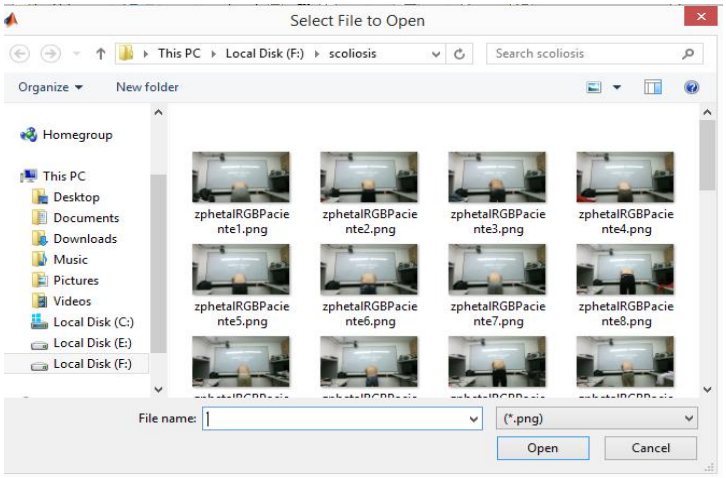
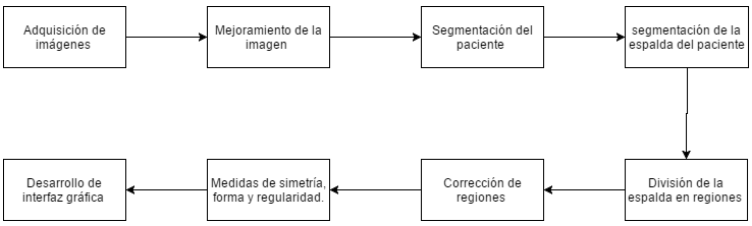
Application Description:
To run the program, first run the main_scoliosis.m file in the scoliosis folder.



In the dialog box, select the file you want.Of course the files are the same as the article files.
After reading the image, the RGB values are calculated and used for analysis.And work on the block diagram.



Get the picture– Improved image - Split the patient -Split back of patient - Redistribute to regions - Correct the area- Form measures and symmetry– Output
To eliminate image noise, the pixel values of all frames are averaged in pixels per pixel to produce a new image that results in a lower noise image and smoother changes. Then, to achieve a more accurate image where changes and textures can be better viewed, an intermediate filter is used, which results in an image in which each pixel has its own values relative to its neighbors. Takes the picture and makes no noise.
To find black and white points we increase it to a low or high. And it detects RGB points.

```
xy_R_1_b = img_raw_R<a_b;
xy_G_1_b = img_raw_G<a_b;
xy_B_1_b = img_raw_B<a_b;
xy_R_1_w = img_raw_R>a_w;
xy_G_1_w = img_raw_G>a_w;
```

```
xy_B_1_w = img_raw_B>a_w;
```

And where the difference is less, we consider it white. And this makes the image behind the patient detached.

```
xy_2 = (abs(img_raw_R - img_raw_G)< b & abs(img_raw_R - img_raw_B)<
b);
for i = 1:size(xy_3,1)-1
    for j = 1:size(xy_3,2)-1
        if xy_3(i,j)*xy_3(i+1,j)*xy_3(i,j+1) == 0
            xy_3(i,j) = 0;
        end
    end
end
```

Identifies landmarks by neighborhood.

```
for l= 1:size(Candinate,2);
      y = Candinate(1,l);
      x =  Candinate(2,l);
      tmp = X(y-1:y+1,x-1:x+1); %  consider 8 neighbor pixel
      tmp = tmp(:);
         [val, ind ] = sort(tmp,'descend');
      for n=1:N
         if (ind(n) == 5)
            cnt = cnt+1;
            PeakList(:,cnt) = [Candinate(:,l);-inv(val(n))];
         end
      end
   end
```

