Coursework1 Image Filtering and Hybrid Images (For COMP6223)

Author: RUNYAO ZHOU 30018714

Coding Language: MATLAB

Code Statement

Five .m files have been created:

1. myhybrid.m
2. gaussian_template.m
3. template_convolve1.m
4. gaussian_template_test.m
5. myhybrid_test.m

1. 'function [result,gaussian1,gaussian2] = myhybrid(image1,image2)' is a function to hybridize image1 and image2, and create visualisations to show the effect. It also returns gaussian templates for checking.

The low frequencies of image1 and the high frequencies of image2 will be added together as the hybrid image.

It needs inputs of two different sigmas for two images, and the number of images you want to arrange in the result. The size of the result will change automatically with the number.

2. 'function template = gaussian_template(winsize,sigma)' is a function to create gaussian template with specific size and sigma.

Template here has same rows and cols. But I will prove later that my implementation works with arbitrary shaped template, as long as both dimensions are odd in '4.'.

3. 'function convolved = template_convolve1(image,template)' is a function to convolve the image with the template.

The template is rotated 180 degrees.

Pixel values are multiplied by the corresponding weighting coefficient and added together as a new value in the centre.

To avoid the black borders after convolving, the edge of image is expanded by zeros. The width is the half of the template.

4. ' function template = gaussian_template_test(winsize,sigma)' is a function to create gaussian template which has different size in two dimensions. Of course, they are odd.

5. ‘function [result,gaussian1,gaussian2] = myhybrid_test(image1,image2)’is almost the same with the ‘1.’, just change ‘gaussian_template’ to ‘gaussian_template_test’. It is the proof that my implementation can work with arbitrary shaped template.

Code Running

1. Files needed (all in MATLAB.zip)

   myhybrid.m
   gaussian_template.m
   template_convolve1.m
   gaussian_template_test.m
   myhybrid_test.m

   bicycle.bmp
   bird.bmp
   cat.bmp
   dog.bmp
   einstein.bmp
   fish.bmp
   marilyn.bmp
   motorcycle.bmp
   plane.bmp
   submarine.bmp

2. Load images
   All the images will be loaded in ‘double’. It is not necessary to convert the data type throughout the process, which is convenient.
   Enter following code in the command window:

```
image_dog=im2double(imread('dog.bmp'));
image_cat=im2double(imread('cat.bmp'));
image_submarine=im2double(imread('submarine.bmp'));
image_fish=im2double(imread('fish.bmp'));
image_motorcycle=im2double(imread('motorcycle.bmp'));
image_bicycle=im2double(imread('bicycle.bmp'));
image_bird=im2double(imread('bird.bmp'));
image_plane=im2double(imread('plane.bmp'));
image_einstein=im2double(imread('einstein.bmp'));
image_marilyn=im2double(imread('marilyn.bmp'));
```

3. Run ‘myhybrid’
   Enter following code in the command window:

```
[result,gaussian1,gaussian2]=myhybrid(image_dog,image_cat);
```

   Then you will be asked to enter ‘sigma1’, ‘sigma2’, ‘number of image’. Just enter them in the command window.
   Then in the ‘figure1’ and ‘figure2’ windows you will see the result.
   Gaussian templates can also be checked in Workspace.

In this code, ‘dog’ and ‘cat’ can be replaced by ‘marilyn’ and ‘einstein’ or other pairs of aligned images.

