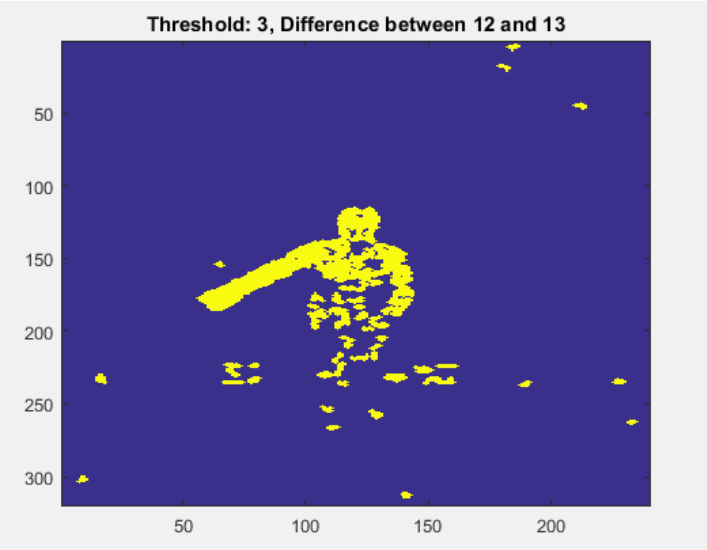
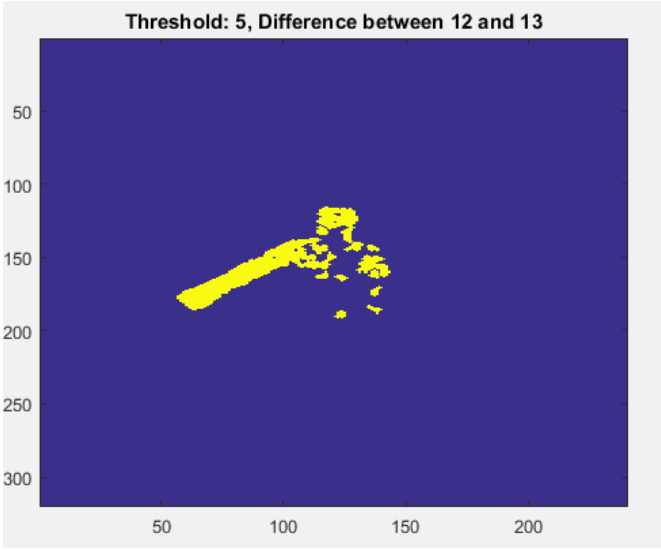
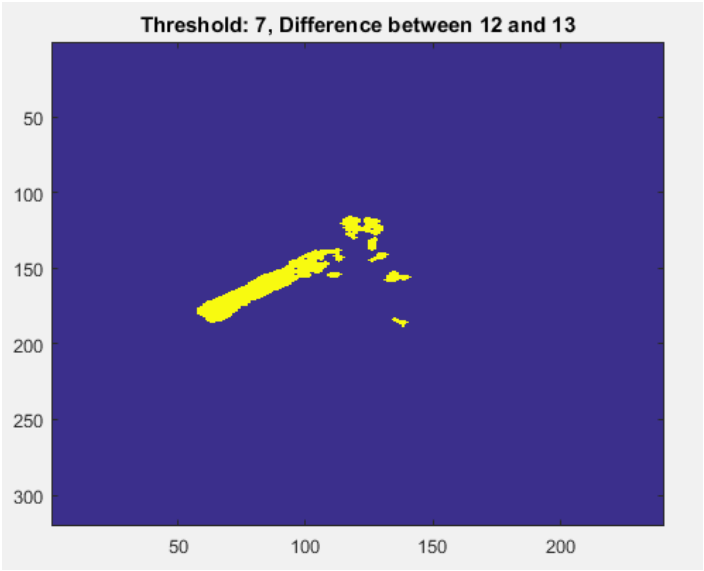
CSE 5524 - Homework #5 **9/27/2016**