And filter scripts are used to soften the image

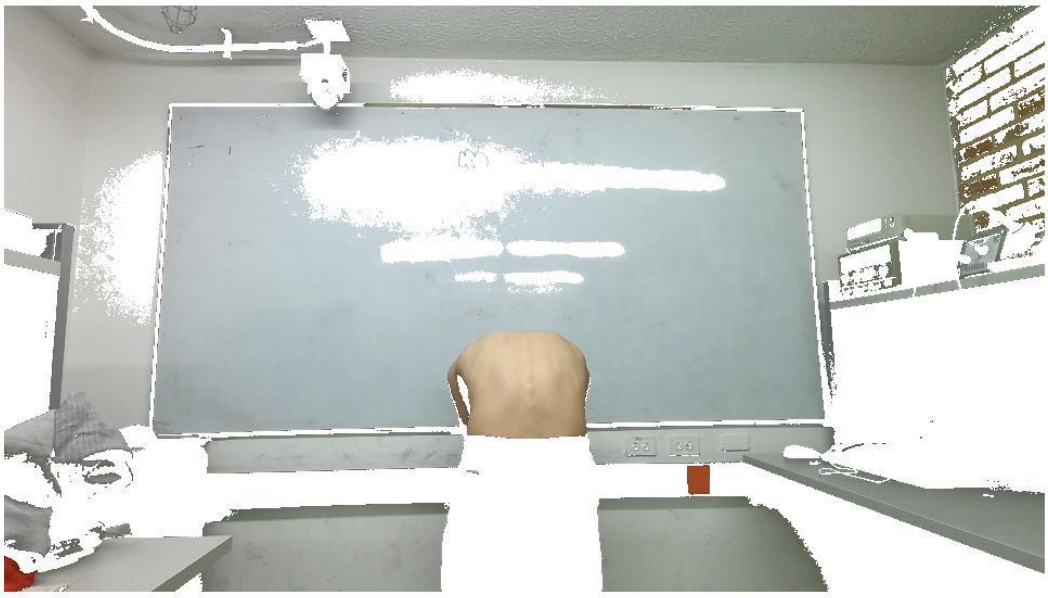
```
S =I((1+r:H-r)-bh,(1+r:W-r)-bw) + I((1+r:H-r)+bh+1,(1+r:W-r)+bw+1) -
I((1+r:H-r)-bh,(1+r:W-r)+bw+1) -I((1+r:H-r)+bh+1,(1+r:W-r)-bw);
T =P((1+r:H-r)-bh,(1+r:W-r)-bw) + P((1+r:H-r)+bh+1,(1+r:W-r)+bw+1) -
P((1+r:H-r)-bh,(1+r:W-r)+bw+1) -P((1+r:H-r)+bh+1,(1+r:W-r)-bw);
M = S./N;
Y = T./N;
St = Y - M.^2;
S1 =I((1+r:H-r)-sh+dh,(1+r:W-r)-sw+dw) + I((1+r:H-r)+sh+dh+1,(1+r:W-
r)+sw+dw+1) - I((1+r:H-r)-sh+dh,(1+r:W-r)+sw+dw+1) - I((1+r:H-
r)+sh+dh+1,(1+r:W-r)-sw+dw);
S2=S-S1;
M1=S1./N1;
M2=S2./N2;
Sb = ((N1*((M1-M).^2)) + (N2*((M2-M).^2)))/N;
MAP((1+r:H-r),(1+r:W-r)) = (Sb./St).*sign(M2-M1);
```

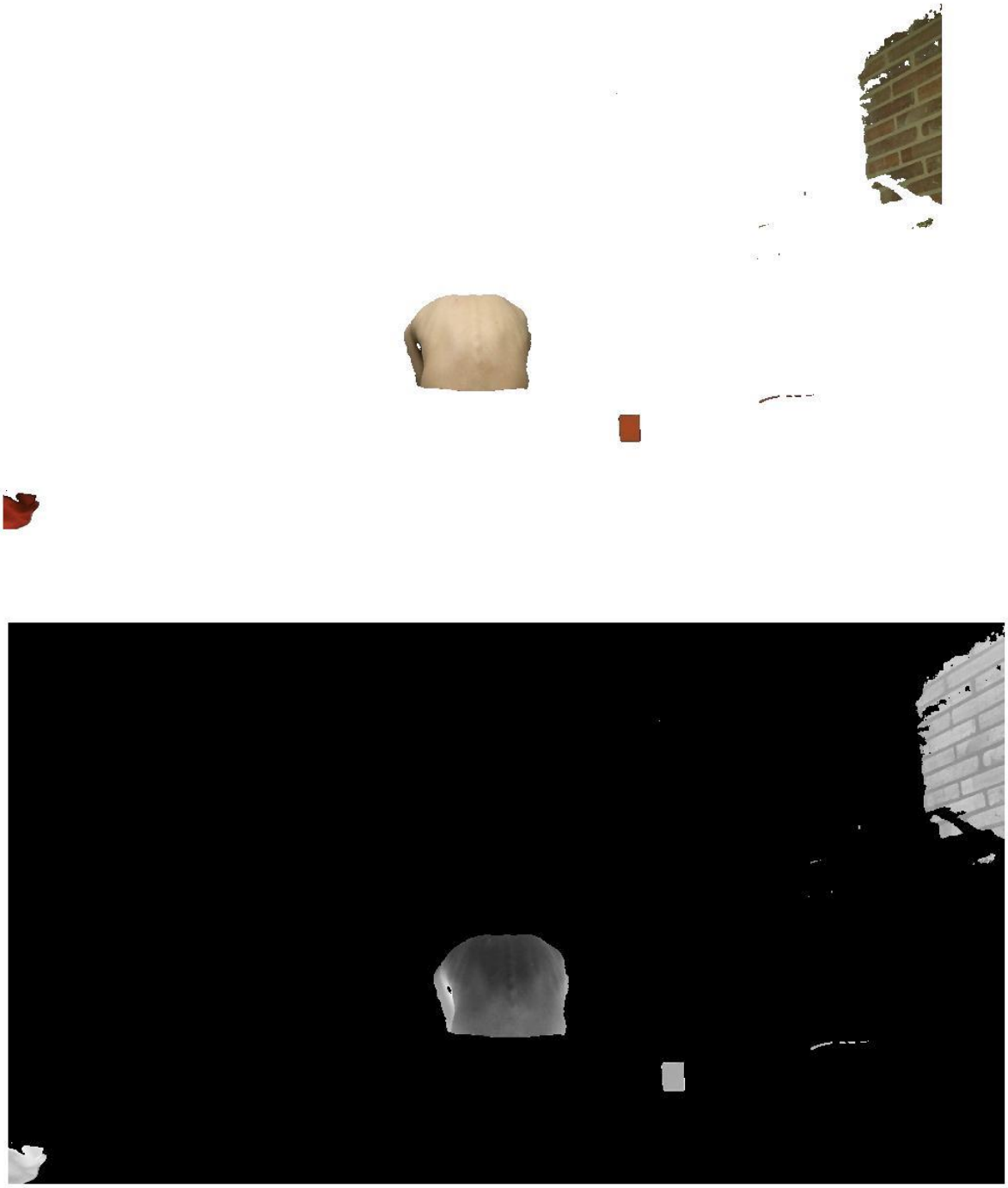
Of course, all programs and scripts are based on block diagrams.

