Jaydeep Mahajan- CE66- IP-LAB5

Smoothing Filters

- Spatial filtering:

  It consist :

  - Image neighbourhood and sub image.
  - Predefined operation.

  The sub-image is called a filter, mask, kernel, template or window.

  If operation is linear, then the filter is called a linear spatial filter otherwise nonlinear.

- Nonlinear filter:

  It requires:

  - Size of neighbourhood.
  - Operation performed.

  Operations like max,min etc.

Smoothing spatial filters:

Linear spatial filter for smoothing:

- Averaging filter
- Lowpass filter

This filters use for -

- Noise reduction
- Undesirable side effect : blur edges

Important Function in Octave:

Standard box filter:

```
function s = standard_box_filter(r,m,n)
   [M,N] = size(r);
   a = (m-1)/2;
   b = (n-1)/2;
   new_imsize = zeros(M + 2*a,N + 2*b);
   new_imsize(1+a:M+a,1+b:N+b) = r;
   sub_image = ((1/(m*n)).*ones(m,n));
   s = zeros(size(r));
   for i = 1+a:M+a,
     for j = 1+b:N+b,
       k = new_imsize(i-a:i+a,j-b:j+b);
       s(i-a,j-b) = sum(sum(k.*sub_image));
     endfor
   endfor
   s = uint8(s);
endfunction
```

Weighted average filter:

```
function s = weighted_average_filter(r,m,n)
  [M,N] = size(r);
  a = (m-1)/2;
  b = (n-1)/2;
  new_imsize = zeros(M + 2*a,N + 2*b);
  new_imsize(1+a:M+a,1+b:N+b) = r;
  weight_image = ones(m,n);
  weight_image(a+1,b+1) = 4;
  weight_image(a+1,b)=2;
  weight_image(a+1,b+2)=2;
  weight_image(a,b+1)=2;
  weight_image(a+2,b+1)=2;
  sub_image = ((1/(sum(sum(weight_image)))).*weight_image);
  s = zeros(size(r));
  for i = 1+a:M+a,
    for j = 1+b:N+b,
      k = new_imsize(i-a:i+a,j-b:j+b);
      s(i-a,j-b) = sum(sum(k.*sub_image));
    endfor
  endfor
  s = uint8(s);
endfunction
```

Task A: Take any of your grayscale photos and blur it with a standard box filter of size 3x3, 5x5 ,7x7 and 9x9. Comment on the amount of blurring and filter size. Assume padding of zeros.

* taskA.m | standard_box_filter.m

```
pkg load image;
r = imread('D:\Sem7\Image_Processing\Lab5\jaydeep.jpg');
imshow(r);
title("Original Image");

figure;

subplot(2,2,1);
s1 = standard_box_filter(r,3,3);
imshow(s1);
title("3 x 3 image");

subplot(2,2,2);
s2 = standard_box_filter(r,5,5);
imshow(s2);
title("5 x 5 image");

subplot(2,2,3);
s3 = standard_box_filter(r,7,7);
imshow(s3);
title("7 x 7 image");

subplot(2,2,4);
s4 = standard_box_filter(r,9,9);
imshow(s4);
title("9 x 9 image");
```

Original image

3 x 3 image

5 x 5 image

7 x 7 image

9 x 9 image

From the above images, We can conclude that if filter size increases, blurring gets more effective.

Task B : Observe border of image for results in (a). Justify the reason for dark borders. Comment on thickness of the border and filter size. Suggest a way to solve the issue. Implement your suggestion and show the code and results.

Code:

taskA.m | standard_box_filter.m | task2.m | border_solved_filter.m

```
1  r = imread('D:\Sem7\Image_Processing\Lab5\jaydeep.jpg');
2
3  subplot(1,2,1);
4  s1=standard_box_filter(r,7,7);
5  imshow(s1);
6  title("Blur Image with border");
7
8  subplot(1,2,2);
9  s2 = border_solved_filter(r,7,7);
10 imshow(s2);
11 title("Blur Image with no border");
12
13
```

Reason for dark border :

Previously, we took zero padded around image for filtering that means for edge and corner pixel for set mxn filter on image we used zero padding around the main image and because of zero padded image we get border around the blurred image.

And also note that as filter size increases, the border gets more darker and thick.

Solving this issue below code would be helpful.

In this problem replace zero padding with top,down,right,left pixel sequence of required size and for corner pixels take the average of the corresponding pixel from each corner from the main image.

