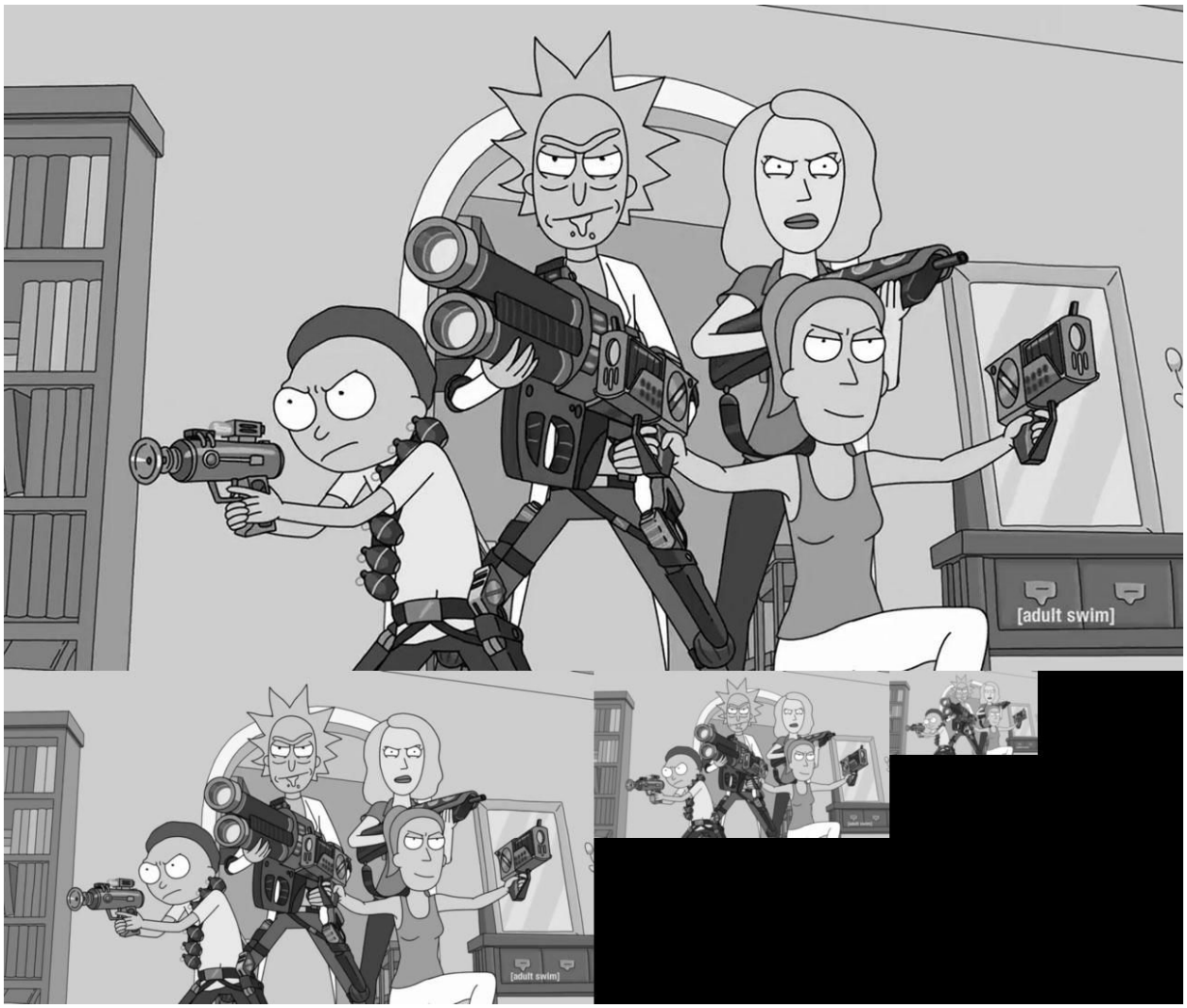
Computer Vision for HCI — Abhinav Mahalingam (.15)