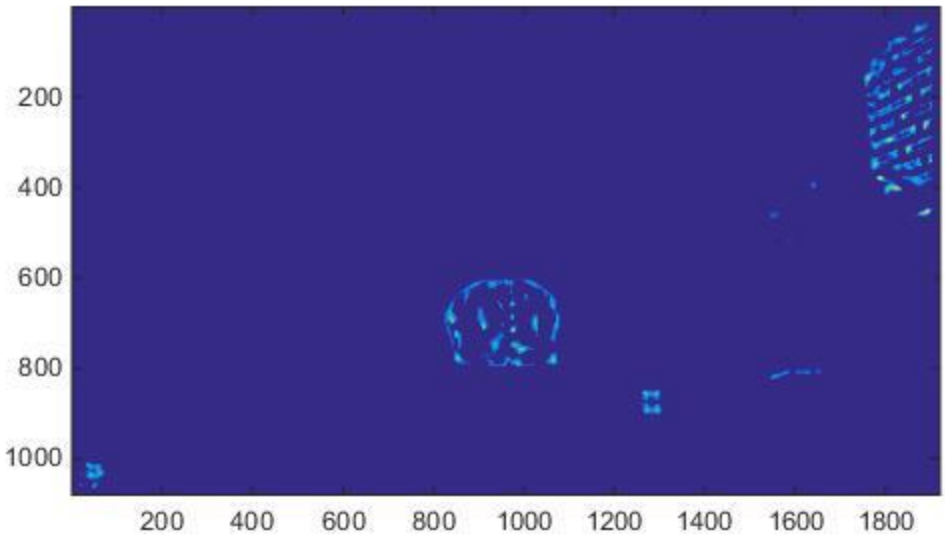
Sample implementation

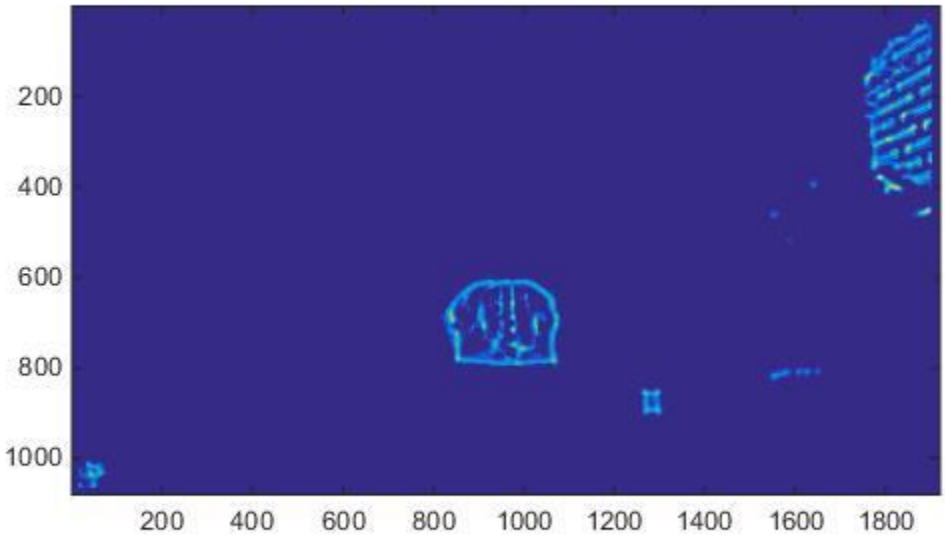
Distance error :
1.5697
Big mistake :
-2.2142
Area error :
1.6344e+03
Shoulder and angle symmetry error :
-2.2245

