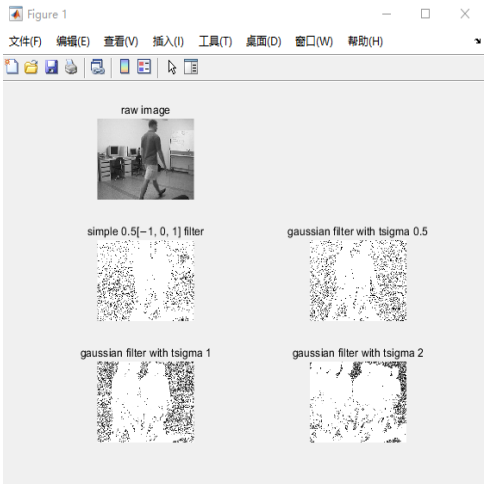

Figure 1: simple 0.5[-1, 0, 1] filter and Gaussian filter with different tsigma value.

As shown in the figure, after using a simple filter, the motion picture of two people is detected. This is because the person in the previous frame has moved to the next position, but the person in the next frame is still in the previous position, which causes the human movement in the frame, with little movement in the background. However, the noise in the picture still had a considerable impact on the result. Look at the results of the Gaussian filter with different tsigma values. When the tsigma value is smaller, the filter size is smaller, and the number of frames that affect the result is also smaller, which leads to when tsigma is equal to 0.5, the result is almost the same as the simple 0.5[−1, 0, 1] filter. When the tsigma value becomes larger, the length of the filter is also increased, and more front and rear frames are calculated, resulting in longer and longer human movement detection results. At the same time, the noise visible to the naked eye is reduced. In the case of considering more frames, the noise is also filtered to a certain extent in disguise.After our analysis, we can see that when tsigma is equal to 1, there will be better results

2.2 Do smoothing before using temporal derivative filter

Try applying a 2D spatial smoothing filter to the frames before applying the temporal derivative filter.We first try box 3*3 filter and 5*5 box filter and the result is like figure 2.

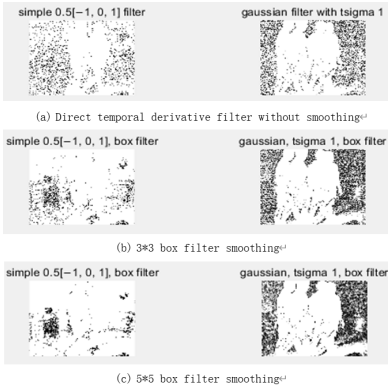

Figure 2: 3*3 box filter and 5*5 box filter

After using the box filter to process the image, apply a temporal derivative filter to the image. It can be seen that Part of the noise is removed. Among them, the noise removal effect of 5*5 is better. Finally, we tried to smooth the image with a Gaussian filter first, and then apply a temporal derivative filter to the image. The result is like figure 3.

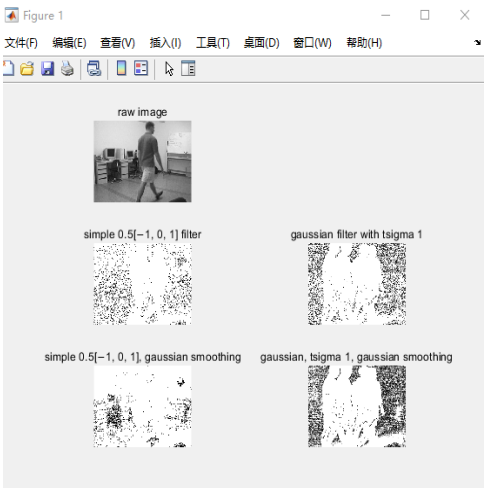

Figure 3: temporal derivative filter after Gaussian filter smoothing

We chose a 2D Gaussian filter with ssigma of 1. Compared with the previous box filter, the Gaussian filter has a more complete noise removal effect, and it will not lose much in some detection details.

2.3 The result after adding the threshold and the method of adding the threshold automatically

Finally, we add a threshold to all the previous results, and combine the mask with the original frame to display the results as figure 4.

Figure 4: The result of the previous result when the threshold is 5

It is clear that all the results filter out most of the background noise, and all have obvious human motion detection results. In order to find the most suitable threshold, we used the same other parameters (Gaussian filter smoothing ssigma=1, Gaussian difference tsigma=1), and different thresholds to calculate the results. The results are shown in Figure 5.

