**Motion Detection**

Song Jiang and Mengyun Xia

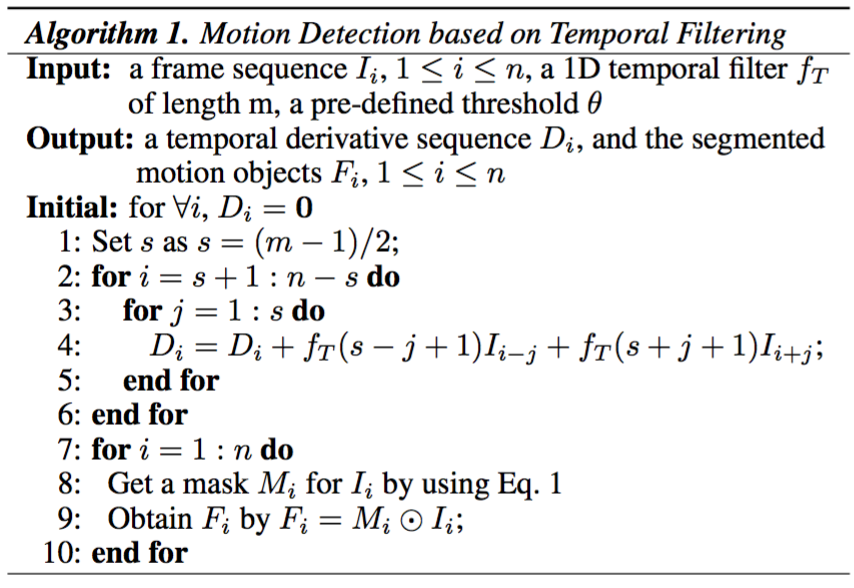
**Abstract**

In this project we will explore a simple technique for motion detection in image sequences captured with a stationary camera where most of the 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.

1. **Introduction**

Motion detection is an interesting and classical topic in the computer vision community, which is a widely-used tool in video surveillance, object tracking, and *etc*. In this paper, we propose a simple implementation for it, based on several image filtering techniques and we will discuss which one is better.

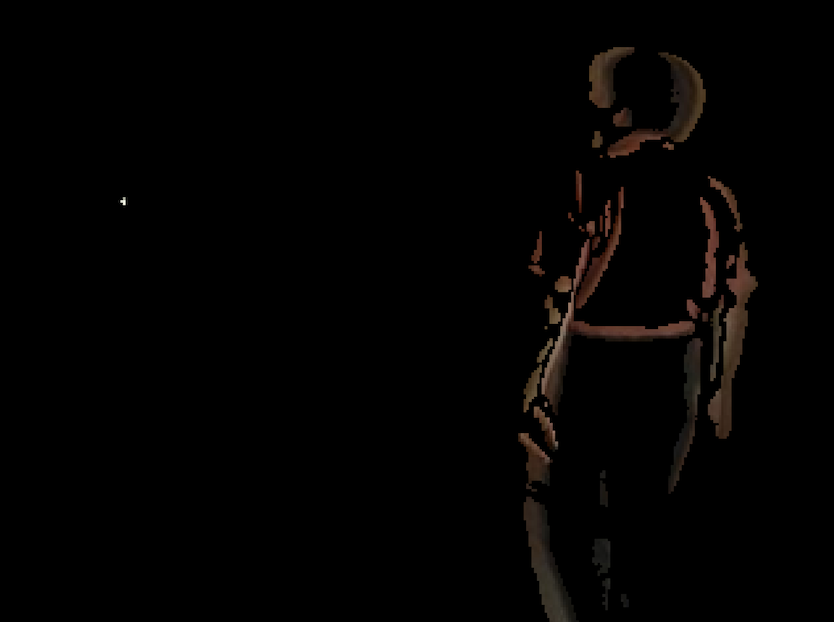
1. **Algorithm**

****

1. **Experiments**
2. Read in a sequence of image frames and make them grayscale and apply a 1-D differential operator at each pixel to compute a temporal derivative.
3. Derivative of Gaussian window size: 5\*tsigma



original image



simple 0.5[ 1, 0, 1] filter   1D derivative of Gaussian with tsigma=1



1D derivative of Gaussian with tsigma=1.8 1D derivative of Gaussian with tsigma=2.6