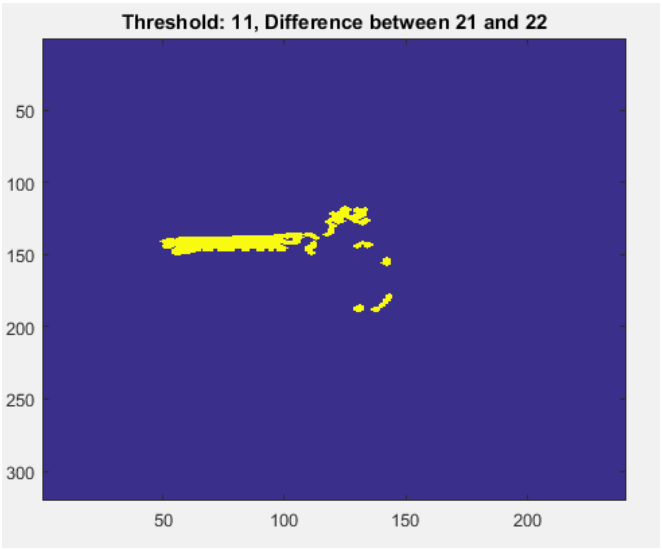
Chandrasekar Swaminathan (swaminathan.42)

1) Using the images (aerobic-[001-022].bmp) provided on the class WWW site, experiment with simple motion detection between consecutive frames using (abs) image differencing. Remove any tiny regions (e.g., use bwareaopen, median filtering, etc.). Experiment with different thresholds