4. Result

```
>> [result,gaussian1,gaussian2] = myhybrid(image_dog,image_cat);
sigma1=6
sigma2=7
number of image:5
```

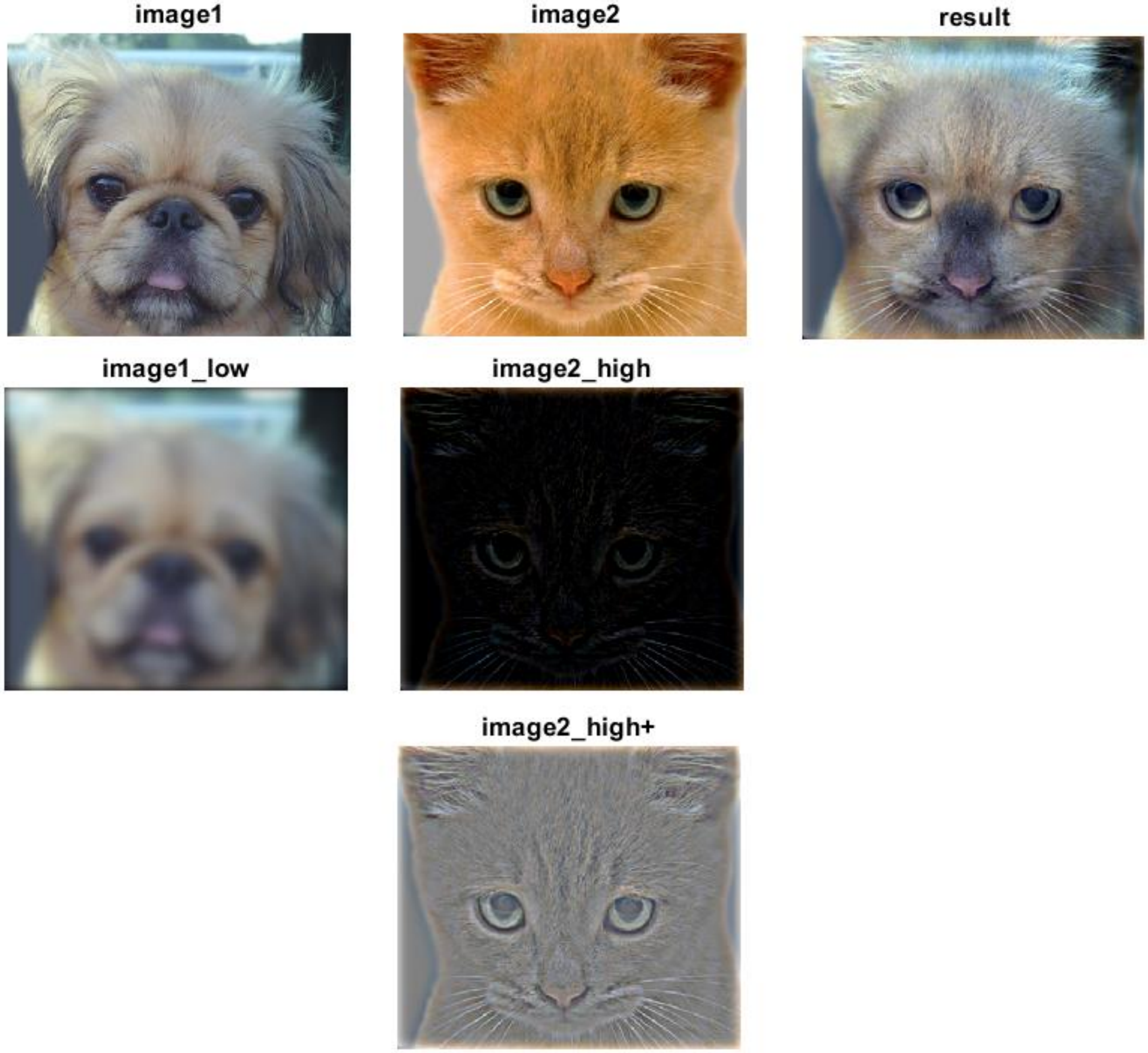


Figure 1 dogcat_1