1. Gaussian filter window size: 5\*ssigma



original image

simple 0.5[ 1, 0, 1] filter   1D derivative of Gaussian with tsigma=1

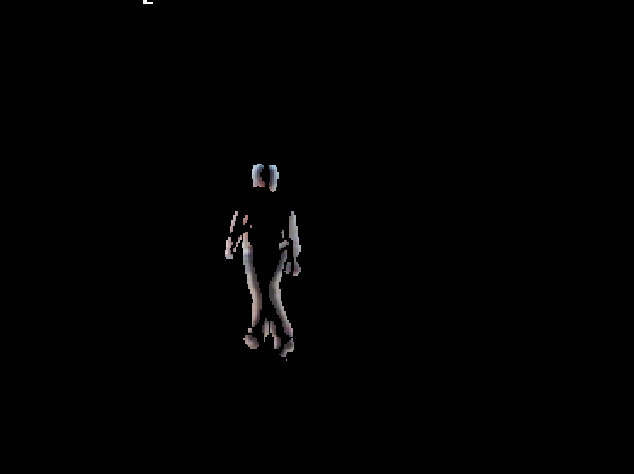
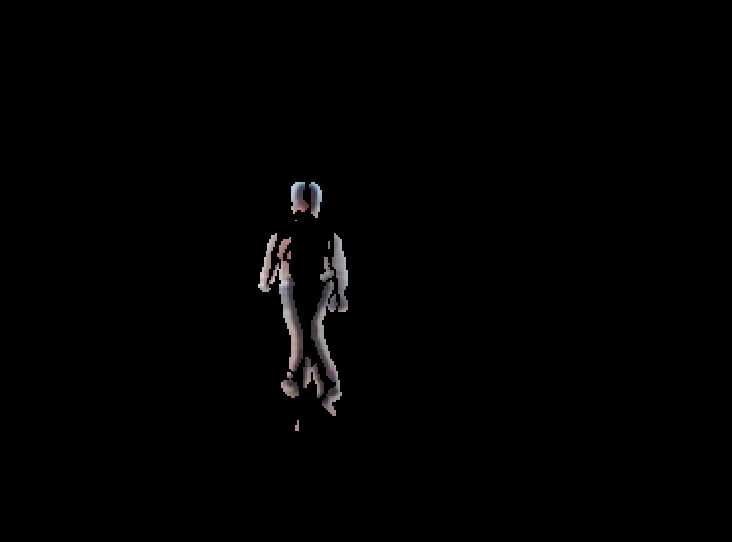
 

1D derivative of Gaussian with tsigma=1.8 1D derivative of Gaussian with tsigma=2.6

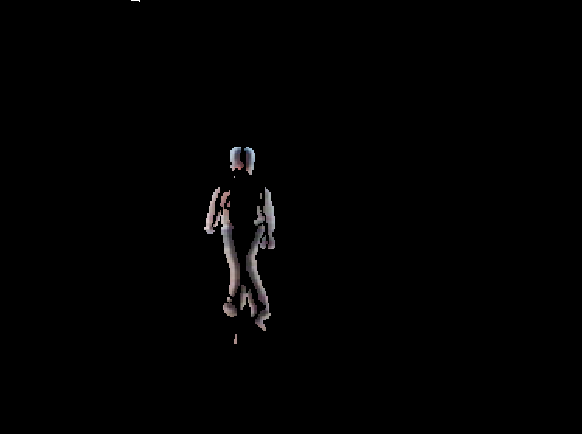
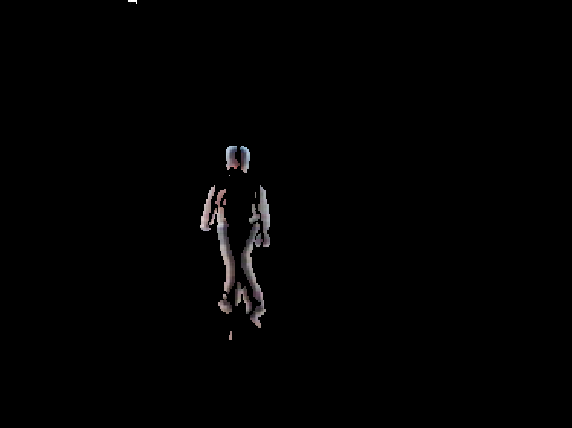
1. Threshold at every image, take the standard deviation of the Gaussian noise in every image to be the threshold.