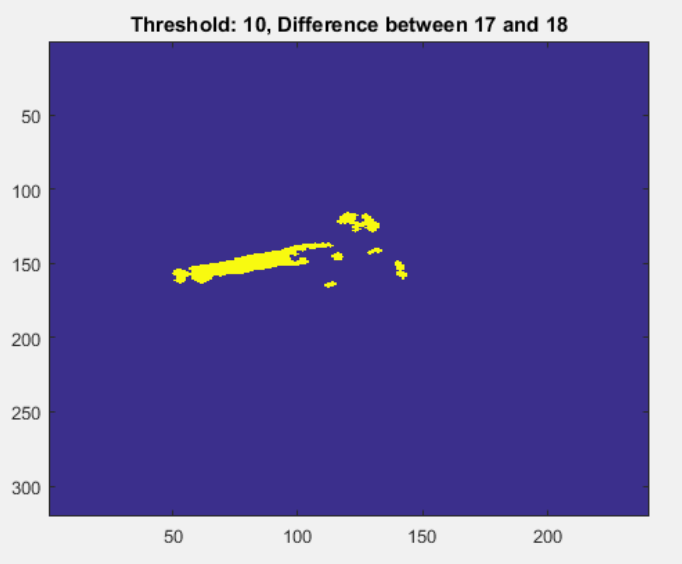


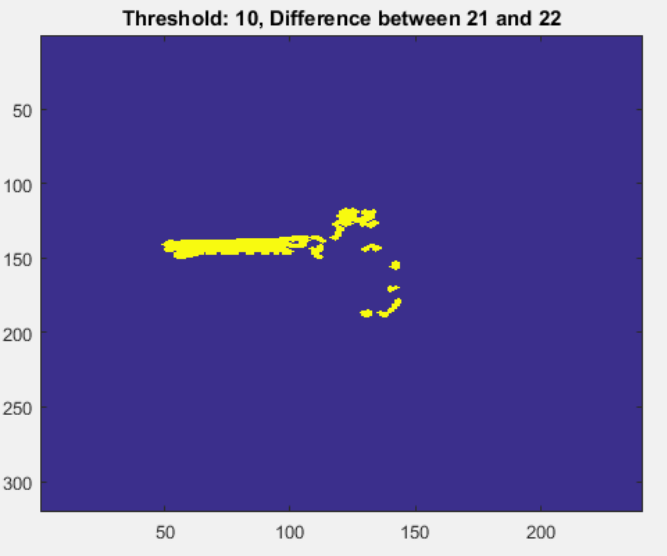






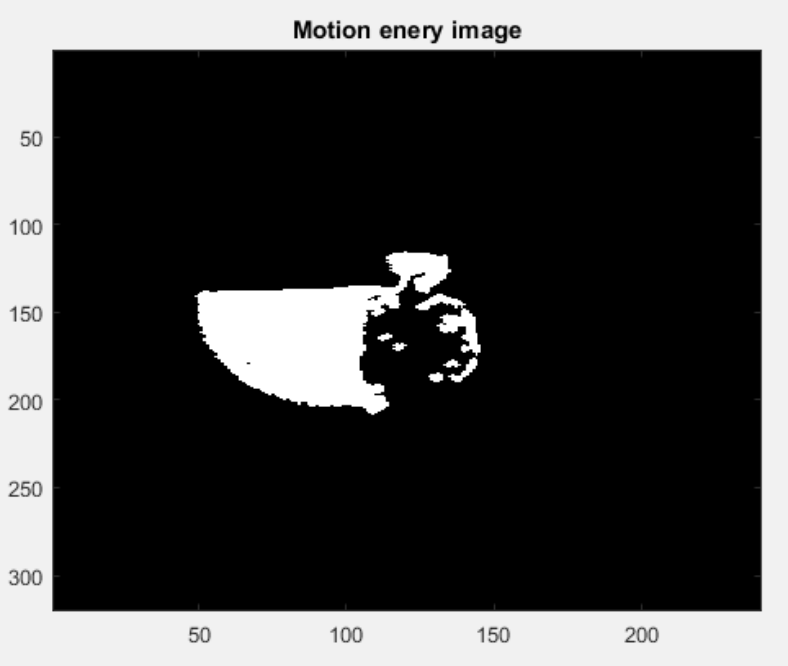
The image was first cleaned up using **medfilt2** and image difference was computed for every pair of images starting from image 1 up to image 22, resulting in 21 difference images. **bwareaopen** function was then applied on the difference images to remove small connected regions that were smaller than 5 pixels. These images were then dilated using the **imdilate** function to ensure that the arm of the person in the image appears as a single connected object.

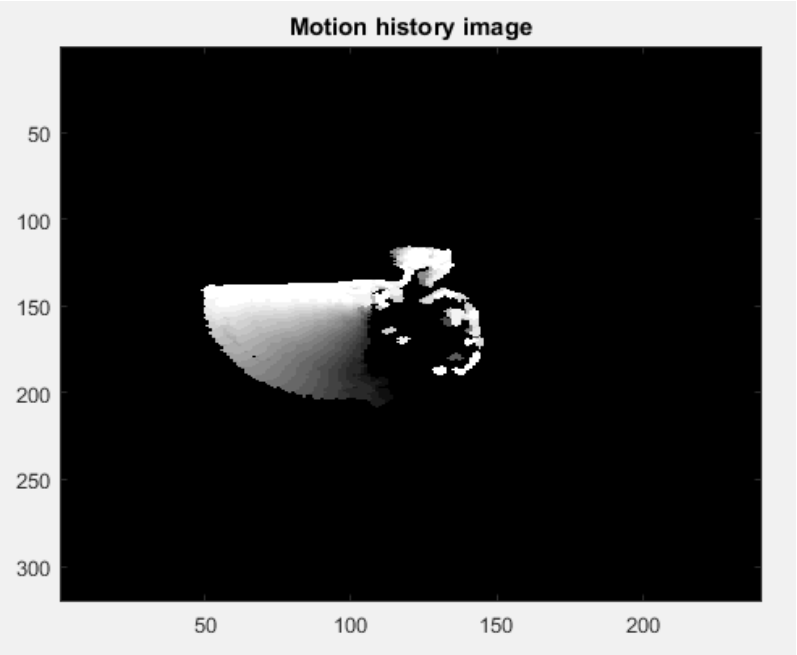




Several threshold were applied on the cleaned up difference image and it can be seen that the threshold value of 10 yields a good difference image, as it adequately captures the movement near the shoulder and neck region that starts around the 12^{th} and 13^{th} frame.