Figure 5: The results of motion detection at different thresholds

It can be seen from the results in the figure that when the threshold is 5, the result is optimal. However, when we discuss the actual situation, we hope that there can be a method to automatically determine the appropriate threshold, so that the noise can be removed well for different picture situations. We have two different realities. Situation 1, when people or objects move obviously in the picture, the difference result of the overall picture will be larger, and it is easy to produce a larger result. In the second case, there is no obvious movement of people or objects in the picture, and the overall difference result of the picture is small. We must discuss these two situations separately. Among them, the simplest division method is to use the mean value of the picture to distinguish. Sum all the difference results calculated on the screen and divide by the number of pixels in the whole screen to get an average value of the whole screen. Through experiments and figure 6 and figure 7, we obtained a mean value of 3 that is more suitable for this data set scenario.

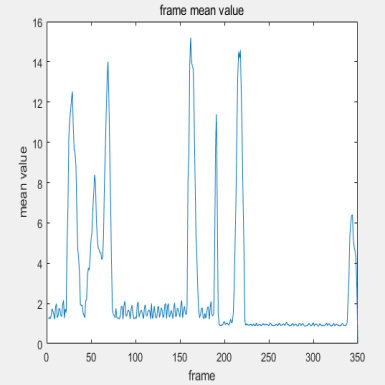

Figure 6: The mean value of the difference result of each frame picture

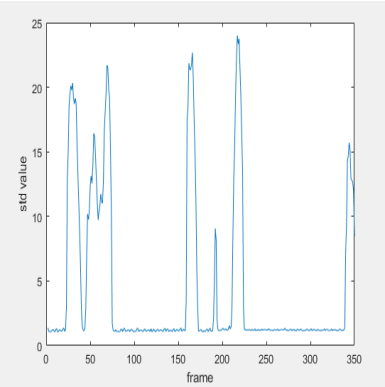

Figure 7: The standard deviation of the difference results for each frame of pictures

We also calculated the standard deviation of all frames of the differential results, and the results are shown in Figure 3-3. For the first case (with obvious moving objects), we can simply set the threshold equal to its standard deviation (because there are obvious moving objects in the picture, its standard deviation will be relatively larger than the value of background noise, and is also smaller than the boundary value of the

moving object), and the movement of the object can be well recognized.The result is shown in Figure 8.

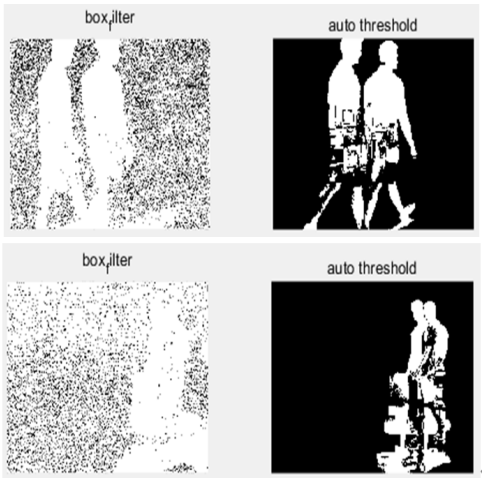

Figure 8: Automatically add threshold results under the first case

In the second case (no significant object movement), we need to amplify its standard deviation so that the threshold is large enough to filter out background noise. In this scenario, we set its value to ten times the standard deviation.The result is shown in Figure 9

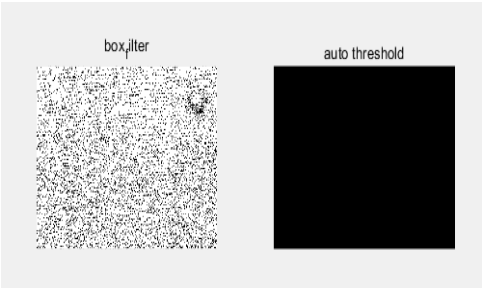

Figure 9: Automatically add threshold results under the second case

3 Conclusions

In this project, we explored motion detection techniques using simple image filtering algorithms. Our program implemented a one-dimensional differential operator to compute the temporal derivative of grayscale image frames and generated a binary mask of the moving objects. We also explored different temporal derivative filters and spatial smoothing filters and designed a strategy to select appropriate thresholds.