original image

simple 0.5[ 1, 0, 1] filter   1D derivative of Gaussian with tsigma=1

1D derivative of Gaussian with tsigma=1.8 1D derivative of Gaussian with tsigma=2.6

We can see that under simple 0.5[ 1, 0, 1] filter, the boundaries are little and not complete. In Gaussian filter, the detection is better. And when tsigma equals to 1, the noise is less than tsigma equals to 1.8 or 2.6. So 1D derivative Gaussian with tsigma=1 is the best choice in this situation.

1. Applying a 2D spatial smoothing filter to the frames to reduce the noise before applying the temporal derivative filter. For the spatial smoothing filter, we try and compare 3x3, 5x5 box filters and 2D Gaussian filters with a defined standard deviation. In the report, we choose 1D derivative of Gaussian with tsigma=1 above in step (1) as reference.

(a)



3x3 box filter 5x5 box filter



2D Gaussian filter with ssigma=1 2D Gaussian filter with ssigma=1.8

(b)

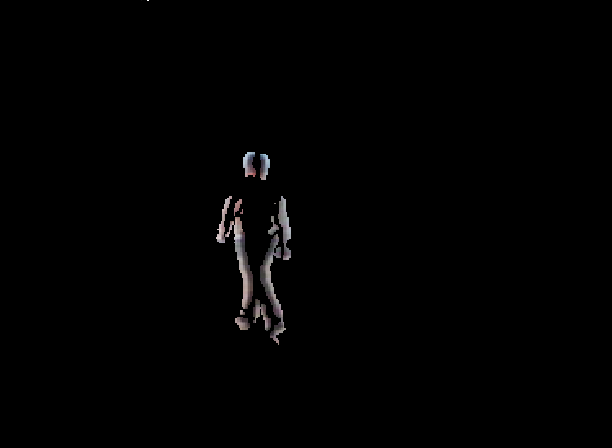


3x3 box filter 5x5 box filter

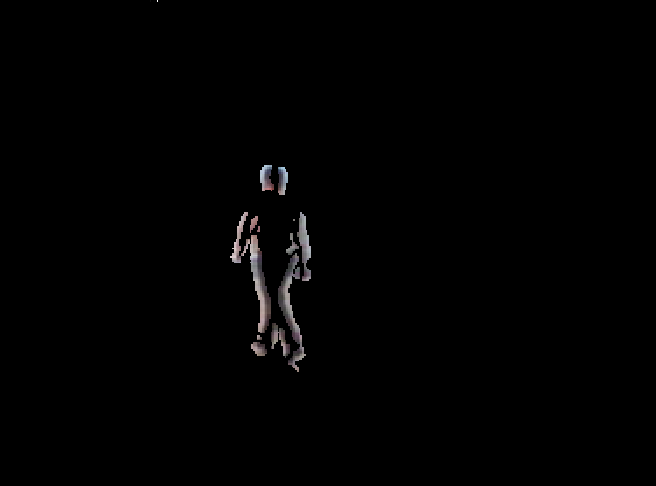


2D Gaussian filter with ssigma=1 2D Gaussian filter with ssigma=1.8

(c)