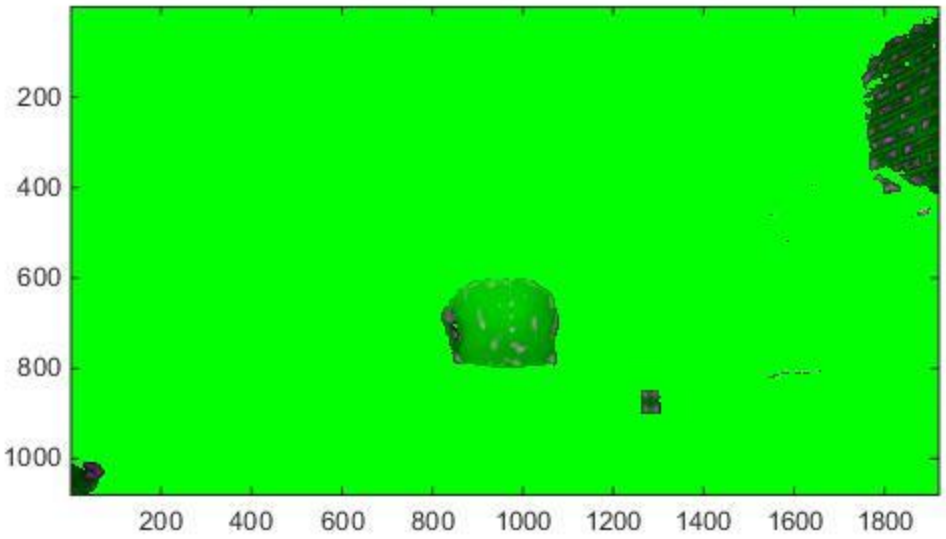
Original Data

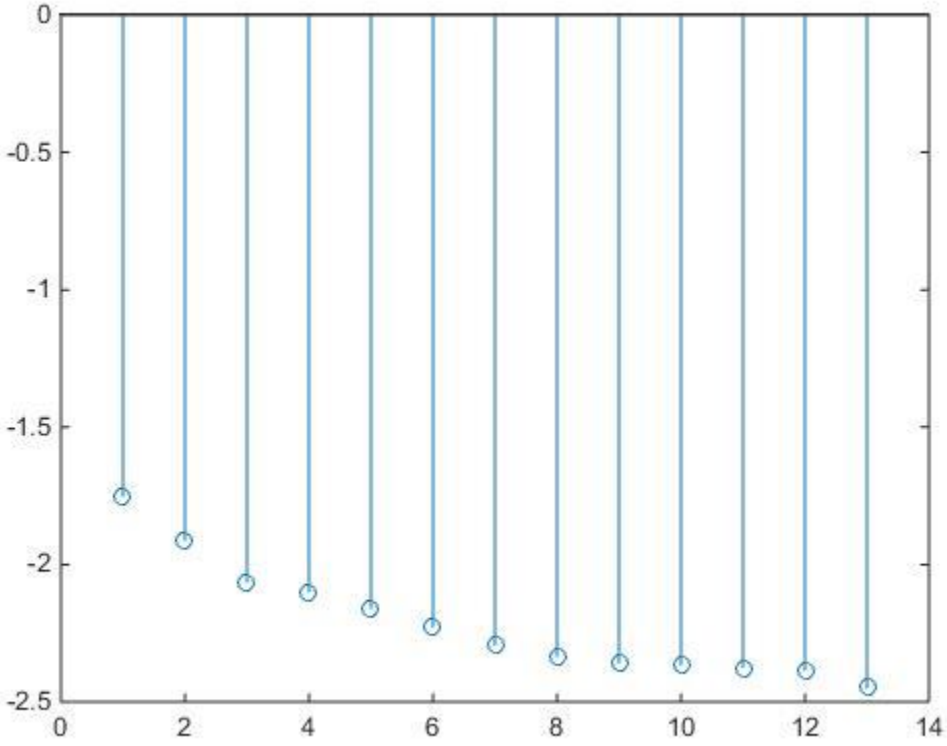


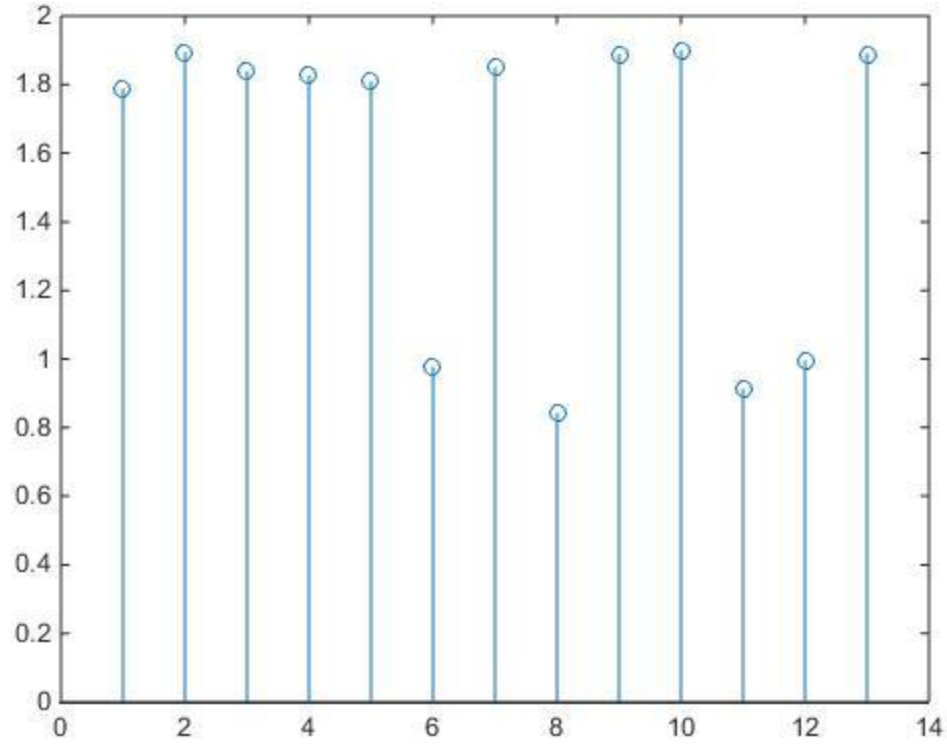


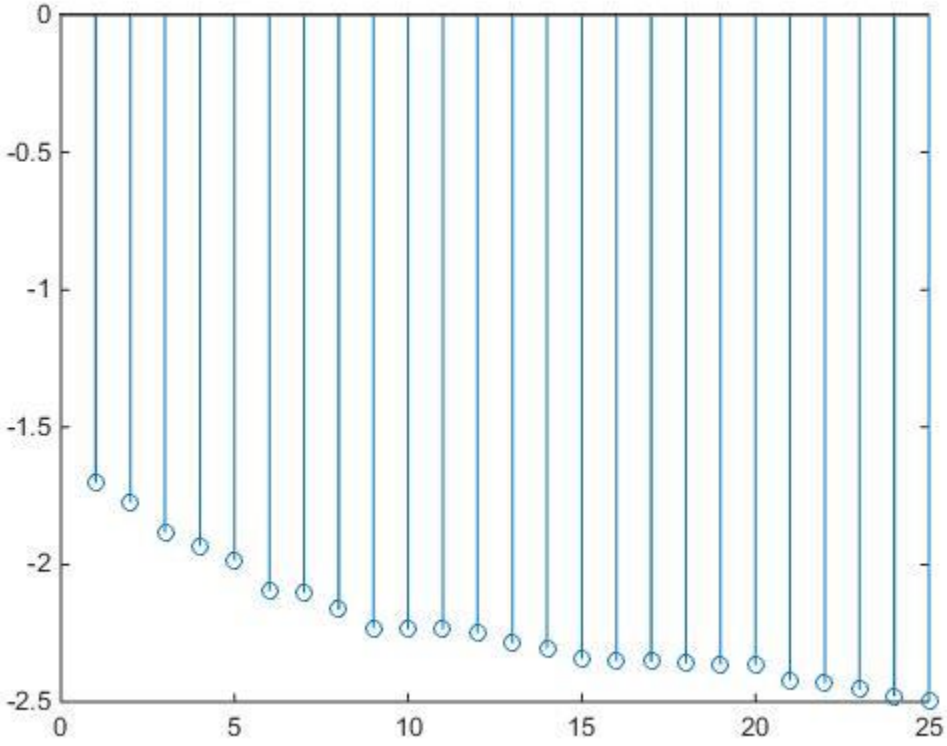

200
400
600
800
1000
200
400
600
800
1000
1200
1400
1600
1800

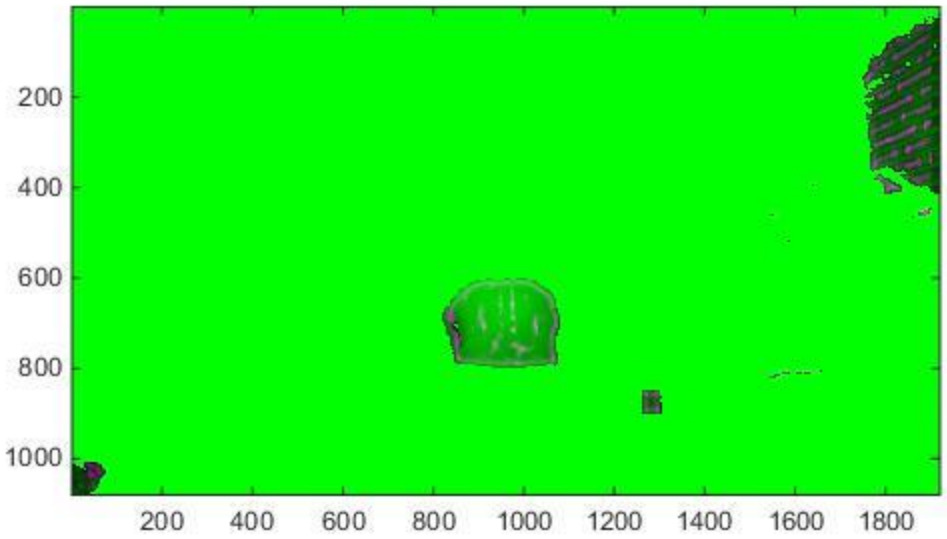

200