Border_solved_filter function :

taskA.m | standard_box_filter.m | task2.m | border_solved_filter.m

```
function s = border_solved_filter(r,m,n)
  [M,N] = size(r);
  a = (m-1)/2;
  b = (n-1)/2;
  new_imsize = zeros(M + 2*a,N + 2*b);
  new_imsize(1+a:M+a,1+b:N+b) = r;

  left_part = r(:,1:b);
  right_part = r(:,N-b:N);
  upper_part = r(1:a,:);
  down_part = r(M-a:M,:);

  [M1,N1] = size(new_imsize);
  new_imsize(1+a:M1-a,1:b) = left_part;
  new_imsize(1+a:M1-a,N1-b:N1) = right_part;
  new_imsize(1:a,1+b:N1-b) = upper_part;
  new_imsize(M1-a:M1,1+b:N1-b) = down_part;

  new_imsize(1:a,1:b) = sum(sum(r(1:a,1:b)))/(a*b);
  new_imsize(M1-a+1:M1,1:b) = sum(sum(r(M-a+1:M,1:b)))/(a*b);
  new_imsize(1:a,N1-b+1:N1) = sum(sum(r(1:a,N-b+1:N)))/(a*b);
  new_imsize(M1-a+1:M1,N1-b+1:N1) = sum(sum(r(M-a+1:M,N-b+1:N)))/(a*b);

  sub_image = ((1/(m*n)).*ones(m,n));
  s = zeros(size(r));
  for i = 1+a:M+a,
    for j = 1+b:N+b,
      k = new_imsize(i-a:i+a,j-b:j+b);
      s(i-a,j-b) = sum(sum(k.*sub_image));
    endfor
  endfor
```

Output

Task C :Take any of your grayscale photos and blur it with a weighted average filter. Compare the amount of blurring with the standard box filter of the same size.

taskA.m | standard_box_filter.m | task2.m | border_solved_filter.m | task3.m

```
1  r = imread('D:\Sem7\Image_Processing\Lab5\jaydeep.jpg');
2  imshow(r);
3
4  figure;
5
6  s = weighted_average_filter(r,3,3);
7  imshow(s);
8  title("Image(weighted average filter 3 x 3)");
9
```

Original image

Image(weighted average filter 3 x 3)

Conclusion:

In weighted average filter pixels are multiplied by different coefficients, thus giving more importance to some pixels than others. In the mask the pixel at the center of the mask is multiplied by a higher value than other, thus giving this pixel more importance in the calculation of the average.Hence compared to standard box filter , weighted average filter is more effective.

Task D : Assume that you are working on some image enhancement application which gives the following functionality to the user.

1) Anti-aging: Removes the wrinkles on the input face image.

2) Beautify: Removes facial marks.

Take any of the color photos of a face and implement any (or both) of above functionality.

I.Anti-aging: Removes the wrinkles on the input face image.

er.m | task2.m | border_solved_filter.m | task3.m | weighted_average_filter.m | antiaging.m

```
function s = antiaging(r)
  [M,N] = size(r);
  a=1;
  b=1;
  new_imsize = zeros(M + 2*a,N + 2*b);
  new_imsize(1+a:M+a,1+b:N+b) = r;
  filter = zeros(3,3);
  filter_box(1,1) = 5;
  filter_box(1,2) = 7;
  filter_box(1,3) = 5;
  filter_box(2,1) = 7;
  filter_box(2,2) = 9;
  filter_box(2,3) = 7;
  filter_box(3,1) = 5;
  filter_box(3,2) = 7;
  filter_box(3,3) = 5;
  filter_box = filter_box/sum(sum(filter_box));
  s = zeros(size(r));
  for i = 1+a:M+a,
    for j = 1+b:N+b,
      k = new_imsize(i-a:i+a,j-b:j+b);
      s(i-a,j-b) = sum(sum(k.*filter_box));
    endfor
  endfor
  s = uint8(s);
endfunction
```

2.m | border_solved_filter.m | task3.m | weighted_average_filter.m | antiaging.m | task4a.m

```
r = imread("wrinkle_face.jpg");
face = r(75:400,200:500,:);
subplot(1,2,1);
imshow(face);
title("Wrinkled face");
s = uint8(zeros(size(face)));
s(:,:,1) = antiaging(face(:,:,1));
s(:,:,2) = antiaging(face(:,:,2));
s(:,:,3) = antiaging(face(:,:,3));
subplot(1,2,2);
imshow(s);
title("wrinkleless face");
```