****

3x3 box filter 5x5 box filter

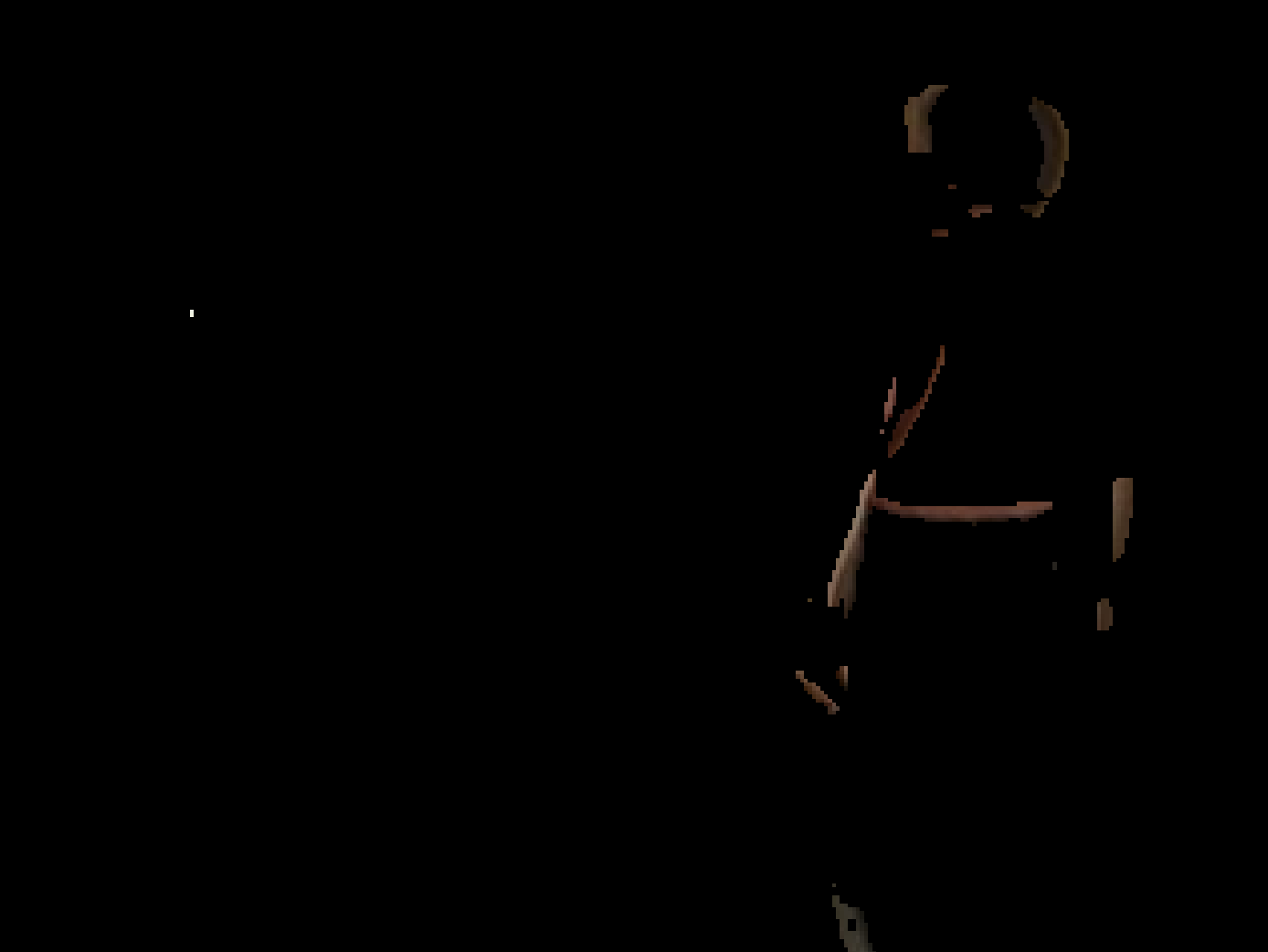
****

2D Gaussian filter with ssigma=1 2D Gaussian filter with ssigma=1.8

We can see that under 5x5 box filter or 2D Gaussian filter with ssigma=1.8, it filters too much important information and make the detection not complete. 3x3 box filter and 1D Gaussian filter with ssigma=1 is both effective to make the output better.