400
600
800
1000
200
400
600
800
1000
1200
1400
1600
1800

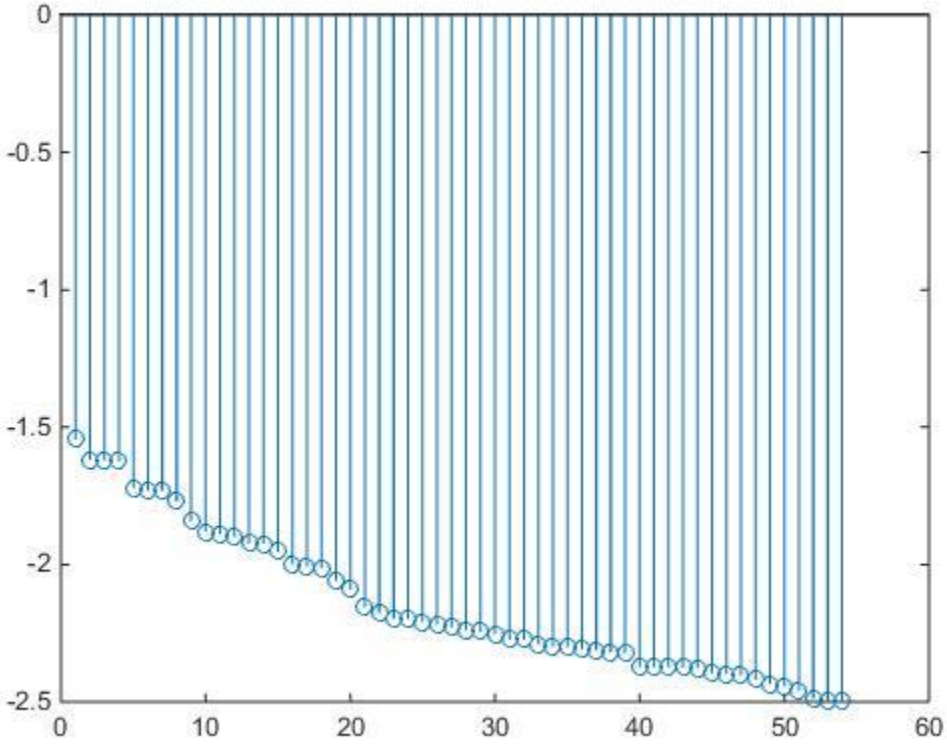

200
400
600
800
1000
200
400
600
800
1000
1200
1400
1600
1800

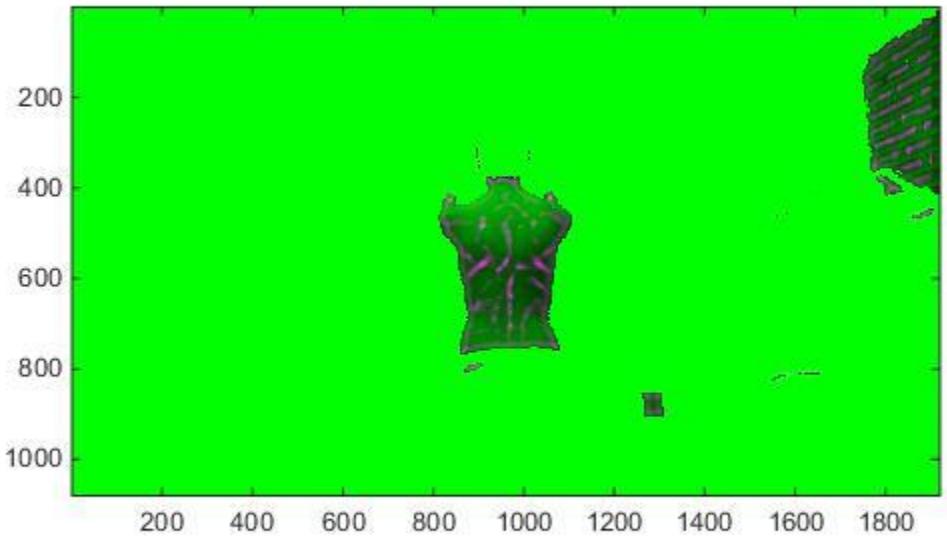

0
-0.5
-1
-1.5
-2
-2.5
0
2
4
6
8
10
12
14

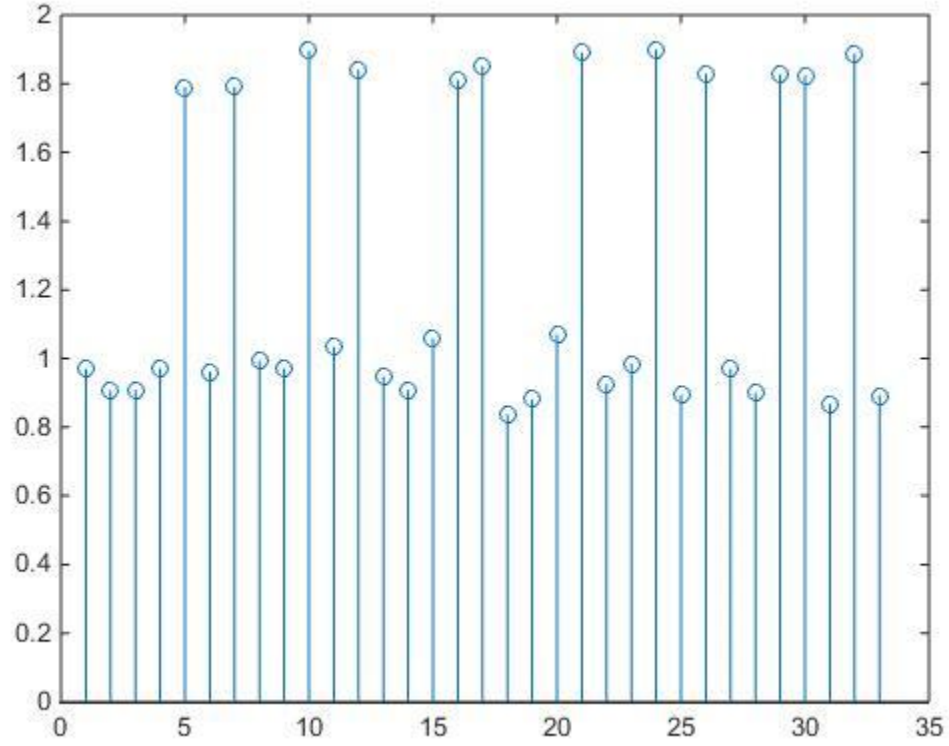