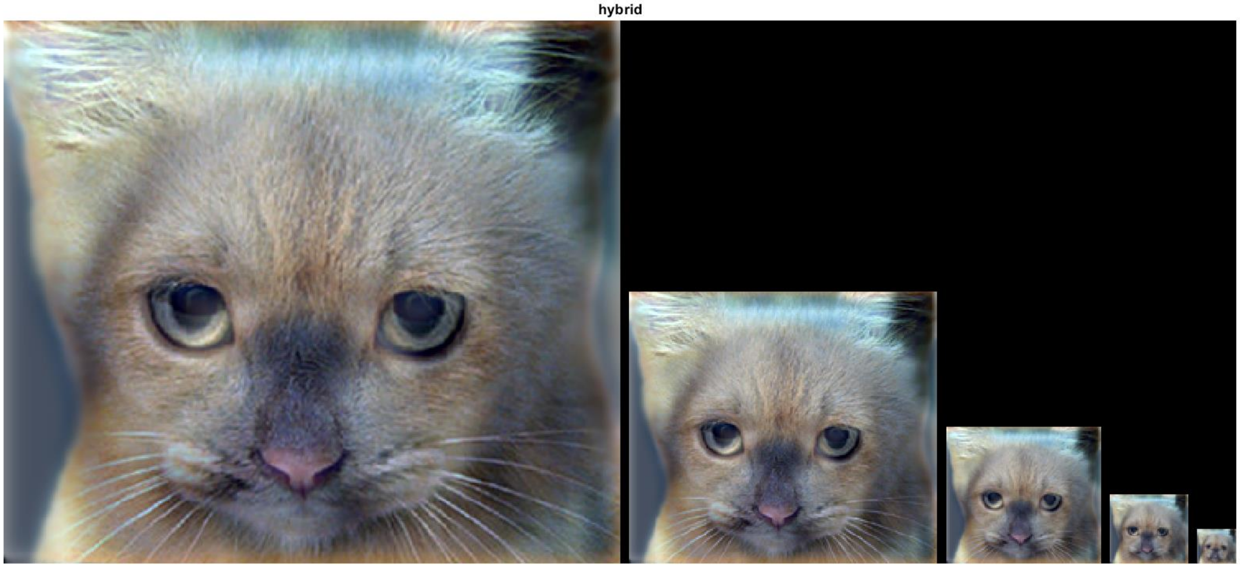


Figure 2 dogcat_2

Other hybrid images:

```
>> [result,gaussian1,gaussian2] = myhybrid(image_marilyn,image_einstein);
sigma1=3
sigma2=2
number of image:6
```

