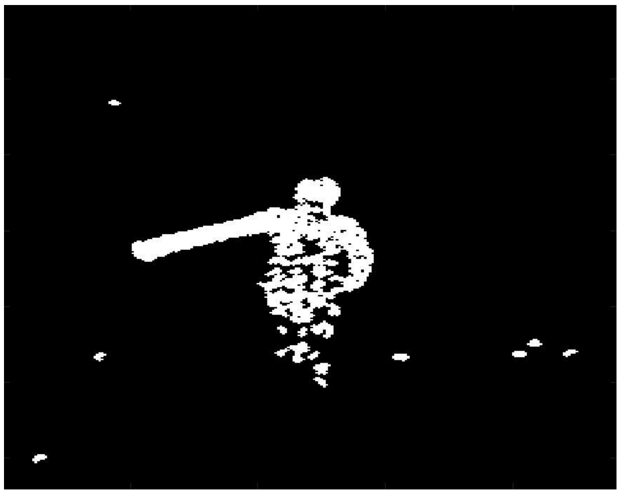
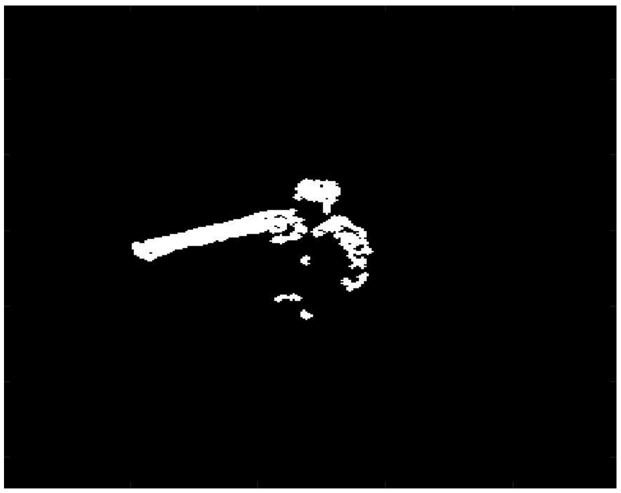
Computer Vision for HCI Abhinav Mahalingam (.15)

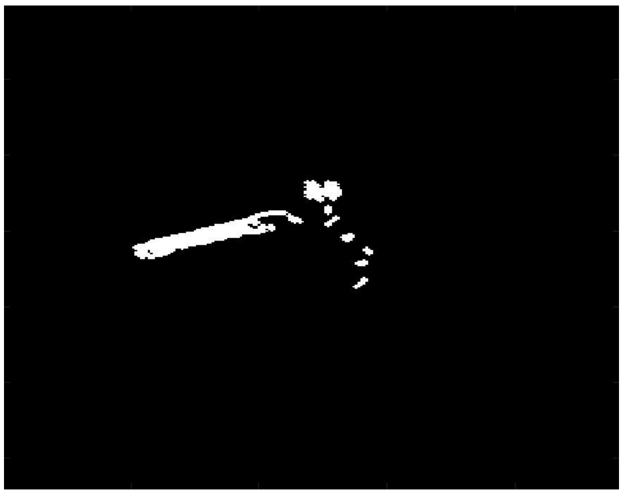
CSE 5524

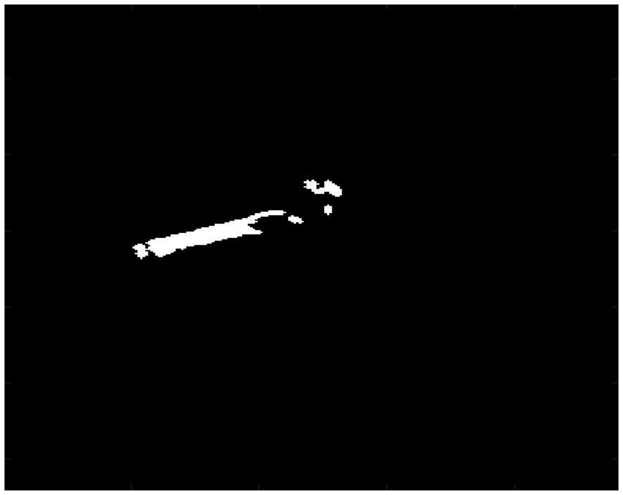
Homework #5

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. [2 pts]


