CSE 5524 HW3

Professor: Jim Davis

Author: Yi Zhao ([zhao.2175@osu.edu](mailto:zhao.2175@osu.edu))

1. The filter is obtained by two separate given Gaussian filter with a=.4. (Gy = Gx.T) The image from higher level is filtered from image in lower level. The Gaussian and Laplacian pyramid is shown below: (The image size is proportionally cropped for printing purpose.)

The size from left to right is cropped in half between each layer. The Gaussian pyramid shows the low-frequency images and the error pyramid shows the high-frequency images (you can see outlines are showed in detail in error pyramid). If you try to zoom in the smaller image to the same size as the larger one – it won’t be as clear as the one in lower layers, as half of the details are lost.

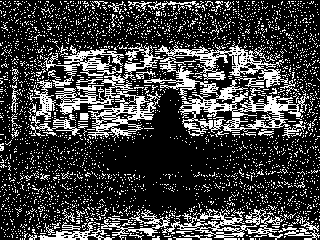
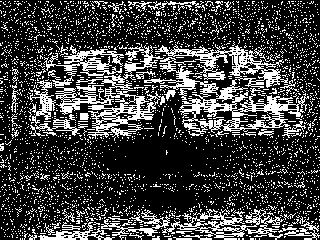
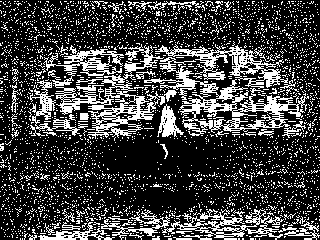
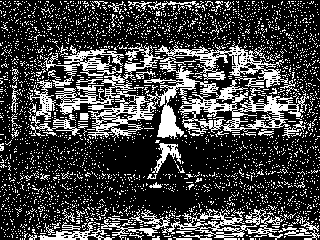
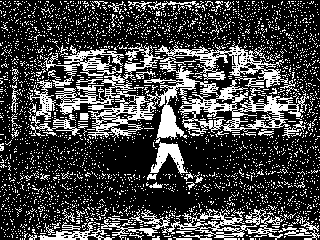
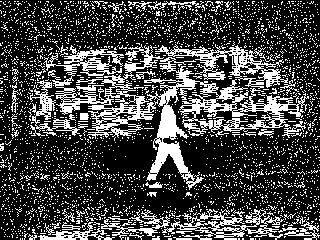
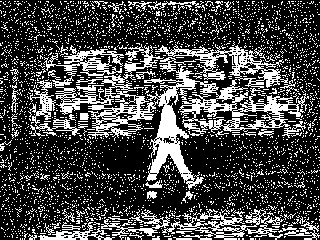
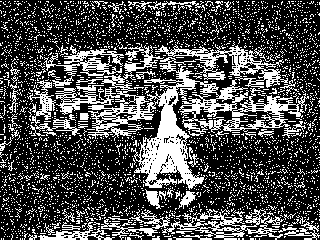
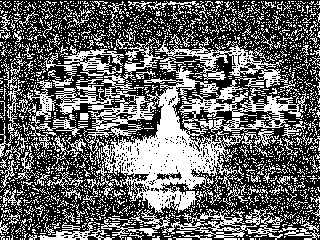


1. By using different threshold values, I obtained 10 images as shown below.

From left to right, top to bottom, the corresponding T value is: **[10 20 35; 50 75 100; 125 150 200;255]**.

Apparently, there is too much noise in background and it’s hard to completely separate the person from noisy backgrounds. Start from 10, the person becomes separated as T value increases, and the best result occurs at **T = 50~75**. After that, the outline and the body of person is gradually melted and disappeared, while most of the noise in background still remains.

The possible reason for these noises is because the difference of ‘background’ between input image and background. The difference between pixels in person is similar to the difference between pixels in other regions, which makes it hard to remove the noises using threshold. It is important to keep the background similar between input I and background R, so that only the difference of person will be extracted and most of other regions will remains 0 (below threshold).