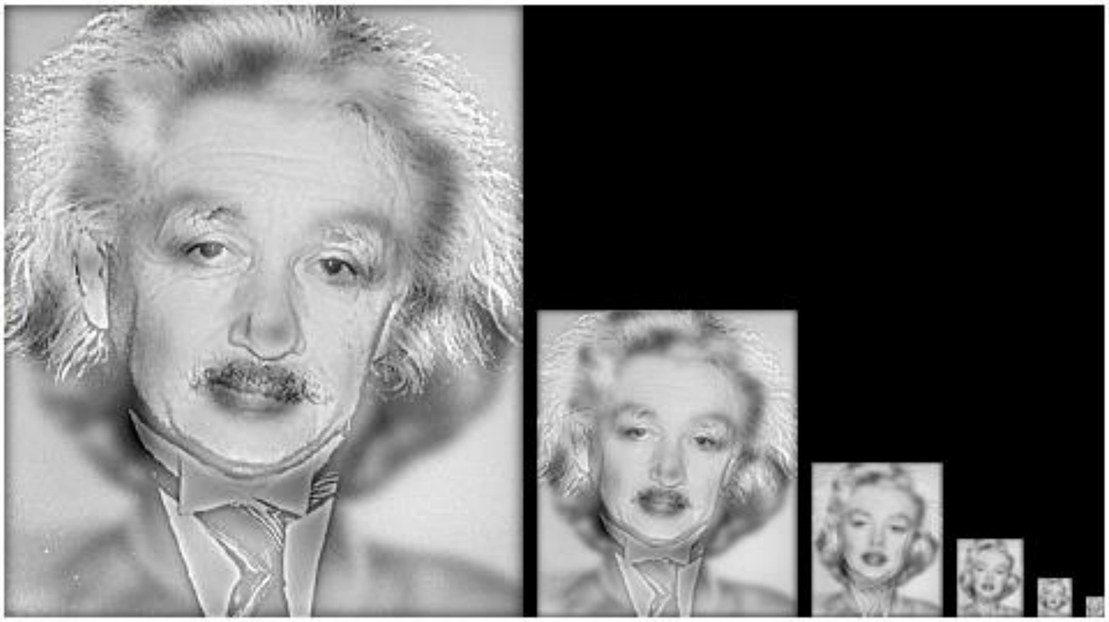


Figure 3 marilyneinstein_2

```
>> [result,gaussian1,gaussian2]=myhybrid(image_submarine,image_fish);
```

sigma1=7
sigma2=4
number of image:6

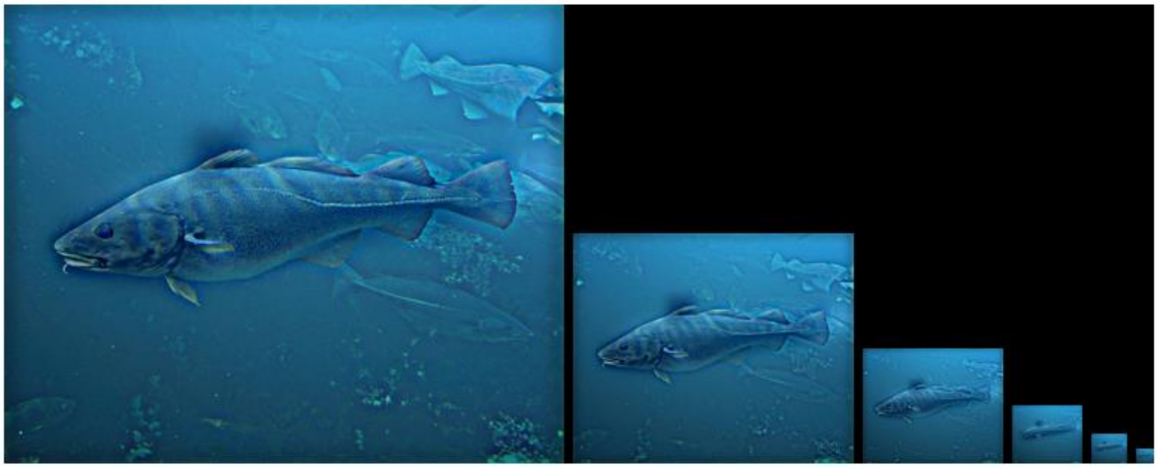
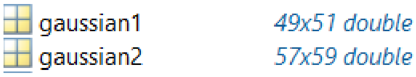


Figure 4 submarinefish_2

Here is the proof that my implementation works with arbitrary shaped template, as long as both dimensions are odd.

```
>> [result,gaussian1,gaussian2]=myhybrid_test(image_dog,image_cat);
sigma1=6
sigma2=7
number of image:6
```



```
gaussian1      49x51 double
gaussian2      57x59 double
```