CSE 5524

Homework #3

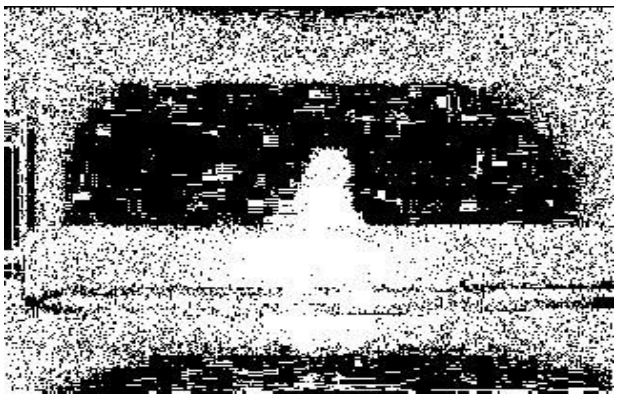
1. Generate a 4-level Gaussian pyramid (consider the original image as level-1) and the corresponding Laplacian pyramid of an image (select one from the web). Use the formula in the notes to first determine a viable image size, and crop the image (if needed) to test the pyramid code. Use a=0.4 for the Gaussian mask – use separable masks! Write/use functions for properly reducing and expanding an image. [7 pts]



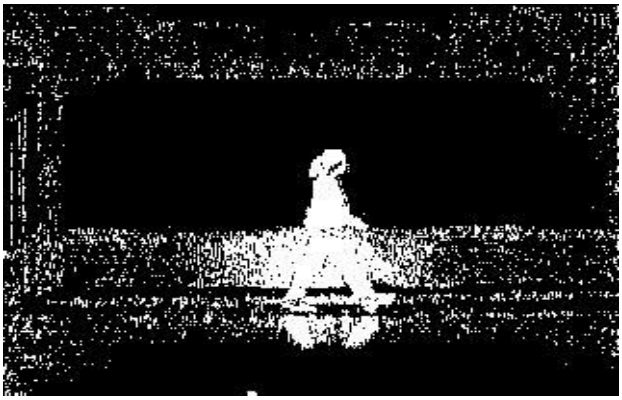
4 - Level Gaussian Pyramid

4 - Level Laplacian Pyramid

2. Using the grayscale images (walk.bmp, bg000.bmp) provided on the WWW site, perform background subtraction 1 using simple image subtraction to identify the object. Make sure your image is of type *double*! Experiment with thresholds. [2 pts]

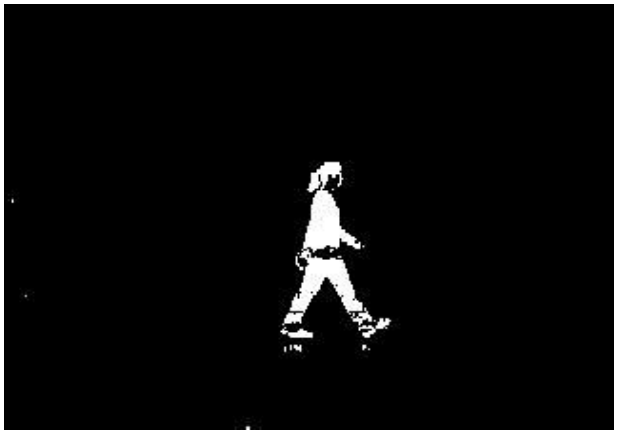


T = 1



T = 11

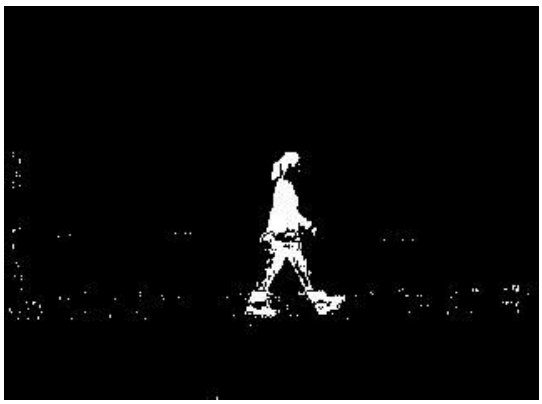
T = 21



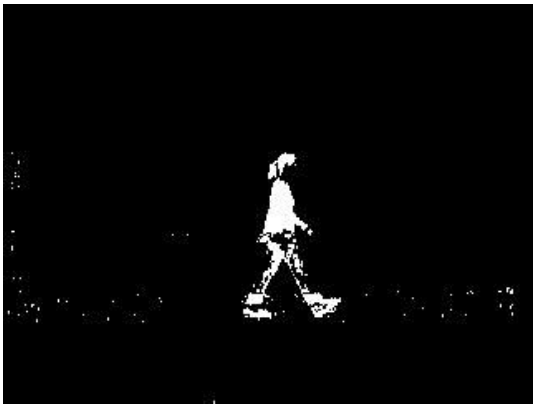
T = 61

As the value of threshold increases, we can see that the background subtraction improves. Although for very high threshold values, parts of the main object is also subtracted. In the above show sequence, we see that T = 61 works very well.