1. Vary the threshold to get the mask and see what works best. Design a strategy to select a good threshold for each image. Then Combine the mask with the original frame. In this report, we choose 1D derivative of Gaussian with tsigma=1 above in step (1) and 2D Gaussian filter with ssigma=1 as reference.

(a)

threshold too high threshold too low



best threshold

(b)

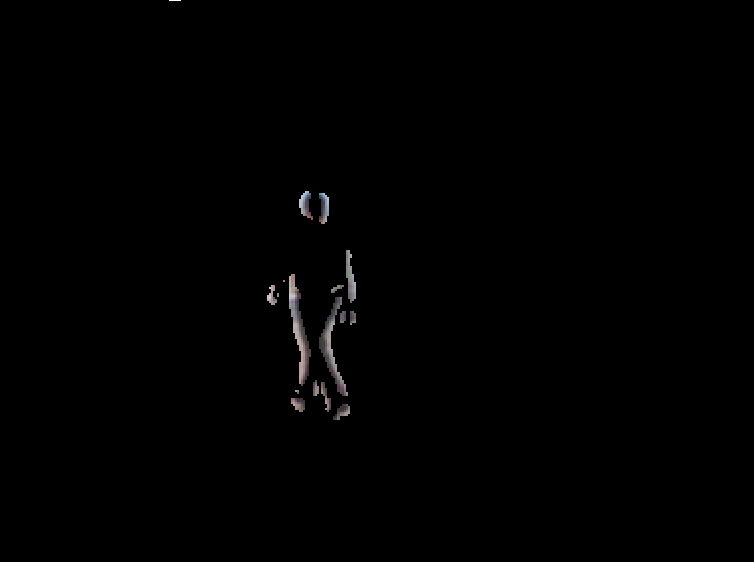


threshold too high threshold too low



best threshold

(c)

threshold too high threshold too low

****

best threshold

We can see that if the threshold is too high, then there will be more gaps. If the threshold is too low, there will be more noise. If the threshold is suitable, the detection is wonderful.

**4 Conclusion**

When doing the motion detection, we can first use a 2D Gaussian filter to erase noise, then apply a 1D derivative of Gaussian filter with the standard deviation of 1.0 or 1.4 at each pixel of a series of continuous frames to compute a temporal derivative. Choose a reasonable threshold and threshold the absolute values of the derivatives to create a 0 and 1 mask of the moving objects. Finally combine the mask with the original frame to display the results.