Figure 5 arbitrary shaped template

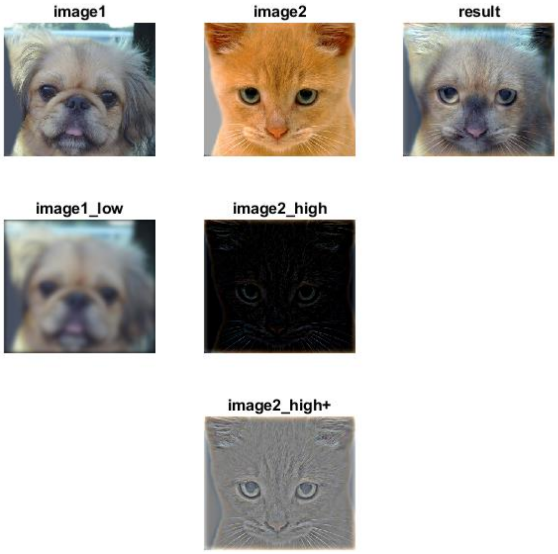


Figure 6 arbitrary shaped template_result

Code

1) myhybrid.m

```
function [result,gaussian1,gaussian2] = myhybrid(image1,image2)
%% the role of this function.
%hybridize image1 and image2.
%keep low frequencies of image1.
%keep high frequencies of image2.

%% create two gaussian filters.
%input two different sigmas for two images separately.
sigma1=input('sigma1=');
winsize1=floor(8*sigma1+1);    %this is from Coursework1 Statement.
if rem(winsize1,2)==0
   winsize1=winsize1+1;
end
sigma2=input('sigma2=');
winsize2=floor(8*sigma2+1);
if rem(winsize2,2)==0
   winsize2=winsize2+1;
end

%create two gaussian filters.
gaussian1=gaussian_template(winsize1,sigma1);
gaussian2=gaussian_template(winsize2,sigma2);

%% first image first.
%divide the image into tree channels of R,G,B.
image1_r=image1(:,:,1);
image1_g=image1(:,:,2);
image1_b=image1(:,:,3);

%convolve each channel with Gaussian template.
image1_r_c=template_convolve1(image1_r,gaussian1);
image1_g_c=template_convolve1(image1_g,gaussian1);
image1_b_c=template_convolve1(image1_b,gaussian1);

%put three channels together and get the low frequencies of image1
image1_c(:,:,1)=image1_r_c;
image1_c(:,:,2)=image1_g_c;
image1_c(:,:,3)=image1_b_c;

%% same for image2.
```

```
image2_r=image2(:,:,1);
image2_g=image2(:,:,2);
image2_b=image2(:,:,3);

image2_r_c=template_convolve1(image2_r,gaussian2);
image2_g_c=template_convolve1(image2_g,gaussian2);
image2_b_c=template_convolve1(image2_b,gaussian2);

image2_c(:,:,1)=image2_r_c;
image2_c(:,:,2)=image2_g_c;
image2_c(:,:,3)=image2_b_c;

%subtract the low-pass version from itself, get the high frequencies of
%image2, named image2_high.
%image2_high is hard to see, so add 0.5 to every pixel.
image2_high=image2-image2_c;
image2_high1=image2_high+0.5;

%% add the low and high frequencies together and get the hybrid image.
result=image1_c+image2_high;

%% create an arrangement of hybrid image to show the effect.
%input the number of picture you want, and it will automatically create the
%arrangement of images.

%strategy of drawing:
%draw a initial blank board whose size equals the size of original hybrid image.
%create a for-loop to add new blank area to 'board' and draw one image each time.
%each image is half in row and col of the former one.

number=input('number of image:');

[row1,col1,~]=size(result);
%~ will not be used, but it is necessary to put it here or it will go error
%because the 'result' has three channels (e.g.'361*410*3 double' for the
%dog and cat).
space=6;    %the space between each image.
scale=1/2;  %scaling ratio
tempresult=result;  %it is the image itself in every step of for-loop.
board(1:row1,1:col1,3)=0;   %the initial blank board must have 3 channels too.
x=1;    %x is abscissa of each drawing.
rows(number)=0; %create matrix to record the size of every image.
cols(number)=0;
for i=1:number
```

```
    [rows(i),cols(i),~]=size(tempresult);
    if i>1
        x=x+cols(1,i-1)+space;  %'i-1' can not be 0, so use 'if' to avoid it.
        tempboard(row1,space+cols(1,i),3)=0;    %create new blank area.
        board=cat(2,board,tempboard);   %add the new blank area to 'board'.
        board(row1-rows(1,i)+1:row1,x:x+cols(1,i)-1,1:3)=tempresult;    %draw new image.
        clear tempboard;    %need a new size of blank area each time.
        %Matlab just tell me that in every loop 'tempboard' will be
        %calculated and has a different size. It will be slow if the
        %for-loop is big. I have tried to create a blank board before the
        %for-loop that has a whole length. The length is calculated by
        %a*(1-r^n)/(1-r).Then I just found out that Matlab use
        %'ceiling' to calculate the size of the image which has been resized.
        %So the 'whole length' may not be equal to the sum of each iamge
        %and the space between them. That's why I calculate the length
        %separately in loop instead of calculating it before for-loop.
        %Maybe the size of each image can be calculate in an indepent
        %for-loop. Then we can get the 'whole length' before the image
        %processing loop. It may be more fast if we need to calculate
        %hundreds of images. But in this coursework we just need a small
        %number of images. So my calculation is still fast.
    else
        board(1:rows(1,i),1:cols(1,i),1:3)=tempresult;
    end
    tempresult=imresize(tempresult,scale);
end

%% show the results
figure(1)
subplot(3,3,1),imshow(image1),title('image1');
subplot(3,3,4),imshow(image1_c),title('image1\_low');
subplot(3,3,2),imshow(image2),title('image2');
subplot(3,3,5),imshow(image2_high),title('image2\_high');
subplot(3,3,8),imshow(image2_high1),title('image2\_high+');
subplot(3,3,3),imshow(result),title('result');
figure(2)
imshow(board),title('hybrid');
```

