HW3.m

```
% Chandrasekar Swaminathan
% swaminathan.42@osu.edu
% CSE5524 - HW3
% 9/6/2016
mkdir 'out';
% Problem 1
% generating gaussian pyramid
% 4-Level guassian pyramid, assuming original image is level 1
pyramid_img = generate_gaussian_pyramid(rgb2gray(imread('images\\glados.jpg')), 4);
imagesc(pyramid_img), text(size(pyramid_img, 2) - 180, size(pyramid_img, 1) - 30, '4L Gaussian
pyramid', 'Color', 'Yellow', 'BackgroundColor', 'Black', 'EdgeColor', 'White'), axis('image'),
colormap('gray');
save_current_frame('out\\gaussian_level4.jpeg');
pause;
% 5-Level guassian pyramid, assuming original image is level 1
pyramid_img = generate_gaussian_pyramid(rgb2gray(imread('images\\glados.jpg')), 5);
imagesc(pyramid_img), text(size(pyramid_img, 2) - 180, size(pyramid_img, 1) - 15, '5L Gaussian
pyramid', 'Color', 'Yellow', 'BackgroundColor', 'Black', 'EdgeColor', 'White'), axis('image'),
colormap('gray');
save_current_frame('out\\gaussian_level5.jpeg');
pause;

% 4-level laplacian pyramid, assuming original image is level 1
generate_laplacian_pyramid(rgb2gray(imread('images\\glados.jpg')),4);

% expand image
level1 = imread('images\laplacian-error-image-level-1.jpeg');
level2 = imread('images\laplacian-error-image-level-2.jpeg');
level3 = imread('images\laplacian-error-image-level-3.jpeg');
level4 = imread('images\laplacian-image-level-4.jpeg');
image_so_far = expand_image(level4, level3);
image_so_far = expand_image(image_so_far, level2);
image_so_far = expand_image(image_so_far, level1);
imagesc(image_so_far), axis('image'), colormap('gray');
pause;

% Problem 2
background_subtraction_1;

% Problem 3
background_subtraction_2;

% Problem 4
dilated_image = dilate(read_indexed_bmp('images\\walk.bmp'));
```

```
imshow(dilated_image);
imwrite(dilated_image, 'out\DilatedImage.jpeg');
pause;

% Problem 5
[binary, gray] = largest_component();
imshow(binary);
pause;
imshow(gray);
imwrite(binary, 'out\LargestBinaryComponent.jpeg');
imwrite(gray, 'out\LargestGrayComponent.jpeg');
```

apply_separable_gaussian_filter.m

```
function [FilteredImage] = apply_separable_gaussian_filter(Image, sigma)
    size = 2 * ceil(3*sigma) + 1;
    Gx = fspecial('gaussian', [1 size], sigma);
    Gy = fspecial('gaussian', [size 1], sigma);
    FilteredImage = imfilter(imfilter(Image, Gx), Gy, 'replicate');
end
```

background_subtraction_1.m

```
function [] = background_subtraction_1 ()
    background = load_all_images;
    avg_background=mean(background,3);
    image = read_indexed_bmp('images\\walk.bmp');
    for T = 0.0:0.1:0.9
        diff = abs(image-avg_background) > T;
        imshow(diff);
        text(10, 20, sprintf('Threshold : %g', T), 'Color', 'Yellow', 'FontSize', 14, 'EdgeColor', 'white',
'BackgroundColor', 'Black');
        save_current_frame(sprintf('out\\bg-1-%g.jpg', T));
        pause;
    end
    close all;
end
```

```
background_subtraction_2.m
function [] = background_subtraction_2 ()
  image = read_indexed_bmp('images\\walk.bmp');
  mahalanobis_dist = mahalanobis(image);
  mkdir('out');
  for T = 0.0:0.1:3.0
    thresholded = mahalanobis_dist > (T^2);
    imshow(thresholded);
    text(10, 20, sprintf('Threshold : %g^2', T), 'Color', 'Yellow', 'FontSize', 14, 'EdgeColor', 'white',
'BackgroundColor', 'Black');
    save_current_frame(sprintf('out\\bg-2-%g.jpg', T));
    pause;
  end
  close all;
end

dilate.m
function [Dil] = dilate (image)
  mahalanobis_dist = mahalanobis(image);
  thresholded = mahalanobis_dist > (2.85^2);
  Dil = bwmorph(thresholded, 'dilate');
end

expand_image.m
function [expanded_image] = expand_image(image_so_far, next_level)
  expanded_image = next_level + imresize(image_so_far, 'bilinear', 'OutputSize', size(next_level),
'Antialiasing', false, 'Dither', false, 'Colormap', 'original');
end

generate_gaussian_pyramid.m
function [gaussian_pyramid] = generate_gaussian_pyramid(baseImage, levels)
  row_offset = 1;
  col_offset = 1;
  gaussian_pyramid = zeros(floor(size(baseImage,1)/2), floor(size(baseImage,2)/2));
  scaled_down_image=baseImage;
  for n=2:levels
    scaled_down_image = reduce_gaussian(scaled_down_image);
    nrows = size(scaled_down_image,1);
    ncols = size(scaled_down_image,2);
    gaussian_pyramid(row_offset:(nrows+row_offset-1), col_offset:(ncols+col_offset-1)) =
scaled_down_image;
    imwrite(scaled_down_image, sprintf('gaussian-pyramid-image-level-%d.png', n));
    if n == 2
      col_offset=col_offset+ncols;
    else
```

```
            row_offset=row_offset+nrows;
        end
    end
end
```