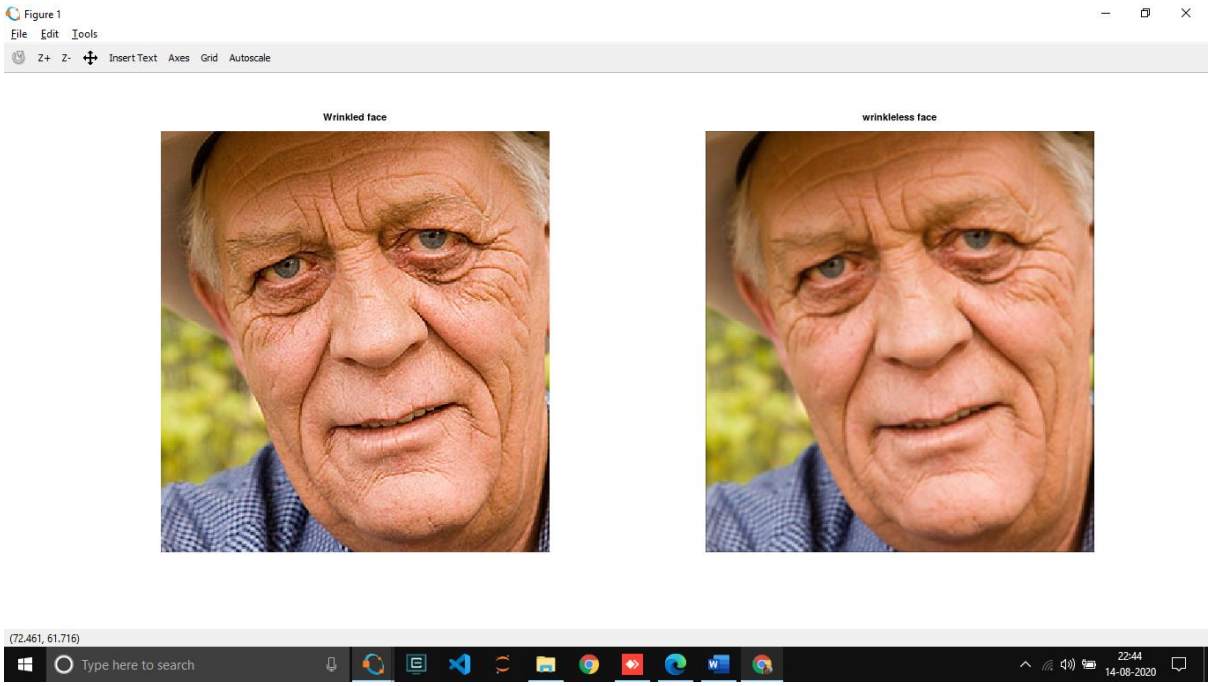


Figure 1
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Wrinkled face
wrinkleless face
(72.461, 61.716)
Type here to search
22:44
14-08-2020

I. Beautify : Removes facial marks.

Code:

ved_filter.m | task3.m | weighted_average_filter.m | antiaging.m | task4a.m | wrinkle_less.m

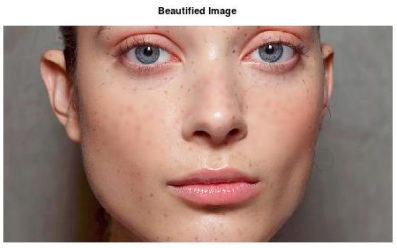
```
subplot(1,2,1);
imshow(r);
title("Original Image");

nose = r(97:240,530:640,:);
for i=1:5
  nose(:,:,1) = border_solved_filter(nose(:,:,1),7,7);
  nose(:,:,2) = border_solved_filter(nose(:,:,2),7,7);
  nose(:,:,3) = border_solved_filter(nose(:,:,3),7,7);
endfor
r(97:240,530:640,:) = nose;

l_cheek = r(180:370,270:440,:);
for i=1:5
  l_cheek(:,:,1) = border_solved_filter(l_cheek(:,:,1),7,7);
  l_cheek(:,:,2) = border_solved_filter(l_cheek(:,:,2),7,7);
  l_cheek(:,:,3) = border_solved_filter(l_cheek(:,:,3),7,7);
endfor
r(180:370,270:440,:) = l_cheek;

r_cheek = r(171:370,680:840,:);
for i=1:5
  r_cheek(:,:,1) = border_solved_filter(r_cheek(:,:,1),7,7);
  r_cheek(:,:,2) = border_solved_filter(r_cheek(:,:,2),7,7);
  r_cheek(:,:,3) = border_solved_filter(r_cheek(:,:,3),7,7);
endfor
r(171:370,680:840,:) = r_cheek;
subplot(1,2,2);
imshow(r);
title("Beautified Image");
```

Output :



Task E : Show the impact of multiple passes of the smoothing filter of the same size. Derive your conclusion on image quality and maximum number of passes of filter? What happens if an infinite(read very high!) number of passes are applied? Will it change image quality?

task3.m | weighted_average_filter.m | antiaging.m | task4a.m | wrinkle_less.m | task5.m

```
1  r = imread('jaydeep.jpg');
2  imshow(r);
3  title("Original Image");
4
5  figure;
6
7  s = standard_box_filter(r,3,3);
8  subplot(3,3,1);
9  imshow(s);
10 title("First Pass");
11
12 for i=2:9
13   s = standard_box_filter(s,3,3);
14   subplot(3,3,i);
15   imshow(s);
16   title(i);
17 endfor
18
```

Original image

First Pass

2

3

4

5

6

7

8

9