2) Compute an MEI and MHI on the image sequence (using your best motion differencing approach from problem #1 for each image pair i and i-1), simulating the MHI "timestamp" for each image pair using the larger (most current) of the image pair index values (i.e., use i, not i-1). The MEI/MHI duration should include all images of the sequence into the final template. Use imagesc to show your results. Compute the 7 similitude moments for the final MEI and the MHI (make sure to normalize the MEI and MHI values to be between 0-1 before computing the moments, i.e. divide by the max timestamp for this example).





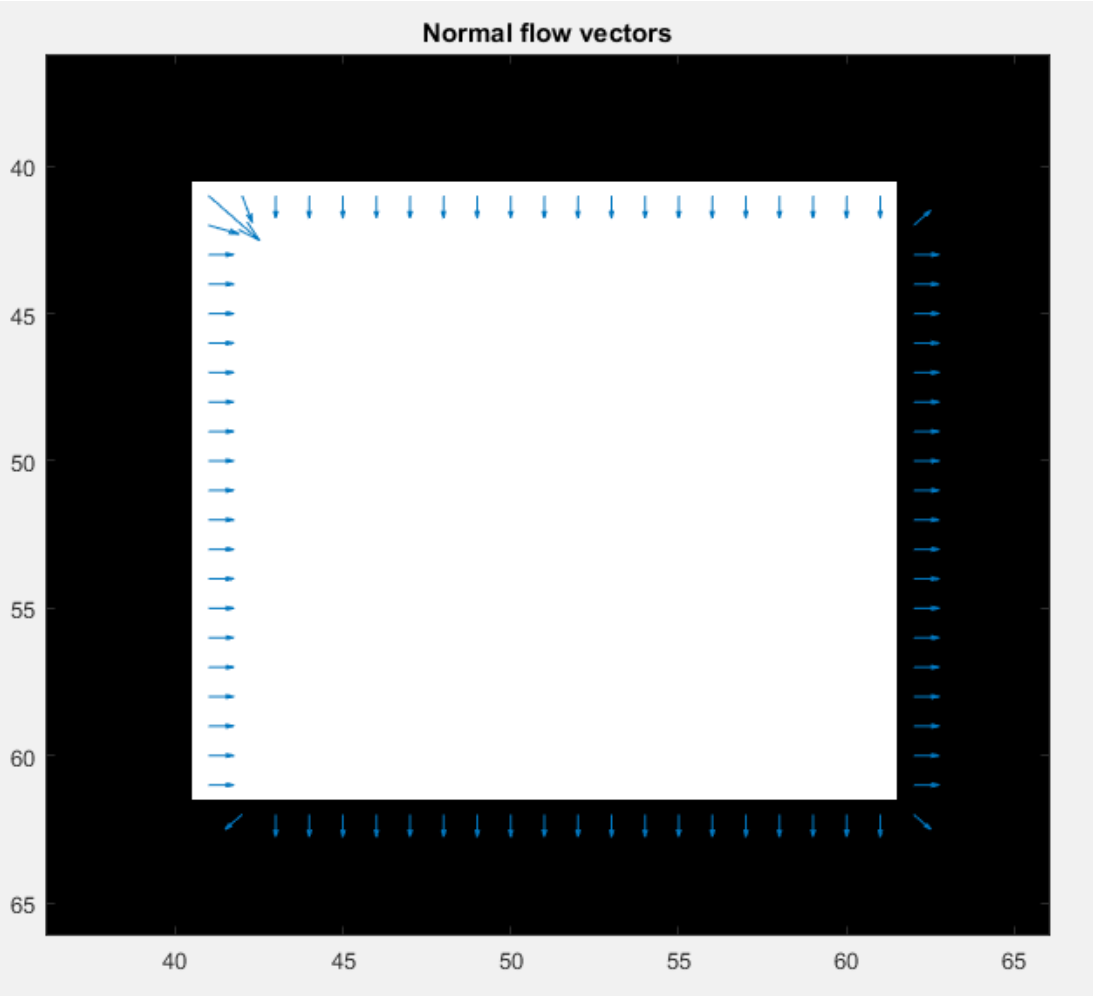
The motion history image shows the direction of motion in the image. We can see the intensity of pixels fading from the top to bottom, indicating that the most recent position of the object under motion is near the top and the oldest position is near the bottom. We can see that the history image also captures subtle movements near the head and body of the person in the image.