Our experiments showed that a 0.5[-1,0,1] filter performed well for detecting moving objects in the image sequences we tested. Applying a 2D Gaussian filter with a standard deviation of 2 pixels before the temporal derivative filter also helped reduce noise in the binary mask. Selecting a threshold based on the standard deviation of the temporal gradients for background pixels was an effective way to create an accurate binary mask.

Overall, this project provided a valuable opportunity to learn about image filtering techniques and their applications in motion detection. These techniques have important practical applications in surveillance systems and traffic monitoring. With further refinement and optimization, our program could potentially be applied to real-world scenarios to detect and track moving objects.

A Appendices

```
%% parameter
clc;
clear; % clear all
tsigma=0.5; %sigma of time temporal
    derivative gaussian filter
Threshold=5; %Threshold of 1D time
    temporal derivative filter [-1 0
    1]

SmoothfilterSize=3; % size of smooth
     filter
ssigma=1; % sigma of gaussian
    smoothing filter

Swich= 'd';
% a: do temporal derivative with
    origin image set
% b: using box filter smooth the
    image and then do temporal
    derivative
% c: using gaussian filter smooth
    the image and then do temporal
    derivative
% d: try different threshold
%% load image
n=354;
I=cell(1,n+1);
for i=2:n
imageName=strcat(['RedChair/RedChair
    /advbgst1_21_',sprintf('%04i',i)
    ],'.jpg');
I{i} = rgb2gray(imread(imageName));
end
% boundry padding
I{1}= rgb2gray(imread(strcat('
    RedChair/RedChair/
    advbgst1_21_0002','.jpg')));
I{355}= rgb2gray(imread(strcat('
    RedChair/RedChair/
    advbgst1_21_0354','.jpg')));
```

```
%% main process

m=double(uint8(5*tsigma))+rem(double
    (uint8(5*tsigma))+1,2);
r=uint16(m/2)-rem(m,2);

[DevImageBox,DevImageGaussian]=
    tfilterfunction(I,tsigma,
    Threshold); % temporal derivative
[DevImageBox1,DevImageGaussian1]=
    tfilterfunction(I,2*tsigma,
    Threshold);
[DevImageBox2,DevImageGaussian2]=
    tfilterfunction(I,4*tsigma,
    Threshold);
imagewithRestult_DevImageBox=
    add_result(I, DevImageBox); % add
     result
imagewithRestult_DevImageBox1 =
    add_result(I, DevImageBox1);
imagewithRestult_DevImageBox2 =
    add_result(I, DevImageBox2);
imagewithRestult_DevImageGaussian =
    add_result(I, DevImageGaussian);



imagewithRestult_DevImageGaussian1 =
     add_result(I, DevImageGaussian1)
    ;
imagewithRestult_DevImageGaussian2 =
     add_result(I, DevImageGaussian2)
    ;

SmoothImage1 = smooth_box(I,
    SmoothfilterSize);
[DevSmoothImage_Box1,
    DevSmoothImage_Gaussian1]=
    tfilterfunction(SmoothImage1,2*
    tsigma,Threshold); % temporal
    derivative again
imagewithRestult_box1= add_result(I,
     DevSmoothImage_Box1);
imagewithRestult_Gaussian1=
    add_result(I,
    DevSmoothImage_Gaussian1);

SmoothImage2 = smooth(I,
    SmoothfilterSize,ssigma); %
    gaussian ssigma = 1
[DevSmoothImage_Box2,
    DevSmoothImage_Gaussian2]=
    tfilterfunction(SmoothImage2,2*
    tsigma,Threshold); % threshold =
    5
imagewithRestult_Box2= add_result(I,
     DevSmoothImage_Box2);
imagewithRestult_Gaussian2=
    add_result(I,
    DevSmoothImage_Gaussian2);

[DevSmoothImage_Box3,
    DevSmoothImage_Gaussian3]=
    tfilterfunction(SmoothImage2,2*
    tsigma,0.2*Threshold); %
    threshold = 1