2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14

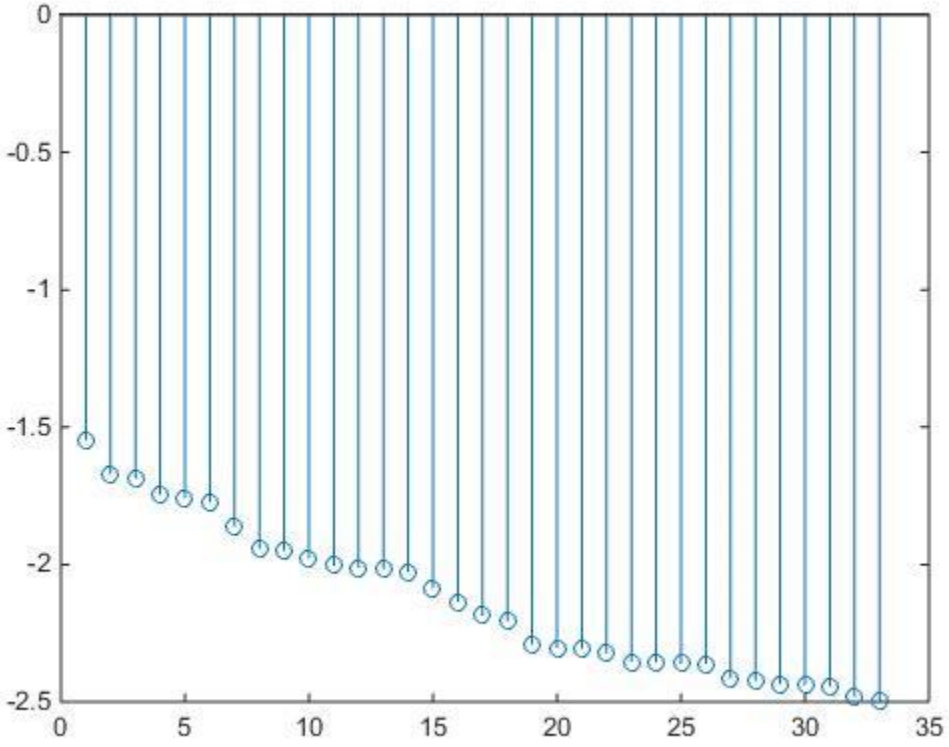

0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25

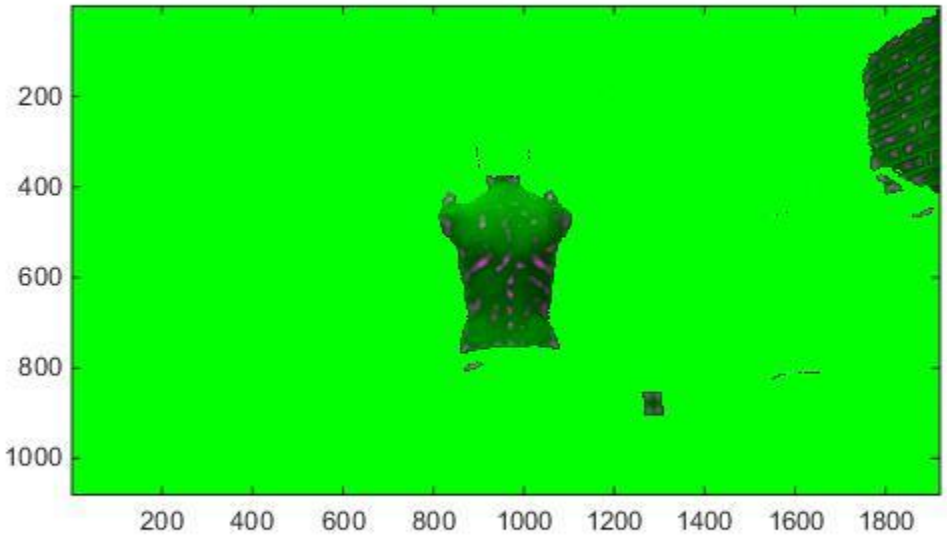

200
400
600
800
1000
200
400
600
800
1000
1200
1400
1600
1800

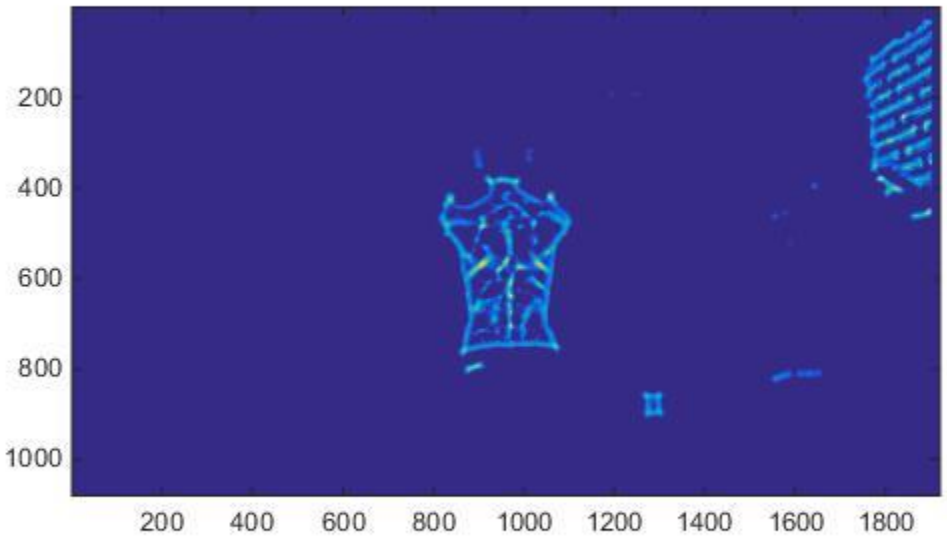