The similitude moments of both the history image and energy image were computed and are as follows:

motion_energy_image:
0.1075 0.0012 -0.0199 0.0145 0.1314 -0.0145 0.0194

motion_history_image:
0.1280 0.0156 -0.0434 0.0199 0.2437 -0.0122 0.0512

3) Create a 101x101 image with a black (0) background and a white (255) box of size 21x21 centered in the image. Create another new box image, but shift the box 1-pixel to the right and 1-pixel down. Compute the normal flow between the images. Use MATLAB's quiver function to draw the vector motions. (Make sure your gradient mask orientations/directions and the plot axes are consistent – examine 'axis(ij)'.) Make sure all masks are "correct" with proper scaling/normalization. Is the result what you expected?



The square at the center of the image moves one pixel to the right and one pixel down. The normal flow vectors were computed and are overlaid on top of the initial image. We can clearly see that the normal vectors point in the direction of the movement along each axes. The length of the arrows in the above image is a measure of the magnitude of movement along that direction. We can see that the length of the arrows is very close to 1.

Code

HW5.m

```
mkdir 'out';
% Problem 1
clear; close all;
num_of_images = 22;
img_width = 240;
img_height = 320;
Images = zeros(img_height, img_width, num_of_images);
base_differences = zeros(img_height, img_width, num_of_images-1);
thresh_differences = zeros(img_height, img_width, num_of_images-1);
for n=1:num_of_images
   Images (:, :, n) = medfilt2(double(imread(fullfile('input',sprintf('aerobic-%03d.bmp',n)))));
   if n > 1
      base_differences(:, :, n-1) = abs(Images(:, :, n) - Images(:, :, n-1));
   end
end
% cycle between various thresholds
for thresh=1:2:15
   for n=1:(num_of_images-1)
      imagesc(imdilate(bwareaopen(base_differences(:, :, n) > thresh, 5), strel('disk', 1)));
      title(sprintf('Threshold: %d, Difference between %d and %d', thresh, n, n+1));
      pause(0.01);
   end
   pause(0.1);
end

good_threshold = 10;
for n=1:size(base_differences,3)
   thresh_differences(:,:,n) = imdilate(bwareaopen(base_differences(:, :, n) > good_threshold, 5),
strel('disk', 1));
   imagesc(thresh_differences(:,:,n));
   title(sprintf('Threshold: %d, Difference between %d and %d', good_threshold, n, n+1));
   pause(0.01);
end
pause;

% Problem 2
motion_history=zeros(img_height, img_width, num_of_images - 1);
duration = num_of_images - 1; % same as the number of difference images.
for frame_number=1:duration
   % H(x, y, t) = duration if D(x,y,t) = 1
   current_frame_history = thresh_differences(:, :, frame_number) .* duration;
   if frame_number > 1
      for row=1:size(current_frame_history, 1)
         for col=1:size(current_frame_history, 2)
            if current_frame_history(row, col) == 0
```

```
                % H(x,y,t) = max(0, H(x,y,t-1) - 1) if D(x,y,t) = 0
                current_frame_history(row, col) = max(0, motion_history(row, col, frame_number - 1) -
1);
            end
          end
        end
    end
    motion_history(:, :, frame_number) = current_frame_history;
end
% normalizing the motion_history_image
motion_history_image = motion_history(:, :, duration)./duration;
% thresholding mhi to get mhe
motion_energy_image = motion_history_image > 0;
imagesc(motion_energy_image);
title('Motion enery image');
colormap('gray');
save_current_frame(fullfile('out', 'motion_energy_image.bmp'));
pause;
imagesc(motion_history_image);
title('Motion history image');
colormap('gray');
save_current_frame(fullfile('out', 'motion_history_image.bmp'));
pause;

mei_sim_moments = similitudeMoments(motion_energy_image);
disp(mei_sim_moments);
mhi_sim_moments = similitudeMoments(motion_history_image);
disp(mhi_sim_moments);

% Problem 3
base_image = zeros(101, 101);
base_image(41:61, 41:61) = ones(21, 21);
shifted_image = zeros(101, 101);
shifted_image(42:62, 42:62) = ones(21, 21);
% fx fy gradient filters assuming the top left corner is origin and y is
% +ve as we go down from top left to bottom left
fx_filter = [-1 0 1; -2 0 2; -1 0 1]./8;
fy_filter = [-1 -2 -1; 0 0 0; 1 2 1]./8;

ft = shifted_image - base_image;
fx = imfilter(shifted_image, fx_filter, 'replicate');
fy = imfilter(shifted_image, fy_filter, 'replicate');
imagesc(fx);
title('After applying fx filter');
pause;
imagesc(fy);
title('After applying fy filter');
```

```
pause;
normal_flow_vectors = zeros(2, 101);
normal_flow_start_points = zeros(2, 101);
number_of_vectors = 0;
for x=1:size(fx,2)
   for y=1:size(fx,1)
      if fx(y,x) == 0 && fy(y,x) == 0
         continue
      end
      number_of_vectors = number_of_vectors + 1;
      c = sqrt(fx(y,x)^2 + fy(y,x)^2);
      magnitude = -ft(y,x) ./ c;
      normal_flow_vectors(1, number_of_vectors) = magnitude * fx(y,x) ./ c;
      normal_flow_vectors(2, number_of_vectors) = magnitude * fy(y,x) ./ c;
      normal_flow_start_points(1,number_of_vectors) = x;
      normal_flow_start_points(2,number_of_vectors) = y;
   end
end
imagesc(base_image);
hold on;
quiver(normal_flow_start_points(1, :), normal_flow_start_points(2, :), normal_flow_vectors(1, :),
normal_flow_vectors(2, :));
% changing axis of the plot so that origin is at top-left
title('Normal flow vectors');
axis('ij');
hold off;
```