imagewithRestult_Gaussian3=
    add_result(I,
    DevSmoothImage_Gaussian3);

[DevSmoothImage_Box4,
    DevSmoothImage_Gaussian4]=
    tfilterfunction(SmoothImage2,2*
    tsigma,5*Threshold); % threshold
    = 25
imagewithRestult_Gaussian4=
    add_result(I,
    DevSmoothImage_Gaussian4);

[DevSmoothImage_Box5,
    DevSmoothImage_Gaussian5]=
    tfilterfunction(SmoothImage2,2*
    tsigma,0); % no threshold
imagewithRestult_Gaussian5=
    add_result(I,
    DevSmoothImage_Gaussian5);

%% a plot
if Swich=='a'
for i= 2:350
gg1 = figure(1);
set(gg1, 'Position',
    [100,100,1600,1200], 'Color', '
    white');

j=1;
subplot(3,4,j);
imshow(I{i});
title('raw␣image');
j=j+4;
subplot(3,4,j);
imshow(DevImageBox{i});
title('simple␣0.5[-1,␣0,␣1]␣filter')
    ;
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_DevImageBox{
    i});



title('ADD␣Result');
j=j+1;

subplot(3,4,j);
imshow(DevImageGaussian{i});
title('gaussian␣filter␣with␣tsigma␣
    0.5');
j=j+1;
subplot(3,4,j);
imshow(
    imagewithRestult_DevImageGaussian
    {i});
title('ADD␣Result');
j=j+1;
```

```
subplot(3,4,j);
imshow(DevImageGaussian1{i});
title('gaussian filter with tsigma 1
    ');
j=j+1;
subplot(3,4,j);
imshow(
    imagewithRestult_DevImageGaussian1
    {i});
title('ADD Result');
j=j+1;

subplot(3,4,j);
imshow(DevImageGaussian2{i});
title('gaussian filter with tsigma 2
    ');
j=j+1;
subplot(3,4,j);
imshow(
    imagewithRestult_DevImageGaussian2
    {i});
title('ADD Result');
j=j+1;

pause(1);
end
end
%% b box filter 3*3 and 5*5
if Swich=='b'
for i= 2:350
gg1 = figure(1);
set(gg1, 'Position',
    [100,100,1600,1200], 'Color', '
    white');

j=1;
subplot(3,4,j);
imshow(I{i});
title('raw image');
j=j+4;

subplot(3,4,j);
imshow(DevImageBox{i});
title('simple 0.5[-1, 0, 1] filter')
    ;
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_DevImageBox{
    i});
title('ADD Result');
j=j+1;

subplot(3,4,j);
imshow(DevImageGaussian1{i});
title('gaussian filter with tsigma 1
    ');
j=j+1;
subplot(3,4,j);
imshow(
    imagewithRestult_DevImageGaussian1
    {i});
title('ADD Result');
j=j+1;

subplot(3,4,j);
imshow(DevSmoothImage_Box1{i});
title('simple 0.5[-1, 0, 1], 3*3 box
    ');
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_box1{i});
title('ADD Result');
j=j+1;

subplot(3,4,j);
imshow(DevSmoothImage_Gaussian1{i});
title('gaussian, tsigma 1, 3*3 box')
    ;
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_Gaussian1{i
    });
title('ADD Result');
j=j+1;

pause(1);
end
end
%% c gaussian filter ssigma = 1
if Swich=='c'
for i= 2:350
gg1 = figure(1);

set(gg1, 'Position',
    [100,100,1600,1200], 'Color', '
    white');

j=1;
subplot(3,4,j);
imshow(I{i});
title('raw image3');
j=j+4;

subplot(3,4,j);
imshow(DevImageBox{i});
title('simple 0.5[-1, 0, 1] filter')
    ;
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_DevImageBox{
    i});
title('ADD Result');
j=j+1;

subplot(3,4,j);
imshow(DevImageGaussian1{i});
title('gaussian filter with tsigma 1
    ');
j=j+1;
subplot(3,4,j);
imshow(
    imagewithRestult_DevImageGaussian1
    {i});
title('ADD Result');
j=j+1;
```

```
subplot(3,4,j);
imshow(DevSmoothImage_Box2{i});
title('simple 0.5[-1, 0, 1], 
    gaussian smoothing');
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_Box2{i});
title('ADD Result');
j=j+1;

subplot(3,4,j);
imshow(DevSmoothImage_Gaussian2{i});
title('gaussian, tsigma 1, gaussian 
    smoothing');