generate_laplacian_error.m

```
function [laplacian_error_image, next_level_gaussian] = generate_laplacian_error (image)
    reduced_image = reduce_gaussian(image);
    approx_image = imresize(reduced_image, 2, 'bilinear', 'OutputSize', size(image), 'Antialiasing',
false, 'Dither', false, 'Colormap', 'original');
    laplacian_error_image = image - approx_image;
    next_level_gaussian = reduced_image;
end
```

generate_laplacian_pyramid.m

```
function [] = generate_laplacian_pyramid (image, levels)
    next_level_gaussian_image = image;
    for n=1:levels-1
        [laplacian_error_image, next_level_gaussian_image] =
generate_laplacian_error(next_level_gaussian_image);
        imwrite(laplacian_error_image, sprintf('out\\laplacian-error-image-level-%d.jpeg', n));
        imagesc(laplacian_error_image), axis('image'), colormap('gray');
        pause;
    end
    imwrite(next_level_gaussian_image, sprintf('out\\laplacian-image-level-%d.jpg', levels));
    imagesc(next_level_gaussian_image), axis('image'), colormap('gray');
    pause;
end
```

largest_component.m

```
function [LargestComponentBinary, LargestComponentOrig] = largest_component ()
   orig_img = read_indexed_bmp('images\\walk.bmp');
   dilated_img = dilate(orig_img);
   [Label_img, num] = bwlabel(dilated_img, 8);
   num_of_hist_bins = num + 1;
%   get the frequency of each label in the labelled image.
   hist_counts = histcounts(Label_img, 1:num_of_hist_bins);
%   the most frequent label represents the largest component in the image
   most_frequent_label = find(hist_counts ==  max(hist_counts));
%   get the rows and columns containing the most frequent label
   [r,c] = find(Label_img == most_frequent_label);
   LargestComponentBinary = Label_img(min(r):max(r), min(c):max(c));
   LargestComponentOrig = orig_img(min(r):max(r), min(c):max(c));
end
```

load_all_images.m

```
function [Img] = load_all_images()
   Img = zeros(240, 320, 30);
   for n=1:29
      Img(:,:,n) = read_indexed_bmp(sprintf('images\\bg%03d.bmp', n));
   end
end
```

mahalanobis.m

```
function [Maha] = mahalanobis(image)
   background = load_all_images;
   avg_background=mean(background,3);
   std_dev_background = std(background, 1, 3);
   Maha = ((image-avg_background)./std_dev_background).^2;
end
```

read_indexed_bmp.m

```
function [image] = read_indexed_bmp (image_location)
   [image, map] = imread(image_location);
   image = rgb2gray(ind2rgb(image, map));
end
```

reduce_gaussian.m

```
function [reduced_image] = reduce_gaussian (img)
   gauss_filt = apply_separable_gaussian_filter(img, 0.4);
   dim = size(img);
   reduced_image = gauss_filt(1:2:dim(1), 1:2:dim(2));
end
```