T = 3

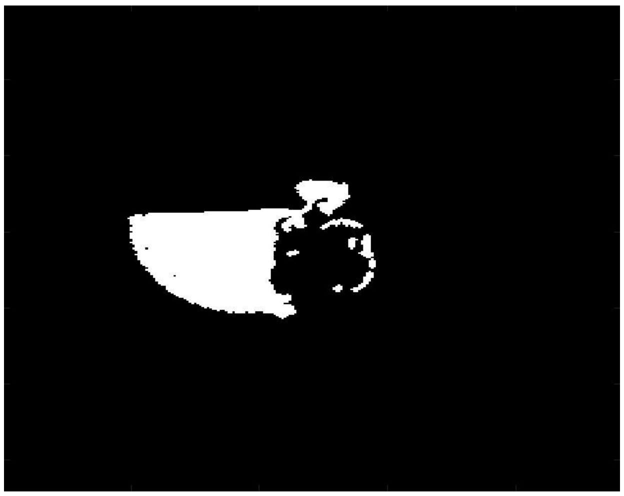

T = 5

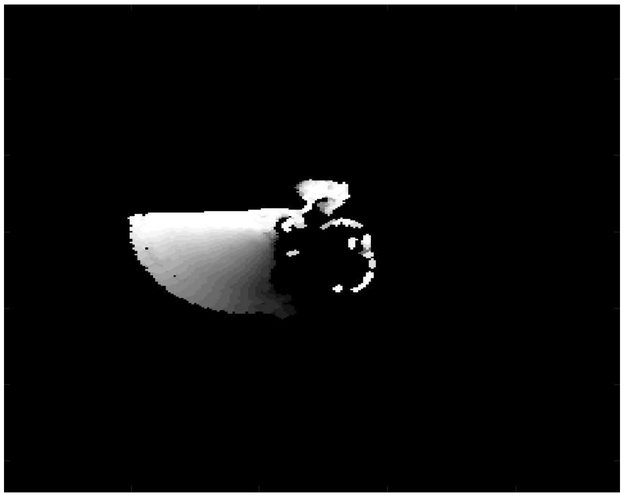

T = 9


T = 11

The input images were filtered using **medfilt2** and any region with less than 5 pixels were removed using **bwareaopen**. The **Imdilate** function was used to ensure that the arm appeared as one connected region. It can be seen that as the threshold value increases, the noise detected decreases but if the value of threshold is too high, then some movements such as shoulder and head movement is not detected. The threshold value of 11 gives a good result.

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 using the given formula of max[0, (*i*-1.0)/21.0] for this example). [4 pts]