2) gaussian_template.m

```
function template = gaussian_template(winsize,sigma)
%create gaussian template
centre=floor(winsize/2)+1;
```

```
sum=0;
for i=1:winsize
   for j=1:winsize
      template(j,i)=exp(-(((j-centre)*(j-centre))+((i-centre)*(i-centre)))/(2*sigma*sigma));
      sum=sum+template(j,i);
   end
end
template=template/sum;
```

3) template_convolve1.m

```
function convolved = template_convolve1(image,template)
%convolve the image and the template
%expand image to reduce black borders
[rows,cols]=size(image);
[trows,tcols]=size(template);
tr=floor(trows/2);
tc=floor(tcols/2);
convolved(1:(rows+2*tr),1:(cols+2*tc))=0;
%expand the edge of the image with zeros, the width is half of template
temp(1:(rows+2*tr),1:(cols+2*tc))=0;
temp(tr+1:tr+rows,tc+1:tc+cols)=image;
image=temp;
for x=tc+1:cols+tc
   for y=tr+1:rows+tr
      sum=0;
      for iwin=1:tcols
         for jwin=1:trows
            sum=sum+image(y+jwin-tr-1,x+iwin-tc-1)*template(trows-jwin+1,tcols-iwin+1);
            %the template is rotated 180 degrees.
            %pixel values are multiplied by the corresponding weighting
            %coefficient and added together as a new value in the
            %centre.
         end
      end
      convolved(y,x)=sum;
      end
end
temp1(1:rows,1:cols)=convolved(tr+1:tr+rows,tc+1:tc+cols);  %delete borders
convolved=temp1;
```

4) gaussian_template_test.m

```
function template = gaussian_template_test(winsize,sigma)
%create gaussian template [winsizer,winsizerc]
winsizer=winsize;
winsizec=winsize+2;

centrer=floor(winsizer/2)+1;
centrec=floor(winsizec/2)+1;
sum=0;
for i=1:winsizec
   for j=1:winsizer
      template(j,i)=exp(-(((j-centrer)*(j-centrer))+((i-centrec)*(i-centrec)))/(2*sigma*sigma));
      sum=sum+template(j,i);
   end
end
template=template/sum;
```

5) myhybrid_test.m

change

```
gaussian1=gaussian_template(winsize1,sigma1);
gaussian2=gaussian_template(winsize2,sigma2);
```

into

```
gaussian1=gaussian_template_test(winsize1,sigma1);
gaussian2=gaussian_template_test(winsize2,sigma2);
```