1. **Appendix**
2. clear;
3. close all;
4. clc;
6. %%read and store multiple image frames
7. file\_path = './EnterExitCrossingPaths2cor/';
8. img\_path\_list = dir(strcat(file\_path,'\*.jpg'));
9. %% the number of image frames
10. img\_num = length(img\_path\_list);
11. %% used to store original image frames
12. img = cell(1,img\_num);
13. %%used to store gray values of image frames
14. img\_gray = cell(1,img\_num);
15. finalImageSF=cell(1,img\_num);
16. finalImageG1=cell(1,img\_num);
17. finalImageG2=cell(1,img\_num);
18. finalImageG3=cell(1,img\_num);
19. for i=1:img\_num
20. image\_name=img\_path\_list(i).name;
21. image\_name=strcat(file\_path,image\_name);
22. img{i}=imread(image\_name);
23. finalImageSF{i}=double(img{i});
24. finalImageG1{i}=double(img{i});
25. finalImageG2{i}=double(img{i});
26. finalImageG3{i}=double(img{i});
27. %% make image frames grayscale
28. img\_gray{i}=rgb2gray(img{i});
29. end
31. h3=fspecial('average',[3,3]);
32. h5=fspecial('average',[5,5]);
33. G1=fspecial('Gaussian',[5,5],1.0);
34. G2=fspecial('Gaussian',[9,9],1.8);
35. I\_G1=cell(1,img\_num);
36. for i=1:img\_num
37. I\_G1{i}=imfilter(img\_gray{i},G1,'replicate');
38. end
40. %%the length value of the matrix correspond to one image frame
41. len=size(img{1},2);
42. %%the width value of the matrix correspond to one image frame
43. wid=size(img{1},1);
44. %%sf\_G1 is a?wid x len?matrix,
45. %%each entry is also a (1 x img\_num) matrix
46. gx1\_G1=cell(wid,len);
48. for i=1:wid
49. for j=1:len
50. for k=1:img\_num
51. gx1\_G1{i,j}(k)=I\_G1{k}(i,j);
52. end
53. end
54. end
55. sf\_G1=gx1\_G1;
56. gx2\_G1=gx1\_G1;
57. gx3\_G1=gx1\_G1;
58. %%create a simple 0.5[-1, 0, 1] filter
59. sim\_f=[-0.5,0,0.5];
60. t1=1.0; %%value of sigma t1
61. t2=1.8; %%value of sigma t2
62. t3=2.6; %%value of sigma t3
63. gx1\_mask=zeros(1,5);%range 5\*t1
64. gx2\_mask=zeros(1,9);%range 5\*t2
65. gx3\_mask=zeros(1,13);%range 5\*t3
67. %%a 1D derivative of Gaussian filter with standard devition of 1.0
68. for x=-2:2
69. gx1\_mask(x+3)=-x/(t1\*t1)\*exp(-(x\*x)/(2\*t1\*t1));
70. end
71. %%a 1D derivative of Gaussian filter with standard devition of 1.8
72. for x=-4:4
73. gx2\_mask(x+5)=-x/(t2\*t2)\*exp(-(x\*x)/(2\*t2\*t2));
74. end
75. %%a 1D derivative of Gaussian filter with standard devition of 2.6
76. for x=-6:6
77. gx3\_mask(x+7)=-x/(t3\*t3)\*exp(-(x\*x)/(2\*t3\*t3));
78. end
80. for i=1:wid
81. for j=1:len
82. %sf\_G1{i,j}=imfilter(double(sf\_G1{i,j}),sim\_f,'replicate');
83. gx1\_G1{i,j}=imfilter(double(gx1\_G1{i,j}),gx1\_mask,'replicate');
84. %gx2\_G1{i,j}=imfilter(double(gx2\_G1{i,j}),gx2\_mask,'replicate');
85. %gx3\_G1{i,j}=imfilter(double(gx3\_G1{i,j}),gx3\_mask,'replicate');
86. %gx1\_G1{i,j}=imfilter(gx1\_G1{i,j},h3,'replicate');
87. %gx1\_G1{i,j}=imfilter(gx1\_G1{i,j},h5,'replicate');
88. gx1\_G1{i,j}=imfilter(gx1\_G1{i,j},G1,'replicate');
89. %gx1\_G1{i,j}=imfilter(gx1\_G1{i,j},G2,'replicate');
90. end
91. end

94. thr\_sf\_G1=cell(1,img\_num);
95. thr\_gx1\_G1=cell(1,img\_num);
96. thr\_gx2\_G1=cell(1,img\_num);
97. thr\_gx3\_G1=cell(1,img\_num);
99. for k=1:img\_num
100. for i=1:wid
101. for j=1:len
102. %thr\_sf\_G1{k}(i,j)=sf\_G1{i,j}(k);
103. thr\_gx1\_G1{k}(i,j)=gx1\_G1{i,j}(k);
104. %thr\_gx2\_G1{k}(i,j)=gx2\_G1{i,j}(k);
105. %thr\_gx3\_G1{k}(i,j)=gx3\_G1{i,j}(k);
106. end
107. end
108. end