0
-0.5
-1
-1.5
-2
-2.5
0
10
20
30
40
50
60

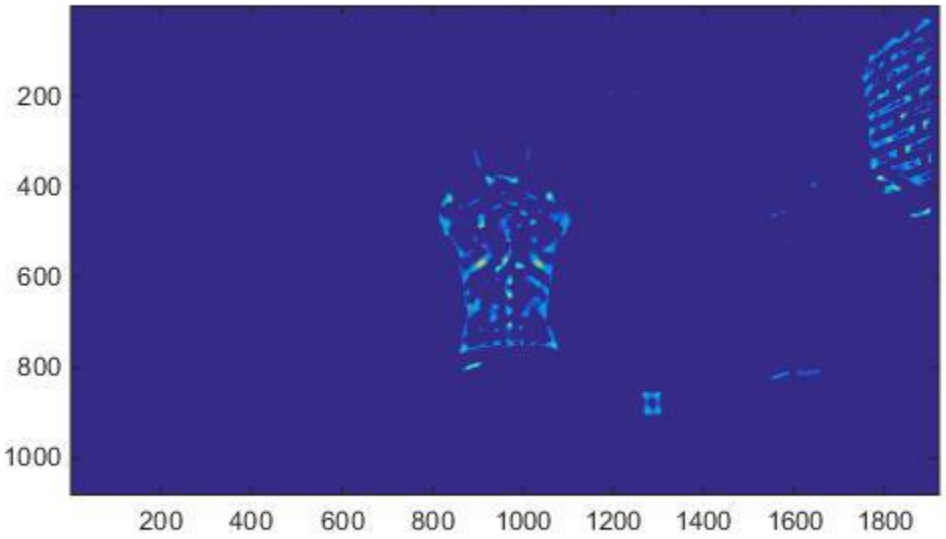

200
400
600
800
1000
200
400
600
800
1000
1200
1400
1600
1800

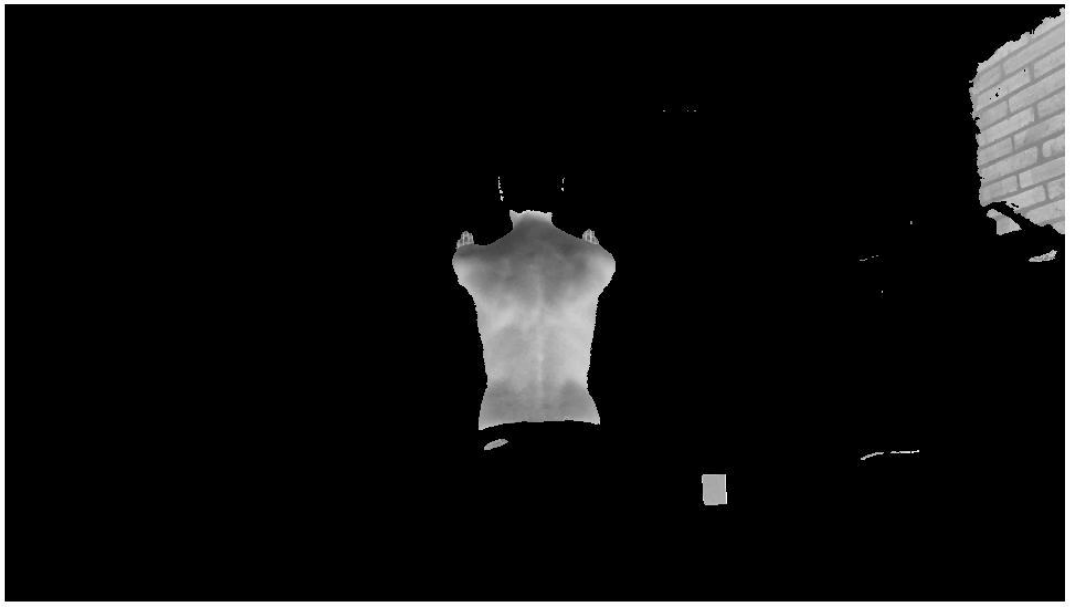

2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
30
35

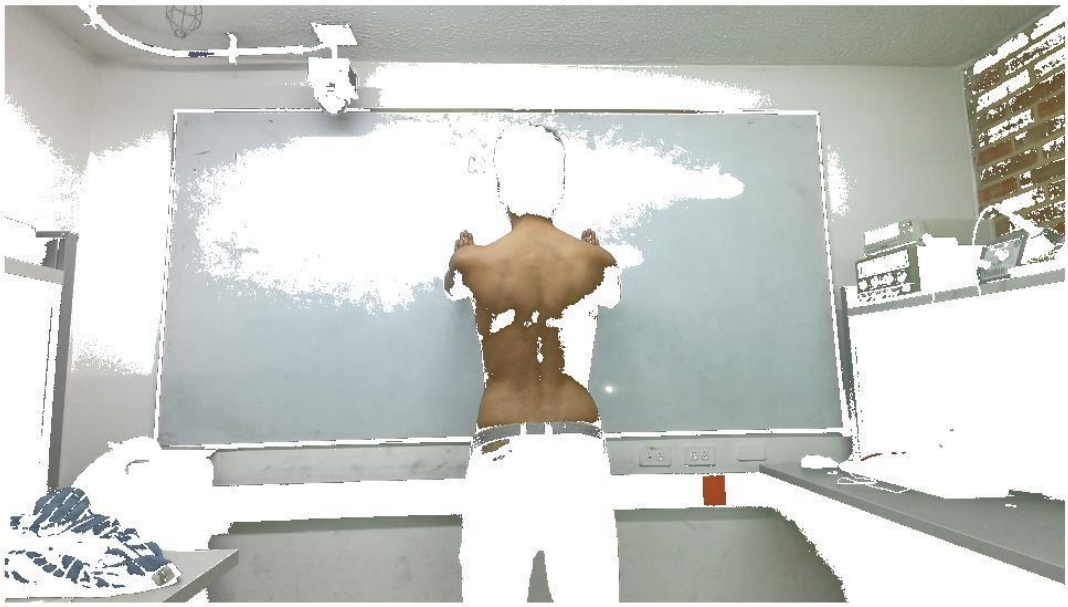

0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25
30
35


200
400
600
800
1000
200
400
600
800
1000
1200
1400
1600
1800


200
400
600
800
1000
200
400
600
800
1000
1200
1400
1600
1800


200
400
600
800
1000
200
400
600
800
1000
1200
1400
1600
1800





Original Data