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_Gaussian2{i
    });
title('ADD Result');
j=j+1;

pause(1);
end
end

%% d different threshold
if Swich == 'd'
for i= 2:350
gg1 = figure(1);
set(gg1, 'Position',
    [100,100,1600,1200], 'Color', '
    white');

j=1;
subplot(3,4,j);
imshow(I{i});
title('raw image3');
j=j+4;

subplot(3,4,j);
imshow(DevSmoothImage_Gaussian5{i});
    % no threshold
title('no threshold');
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_Gaussian5{i
    });
title('ADD Result');
j=j+1;

subplot(3,4,j);
imshow(DevSmoothImage_Gaussian3{i});
    % threshold = 1
title('threshold = 1');
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_Gaussian3{i
    });
title('ADD Result');
j=j+1;

subplot(3,4,j);
imshow(DevSmoothImage_Gaussian2{i});
    % threshold = 5
title('threshold = 5');
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_Gaussian2{i
    });
title('ADD Result');
j=j+1;

subplot(3,4,j);

imshow(DevSmoothImage_Gaussian4{i});
    % threshold = 25
title('threshold = 25');
j=j+1;
subplot(3,4,j);
imshow(imagewithRestult_Gaussian4{i
    });
title('ADD Result');
j=j+1;

pause(0.5);
end
end

%% tfilter function
function [ DevImageBox,
    DevImageGaussian] =
    tfilterfunction(I,sigma,Threshold
    )
% 1D box filter
tfilterBox = 0.5*[-1, 0, 1];
% gaussian filter
m=double(uint8(5*sigma))+rem(double(
    uint8(5*sigma))+1,2);
r=uint16(m/2)-rem(m,2);
tfilterGaussian=fspecial('gaussian'
    ,[1,m],sigma);
tfilterGaussian1 = tfilterGaussian;
r = double(r);
for j= -r:r
tfilterGaussian1(j+r+1)= -1*double
    ((-j)/sigma^2)*double(
    tfilterGaussian(j+r+1));
end
tfilterGaussian = tfilterGaussian1;
Sum=sum(abs(tfilterGaussian));
tfilterGaussian=1/Sum*
    tfilterGaussian;
% output image initicizalize
DevImageBox=cell(1,355);
DevImageGaussian=cell(1,355+m);

% padding
for i=1:355
DevImageBox{i} = zeros(240,320);
end
for i=1:355+2*r
DevImageGaussian{i} = zeros(240,320)
    ;
end
% modify the size of input image for
```

```
    gaussian filter
GaussianI=cell(1,355+m);
for i=1:r
GaussianI{i}= 0*I{i};


end
for i=r+1:355+r
GaussianI{i}= I{i-r};
end

for i=355+r:355+2*r
GaussianI{i}= 0*I{1};
end
% filting with boxfilter
for i= 2:354
DevImageBox{i} = tfilterBox(3)*
    double(I{i+1})+tfilterBox(1)*
    double(I{i-1})+tfilterBox(2)*
    double(I{i});
DevImageBox{i} =abs(DevImageBox{i})-
    Threshold;
%imshow(DevImageBox{i});
end
% r = uint8(r);
% filting with Gaussian filter
for i =1+r:354+r
for j= -r:r
DevImageGaussian{i}=
    DevImageGaussian{i}+
    tfilterGaussian(j+r+1)*double(
    GaussianI{i+j});
end
DevImageGaussian{i} =abs(
    DevImageGaussian{i})-Threshold;
%imshow(DevImageGaussian{i});
%imshow(uint8(DevImageGaussian{i}));
end

% make time frame same
for i = 2:350
DevImageGaussian{i} =
    DevImageGaussian{i+r-1};
end

end

%% smoothing function - gaussian
function [SmoothImage]= smooth(I,
    Size,ssigma)
Smooth = fspecial('gaussian',Size,
    ssigma);
for i=1:355
SmoothImage{i}=imfilter(I{i},Smooth,
    'replicate');
end
end

%% smoothing function - box
function [SmoothImage]= smooth_box(I
    ,Size)
Smooth = 1/(Size^2)*ones(Size,Size);
for i=1:355
SmoothImage{i}=imfilter(I{i},Smooth,
    'replicate');
end
end

%% add result to image
function [imagewithRestult]=
    add_result(I, result)
for i=1:355

imagewithRestult{i}=double(I{i})+
    double(result{i});
imagewithRestult{i}=uint8(
    imagewithRestult{i});
end
end
```