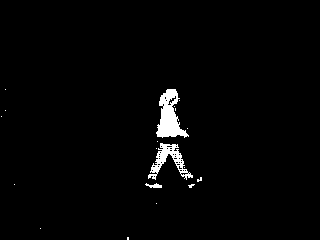
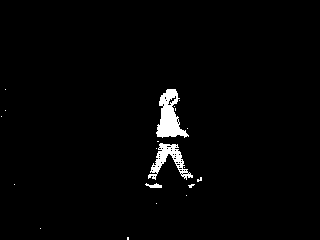
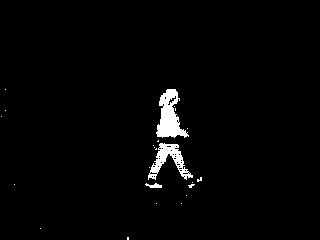
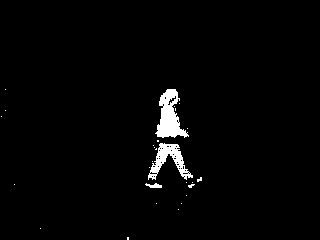
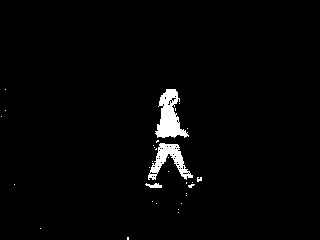
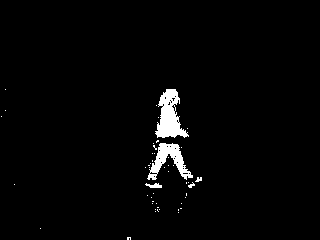
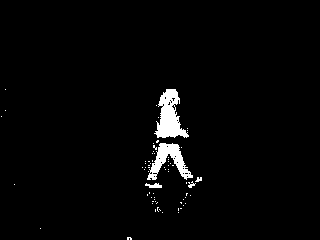
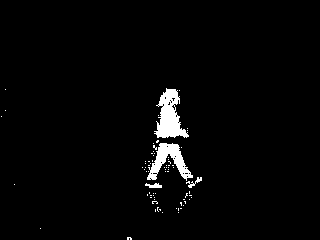
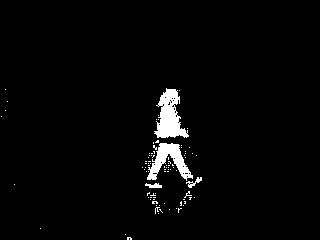
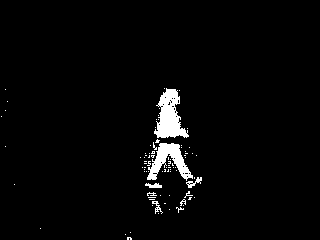
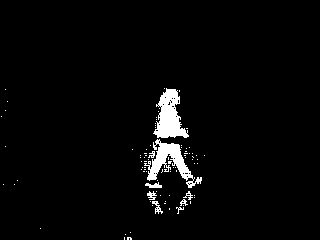
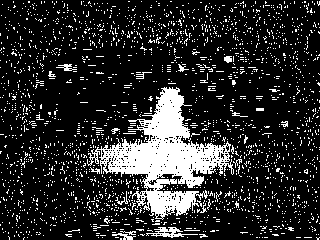
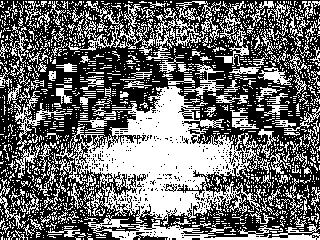


1. By using different threshold values, I obtained 11 images as shown below.

From left to right, top to bottom, the corresponding T value is: **1:20**.

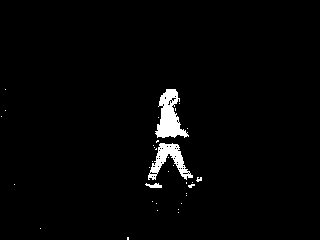
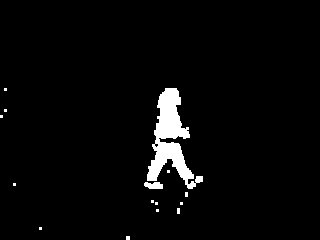
The result is quite satisfying compared to the result in problem 2. As we increase the threshold, the noise from the background is removed peacefully without removing the person’s outline. When threshold reaches **T = 15 - 16**, there are only little white pixels remain in other regions and the target object is separated completely. After that, we can see the target object is losing part of segment as T grows up, and some of the body parts are separated, which is not what we want to see.

The possible reason why the result is better than previous results is that statistical distance between the input image and multiple images of the same background can better measure how many differences between input value and background. The target object (which is supposed to have the most different pixels) will be kept almost entirely when we try to increase the threshold to remove noises from the background. However, if you choose threshold that over 16, it will start removing part of the body pixels. The best separation is only extracting object from background without losing body parts.

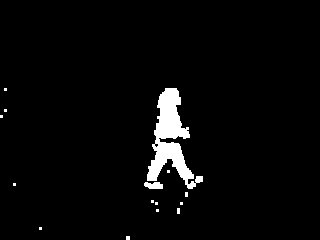
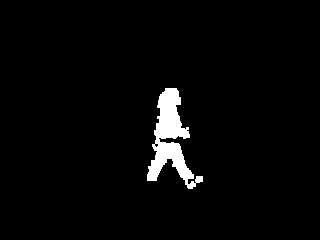


1. By using the chosen image (with T = 16), I obtained the following dilated image. (left input, right result)

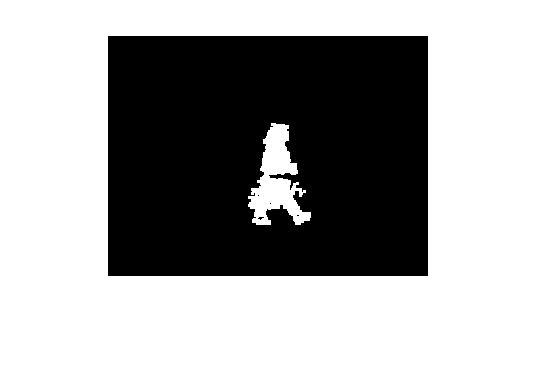
The dilation operation enlarges a binary region. You can see few pixels around the edge from input image are all dilated in result. The small tiny pixel around left bottom side is enlarged as well, and that make sense because dilation operation will put whole structuring element on result as long as a single pixel exists in that covered region.