computeCentroid.m

```
function [centroidx, centroidy] = computeCentroid(image)
   zeroOrderMoment = computeMoment(true, image, NaN, NaN, 0, 0);
   firstOrderXMoment = computeMoment(true, image, NaN, NaN, 1, 0);
   firstOrderYMoment = computeMoment(true, image, NaN, NaN, 0, 1);
   centroidx = firstOrderXMoment/zeroOrderMoment;
   centroidy = firstOrderYMoment/zeroOrderMoment;
end
```

computeMoment.m

```
function [momentValue] = computeMoment(isSpatial, Img, centroidX, centroidY, momentX,
momentY)
   result = 0.0;
   Img = double(Img);
   for x=1:size(Img, 2)
      for y=1:size(Img, 1)
         if isSpatial
            result = result + (((x)^momentX * (y)^momentY)*Img(y, x));
         else
            result = result + (((x-centroidX)^momentX * (y-centroidY)^momentY)*Img(y, x));
```

```
        end
      end
    end
    momentValue = result;
end
```

similitudeMoments.m

```
function [N] = similitudeMoments(image)
    [centroidX, centroidY] = computeCentroid(image);
    momentsToConsider = [ 0 2; 0 3; 1 1; 1 2; 2 0; 2 1; 3 0];
    zeroOrderMoment = computeMoment(false, image, centroidX, centroidY, 0, 0);
    N = zeros (1, size(momentsToConsider, 1));
    for row = 1:size(momentsToConsider, 1)
        p = momentsToConsider(row, 1);
        q = momentsToConsider(row, 2);
        momentValue = computeMoment(false, image, centroidX, centroidY, p ,q);
        N(1, row) = momentValue / (zeroOrderMoment^(((p+q)/2) + 1));
    end
end
```

save_current_frame.m

```
function [] = save_current_frame (outputFileName)
% saves whatever is displayed in the window to the given file. useful when
% there is dynamic text rendered on top of the image and that should also
% be saved to the file
  frame = getframe;
  imwrite(frame.cdata, outputFileName);
end
```