112. %calculate the threshold
113. thres1=cell(1,img\_num);
114. thres2=cell(1,img\_num);
115. thres3=cell(1,img\_num);
116. thres4=cell(1,img\_num);
117. for i=1:img\_num
118. %thres1{i}=(std(double(thr\_sf\_G1{i}(:)))+7)\*ones(wid,len);
119. thres2{i}=(std(double(thr\_gx1\_G1{i}(:)))+7)\*ones(wid,len);
120. %thres3{i}=(std(double(thr\_gx2\_G1{i}(:)))+7)\*ones(wid,len);
121. %thres4{i}=(std(double(thr\_gx3\_G1{i}(:)))+7)\*ones(wid,len);
122. end
124. %Convert image to binary image by thresholding
125. for i=1:img\_num
126. %thr\_sf\_G1{i}=im2bw(abs(thr\_sf\_G1{i})-thres1{i},1/255);
127. thr\_gx1\_G1{i}=im2bw(abs(thr\_gx1\_G1{i})-thres2{i},1/255);
128. %thr\_gx2\_G1{i}=im2bw(abs(thr\_gx2\_G1{i})-thres3{i},1/255);
129. %thr\_gx3\_G1{i}=im2bw(abs(thr\_gx3\_G1{i})-thres4{i},1/255);
130. %finalImageSF{i}(:,:,1) = finalImageSF{i}(:,:,1).\*double(thr\_sf\_G1{i});
131. %finalImageSF{i}(:,:,2) = finalImageSF{i}(:,:,2).\*double(thr\_sf\_G1{i});
132. %finalImageSF{i}(:,:,3) = finalImageSF{i}(:,:,3).\*double(thr\_sf\_G1{i});
133. finalImageG1{i}(:,:,1) = finalImageG1{i}(:,:,1).\*double(thr\_gx1\_G1{i});
134. finalImageG1{i}(:,:,2) = finalImageG1{i}(:,:,2).\*double(thr\_gx1\_G1{i});
135. finalImageG1{i}(:,:,3) = finalImageG1{i}(:,:,3).\*double(thr\_gx1\_G1{i});
136. %finalImageG2{i}(:,:,1) = finalImageG2{i}(:,:,1).\*double(thr\_gx2\_G1{i});
137. %finalImageG2{i}(:,:,2) = finalImageG2{i}(:,:,2).\*double(thr\_gx2\_G1{i});
138. %finalImageG2{i}(:,:,3) = finalImageG2{i}(:,:,3).\*double(thr\_gx2\_G1{i});
139. %finalImageG3{i}(:,:,1) = finalImageG3{i}(:,:,1).\*double(thr\_gx3\_G1{i});
140. %finalImageG3{i}(:,:,2) = finalImageG3{i}(:,:,2).\*double(thr\_gx3\_G1{i});
141. %finalImageG3{i}(:,:,3) = finalImageG3{i}(:,:,3).\*double(thr\_gx3\_G1{i});
142. end
144. %{
145. %display the result of images filtered by a simple[-0.5 0 0.5] filter
146. for i=1:img\_num
147. figure(),imshow(uint8(finalImageSF{i}));
148. pause(0.05);
149. end
150. %}
152. %display the result of images filtered by a 1D derivative
153. %of Gaussian filter with standard devition of 1.0
154. for i=1:img\_num
155. figure(),imshow(uint8(finalImageG1{i}));
156. pause(0.05);
157. end
159. %{
160. %display the result of images filtered by a 1D derivative
161. %of Gaussian filter with standard devition of 1.4
162. for i=1:img\_num
163. figure(),imshow(uint8(finalImageG2{i}));
164. pause(0.05);
165. end
167. %display the result of images filtered by a 1D derivative
168. %of Gaussian filter with standard devition of 1.8
169. for i=1:img\_num
170. figure(),imshow(uint8(finalImageG3{i}));
171. pause(0.05);
172. end
173. %}