3. Using the grayscale images (walk.bmp, bg[000-029].bmp) provided on the WWW site, perform background subtraction 2 using statistical distances. Experiment with thresholds. [5 pts]



T=2.3



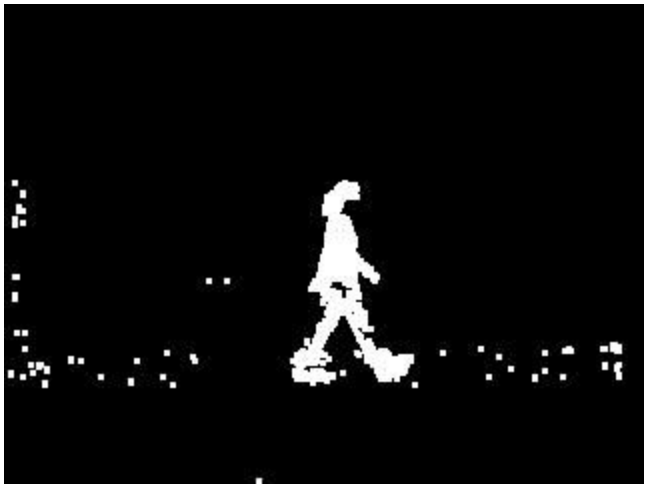
T=2.6

T=2.9

T=3.2

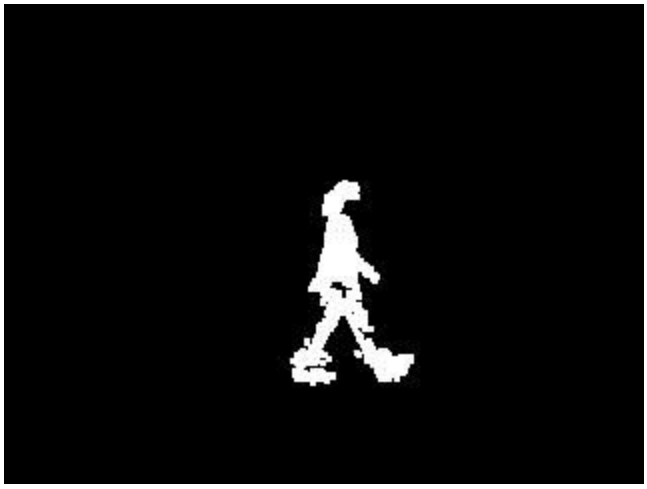
Varying threshold follows the same pattern as observed in the previous question.

4. Dilate the best binary image resulting from problem 3) using [1 pt]



Threshold T=2.6 was chosen as the best image as it produced the optimal image that subtracted the background and preserved the object.

5. Next perform a connected components algorithm, and keep only the largest region in L (save/display as an image). [1 pt]



6. Run the MATLAB canny edge detector, edge(Im,'canny'), on your image and display the default results. How does it compare? [2 pts]

```
% Abhinav Mahalingam
% CSE5524 - HW3
% 9/10/2017
%HW3.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 1
Im = double(imread('input/RandM.jpg'))/255;
grayIm = rgb2gray(Im);
colormap('gray');
level = 4;
[gaussianPyramid,laplacianPyramid] = createLaplacianPyramid(grayIm,level);
imshow(gaussianPyramid);
imwrite(gaussianPyramid, 'results/gaussianPyramid.jpg');
pause;
imshow(laplacianPyramid);
imwrite(laplacianPyramid, 'results/laplacianPyramid.jpg');
pause;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 2
Im = double(imread('input/walk.bmp'));
bgIm = double(imread('input/bg000.bmp'));
simple_backgroud_subtraction(Im,bgIm);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 3
bsIm = statistical_backgroud_subtraction(Im);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 4
d_bsIm = bwmorph(bsIm, 'dilate');
imagesc(d_bsIm);
imwrite(d_bsIm, 'results/dilatedIm.jpg');
pause;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 5
[L, num] = bwlabel(d_bsIm, 8);
hist_counts = histcounts(L,1:num);
max_count = max(hist_counts);
largest_comp = bwareaopen(L,max_count,8);
imagesc(largest_comp);
imwrite(largest_comp, 'results/largestComp.jpg');
```

