Name:LAI WEIZHONG

NO.28969367

EMAIL:wl4a17@soton.ac.uk

* Object

Make an image which have different appearances in different distance. The main method is to combine the low-pass image and the high-pass image, which can let human’s eye observe different display in different distance. Firstly, create a convolution function to get a way to change an image. Secondly, create a Gaussian template to blur an image. Thirdly, use the original image minus the low-pass image then get the high-pass image. Finally, create a different size of the combine image to observe the different of the display in different distance.

* Convolution

Code for convolution:

function convolution = convolute(image,template)

    [imageRow, imageColum] = size(image);

    [templateRow, templateColum] = size(template);

    temp = zeros(imageRow,imageColum);

    rowi = floor(templateRow/2)+1;

    columj = floor(templateColum/2)+1;

    for i=rowi:imageRow-rowi+1 %i,j is the centrel point of the template.

        for j=columj:imageColum-columj+1

             %caculate sum

             sum = 0;

            for ci = 1:templateRow

                for cj=1:templateColum

                    %-1 means no change at first loop.

                    sum = sum + template(ci,cj)\*image(i-floor(templateRow/2)+ci-1,j-floor(templateColum/2)+cj-1);

                end

            end

            temp(i,j) = sum;

        end

    end

    convolution = temp;

end

In the function of convolution, the elements of each column and each row of template image is multiplied by the elements of each column and each row of corresponding original image and add up all the multiplication results and the results is the central point of the template image. After that, move the template to another position of original images and repeat the same calculation.

* Gaussian template

Code for Gaussian template:

function GaussianTemplate = CreateGaussianTemplate(sigma)

    Tsize = floor(8\*sigma + 1);

    if rem(Tsize,2) == 0

        Tsize = Tsize + 1;

    end

    temp = zeros(Tsize,Tsize);

    summ = 0;

    centre = floor(Tsize/2)+1;

    for i=1:Tsize

        for j=1:Tsize

            temp(i,j) = exp(-(((j-centre)^2+(i-centre)^2))/(2\*sigma^2));

            summ = summ + temp(i,j);

        end

    end

    for i=1:Tsize

        for j=1:Tsize

temp(i,j) = temp(i,j)/summ; %normalise every element

        end

    end

    GaussianTemplate = temp;

end

Gaussian template is to blur the original image, removing all the high frequencies, smoothing the pixel around the centrel pixel.

Gaussian format:

# Hybrid image

* Function of Hybrid image is to combine the low-pass image and high-pass image. The High-pass image is created by original image minus low-pass image. Original image minus low frequencies is high frequencies.

The code for low-pass image:

cat = imread('../image/dog.bmp');

cat = uint8(cat);

a = length(cat(:,1,1)); %get the length of first dimension

b = length(cat(1,:,1)); %get the length of second dimension

c = length(cat(1,1,:)); %get the length of third dimension

templateCat = CreateGaussianTemplate(4);

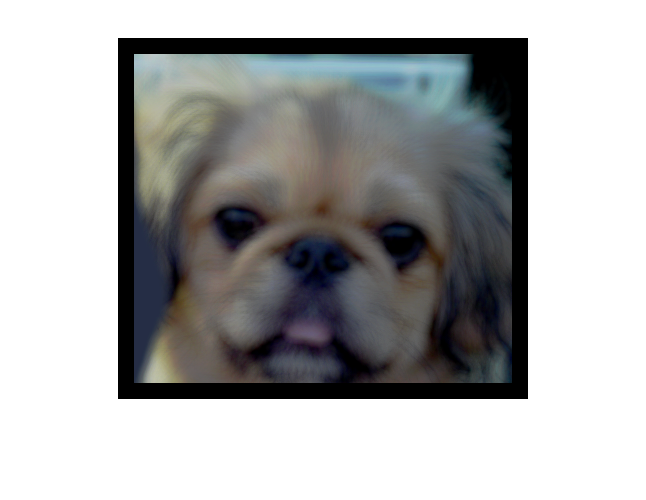
GaussianCat = zeros(a,b,c); %Edge complement 0

GaussianCat(:,:,1) = convolute(cat(:,:,1),templateCat); %make a convolution in first dimension

GaussianCat(:,:,2) = convolute(cat(:,:,2),templateCat); %make a convolution in second dimension

GaussianCat(:,:,3) = convolute(cat(:,:,3),templateCat); %make a convolution in third dimension

imshow(uint8(GaussianCat));



the picture is shown like this above. The sigma of Gaussian template is 4, and calculate red, green, blue separately. The size of black edge is depended on the window size of template. Means while, the window size is depended on sigma.

* Code for window size:

Tsize = floor(8\*sigma + 1);

if rem(Tsize,2) == 0

Tsize = Tsize + 1;

end

* Code for high-pass image:

dog = imread('../image/cat.bmp');

dog = double(dog);

templateDog = CreateGaussianTemplate(4);

GaussianDog = zeros(a,b,c);

GaussianDog(:,:,1) = convolute(dog(:,:,1),templateDog);

GaussianDog(:,:,2) = convolute(dog(:,:,2),templateDog);

GaussianDog(:,:,3) = convolute(dog(:,:,3),templateDog);

Hdog = dog-GaussianDog;

for i=1:a

for j=1:b

for k=1:c

Hdog(i,j,k) = Hdog(i,j,k) + 20;

end

end

end

imshow(uint8(Hdog));



the picture is shown like this above, the high-pass image is created by original image minus low-pass image. After that, add 20 to every pixel.

* Code for combining the low-pass image and high-pass image

for i=1:a

for j=1:b

for k=1:c

HybridImage(i,j,k) = (GaussianCat(i,j,k) + Hdog(i,j,k));

end

end

end

imshow(uint8(HybridImage));



The picture is shown like this above, add the tow pixel between low-pass image and high-pass image.

* Code for Observe in different distance:

%create 3 different size images

Hybrid1 = imresize(HybridImage,0.5);

Hybrid2 = imresize(HybridImage,0.5\*0.5);

Hybrid3 = imresize(HybridImage,0.5\*0.5\*0.5);

%make the 3 different size image the same width

Hybrid1(a,length(Hybrid1(1,:,1)),c) = 255;

Hybrid2(a,length(Hybrid2(1,:,1)),c) = 255;

Hybrid3(a,length(Hybrid2(1,:,1)),c) = 255;

%combine all the 4 image matrices together

combineImage = [HybridImage,Hybrid1,Hybrid2,Hybrid3];

%make the black to white

for i=1:a

for j=1:length(combineImage(1,:,1))

for k=1:c

if combineImage(i,j,k) == 0

combineImage(i,j,k) = 255;

end

end

end

end

imshow(uint8(combineImage));



The picture is shown like this above, combine different sizes of picture matrices in one matrix. The bigger image looks like a cat, and the smaller image look like a dog.

Test in Einstein and Marilyn.



Test in fish and submarine.



* Summaries

According to the Hybrid image, I know the principle of convolution, Gaussian filter and how to make an image which have different display in different distance.