 => 

1. By using bwlabel() function, we can select the desired region (the largest one). The connected components will be labeled the same as an entire region. We use 8-connected selection so that we won’t miss corner pixels (for example, this person’s front feet). We can select second-most frequent label as the target object (since 0 is always the most frequent). The left image is the image before selection, and the right image is the selected region. Now the person is almost entirely separated and extracted from background. However, you can also see this person lost his/her back foot, and the reason is that the color of his/her back ankle has less difference than other body parts from the background. If we increase threshold in question 3, we will most likely lose his waist as well (you can tell this by checking result with T = 18, 19, 20 from question 3: if there is no pixel around waist, then dilation and component connection (even 8-direction) will not work because of the huge gap between upper and lower body).

 => 

Here is the result using T = 9 rather than 16. It uses a small threshold to keep the foot connected. However, there is lot of noises around leg and considering pros and cons, I think the image above (Using T=16) is a better extraction of the target object.



HW3.m

% Author: Yi Zhao

% CSE 5524, HW1

% 09/01/2019

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Problem 1

a = 0.4;

wx = [.25-.5\*a .25 a .25 .25-.5\*a];

wy = reshape(wx, [5,1]);

% double, grayscale image. Make sure it's size is (odd, odd)

gauss = double(rgb2gray(imread('./data/1.jpg')));

for i = 1:4

% save gaussian pyramid from layer1 to layer2

imwrite(gauss/255, sprintf('./output/pyramid/gaussian\_%d.png',i));

nextGaussian = blurNsample(gauss, wx, wy);

est = interpolation(nextGaussian);

imwrite((gauss - est)/255, sprintf('./output/pyramid/laplacian\_%d.png',i));

% Set gauss to nextGaussian to continue building pyramid

gauss = nextGaussian;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Problem 2

T\_list = [10 20 35 50 75 100 125 150 200 255];

backgroundIm = double(imread('./data/bg000.bmp'));

inputIm = double(imread('./data/walk.bmp'));

binary\_output = abs(inputIm - backgroundIm);

for T = T\_list

binary\_copy = binary\_output;

binary\_copy(binary\_copy >= T) = 1;

binary\_copy(binary\_copy ~= 1) = 0;

% imshow(binary\_copy)

% pause;

imwrite(binary\_copy, sprintf('./output/background\_sub1/T%d.png',T))

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Problem 3

% Read images. Store 30 images as a cube

for i = 1:30

filename = sprintf('./data/bg%03d.bmp', i-1);

Im(:,:,i) = double(imread(filename));

end

% Calculate stdev & mean matrix

stdev = std(Im, 0, 3);

avg = mean(Im, 3);

% Do background subtraction 2

bn\_output = (inputIm - avg).^2 ./ stdev.^2;

T\_list2 = [0.1 0.2 0.5 1 2 5 10 20 50 75 100 150 200 250];

for T = T\_list2

bn\_copy = bn\_output;

bn\_copy(bn\_copy >= T) = 1;

bn\_copy(bn\_copy ~= 1) = 0;

% imshow(bn\_copy)

% pause;

imwrite(bn\_copy, sprintf('./output/background\_sub2/T%.1f.png',T))

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Problem 4

% Based on observation, 'T250.0.png' is used for best output

bsIm = double(imread('./output/background\_sub2/T250.0.png'));

d\_bsIm = bwmorph(bsIm, 'dilate');

imwrite(d\_bsIm, './output/dilated\_ouput.png');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Problem 5

[L, num] = bwlabel(d\_bsIm, 8);

% Based on observation, the largest rigion is labeled as 8

L(L ~= 8) = 0;

imwrite(L, './output/final\_ouput.png');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Problem 1 Helper function

% First blur the image using gaussian separable filter wx and wy,

% and then sample the image to half size.

function bluredSampled = blurNsample(image, wx, wy)

blured = imfilter(imfilter(image, wy, 'replicate'), wx, 'replicate');

bluredSampled = blured(1:2:end, 1:2:end);

end

% Implement the interpolation function

function newImage = interpolation(image)

newImage = zeros(size(image)\*2-1);

newImage(1:2:end, 1:2:end) = image;

row\_ave = conv2(image, [1 1], 'valid')/2;

col\_ave = conv2(image, [1;1], 'valid')/2;

newImage(1:2:end,2:2:end) = row\_ave;

newImage(2:2:end,1:2:end) = col\_ave;

mid\_ave = conv2(row\_ave,[1;1], 'valid')/2;

newImage(2:2:end, 2:2:end) = mid\_ave;

end