blurAndSample.m

```
function reducedIm = blurAndSample(grayIm)
   Gm = createGaussianMask(0.4);
   gxIm = imfilter(grayIm, Gm, 'replicate');
   blurredIm = imfilter(gxIm', Gm, 'replicate')';
   dimen = size(blurredIm);
   reducedIm = blurredIm(1:2:dimen(1),1:2:dimen(2));
end
```

calculateNewDimesnsion.m

```
function new_dim = calculateNewDimesnsion(dim,level)
   const = 2^(level-1);
   new_dim = zeros(size(dim));
   for i = 1:2
      if mod(dim(i),const)==0
         new_dim(i)=dim(i)-const+1;
      else
         new_dim(i)=(floor(dim(i)/const)*const)+1;
      end
   end
end
```

createGaussianMask.m

```
function Gm = createGaussianMask(a)
   Gm = [0.25-(0.5*a),0.25,a,0.25,0.25-(0.5*a)];
end
```

simple_backgroud_subtraction.m

```
function [] = simple_backgroud_subtraction(Im,bgIm)
   diff = abs(Im-bgIm);
   for T = 1:10:70
      result = diff>T;
      imagesc(result);
      fname = sprintf('results/simp_bg_sub%d.jpg', T);
      imwrite(result, fname);
      pause;
   end
end
```

createLaplacianPyramid.m

```
% Consider original image as level 1
function [gaussianPyramid,laplacianPyramid] = createLaplacianPyramid(grayIm,level)
   new_dim = calculateNewDimesnsion(size(grayIm),level);
   actualIm = grayIm(1:new_dim(1),1:new_dim(2));
   gaussianPyramid = zeros(ceil(1.5*new_dim(1)),new_dim(2));
   laplacianPyramid = zeros(ceil(1.5*new_dim(1)),new_dim(2));
   gaussianPyramid(1:new_dim(1),1:new_dim(2)) = actualIm;
   g_row_offset = new_dim(1);
   g_col_offset = 0;
   l_row_offset = new_dim(1);
   l_col_offset = 0;
   for i = 2:level
      act_dim = size(actualIm);
      reducedIm = blurAndSample(actualIm);
      approxIm = imresize(reducedIm,act_dim,'bilinear');
      errorIm = actualIm-approxIm;
      red_dim = size(reducedIm);
     gaussianPyramid( g_row_offset+1 : g_row_offset+red_dim(1), g_col_offset+1 :
g_col_offset+red_dim(2) ) = reducedIm;
      g_col_offset = g_col_offset + red_dim(2);

      if i==2
         laplacianPyramid(1:act_dim(1),1:act_dim(2)) = errorIm;
      else
         laplacianPyramid(l_row_offset+1:l_row_offset+act_dim(1),l_col_offset+1:
l_col_offset+act_dim(2)) = errorIm;
         l_col_offset=l_col_offset+act_dim(2);
      end
    actualIm = reducedIm;
   end

laplacianPyramid(l_row_offset+1:l_row_offset+red_dim(1),l_col_offset+1:l_col_offset+red_dim
(2)) =
gaussianPyramid(l_row_offset+1:l_row_offset+red_dim(1),l_col_offset+1:l_col_offset+red_dim
(2));
End
```

statistical_backgroud_subtraction.m

```
function [bestBg] = statistical_backgroud_subtraction(Im)
   bgIm = zeros(240,320,30);
   for n=1:29
      bgIm(:,:,n) = double(imread(sprintf('input/bg%03d.bmp',n)));
   end
   avg_bg = mean(bgIm,3);
   std_bg = std(bgIm,1,3);
   diff = ((Im-avg_bg)./std_bg).^2;
   for T = 2:0.3:4
      result = diff>T^2;
      imagesc(result);
      fname = sprintf('results/stat_bg_sub%03f.jpg', T);
      imwrite(result, fname);
      pause;
   end
   bestBg = diff>2.6^2;
end
```