Motion Energy Image

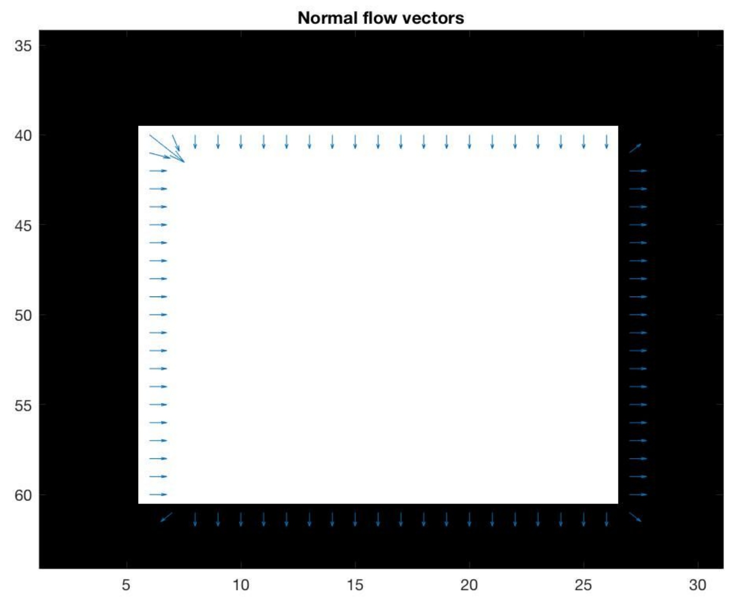

Motion History Image

We can observe the regions where motion occurred from the Motion Energy image and the Motion History image gives us the direction of motion, with the brightest region indicating the current position and the faded region indicating the past position.

```
Motion Energy Image :  0.1142    0.0005   -0.0205    0.0179    0.1256   -0.0169    0.0200
Motion History Image :  0.1386    0.0172   -0.0477    0.0250    0.2298   -0.0198    0.0563
```

3. Create a 101x101 image with a black (0) background and a white (255) box of size 21x21, placing the upper-left corner at pixel (col=6, row=40). 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 on the image (call imagesc, then 'hold on', lastly call quiver). (Make sure your gradient mask orientations/directions and the plot axes are consistent.) Make sure all masks are "correct" with proper scaling/normalization. Is the result what you expected? [5 pts]

The white box moved one pixel to the right and one pixel down. The image below was drawn by displaying the original image and overlaying the normal flow vectors on it. We can see that the normal flow vectors point in the direction of motion and the magnitude of all the arrows is very close to 1.



```
% Abhinav Mahalingam
% CSE5524 - HW5
% 9/23/2017
% HW5.m
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 1
colormap('gray');
img_width = 240;
img_height = 320;
num_images = 22;
Im = zeros(img_height,img_width,num_images);
Im_diff = zeros(img_height,img_width,num_images-1);
Thresh_Im_diff = zeros(img_height,img_width,num_images-1);
for i = 1:num_images
   Im(:,:,i)=medfilt2(double(imread(sprintf('input/aerobic-%03d.bmp',i))));
   if i > 1
      Im_diff(:,:,i-1) = abs(Im(:,:,i)- Im(:,:,i-1));
   end
end
for t = 1:2:15
   for i = 1:(num_images-1)
```

```
        imagesc(imdilate(bwareaopen(Im_diff(:,:,i)>t,5),strel('disk',1)));
        title(sprintf('Threshold: %d, Difference between Im %d and %d',t,i,i+1));
        if i==16
            saveFrame(sprintf('results/t%d.jpg',t));
        end
        pause(0.01);
    end
end
% t=11 gives the best threshold for the aerobic sequence
t = 11;
for i = 1:(num_images-1)
        Thresh_Im_diff(:,:,i) = imdilate(bwareaopen(Im_diff(:,:,i)>t,5),strel('disk',1));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 2
motion_history = zeros(img_height,img_width,num_images-1);
duration = num_images;
for i =1:duration-1
    cur_frame_history = Thresh_Im_diff(:,:,i).*(duration);
    if i>1
        for r = 1:size(cur_frame_history,1)
            for c = 1:size(cur_frame_history,2)
                if cur_frame_history(r,c)==0
                    cur_frame_history(r,c) = max(0,motion_history(r,c,i-1)-1);
                end
            end
        end
    end
    motion_history(:,:,i) = cur_frame_history;
end
MHI = motion_history(:,:,duration-1)./duration;
MEI = MHI > 0;
imagesc(MEI);
title('MEI');
saveFrame('results/MEI.jpg');
pause;
imagesc(MHI);
title('MHI');
```

```
saveFrame('results/MHI.jpg');
pause;
mei_sim_moments = similitudeMoments(MEI);
disp(mei_sim_moments);
mhi_sim_moments = similitudeMoments(MHI);
disp(mhi_sim_moments);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 3
colormap('gray');
base_image = zeros(101,101);
base_image(40:60,6:26) = ones(21,21);
shifted_image = zeros(101,101);
shifted_image(41:61,7:27) = ones(21,21);
fx_filter = [-1 0 1; -2 0 2; -1 0 1]./8;
fy_filter = [-1 -2 -1; 0 0 0; 1 2 1]./8;
fx = imfilter(shifted_image,fx_filter,'replicate');
fy = imfilter(shifted_image,fy_filter,'replicate');
ft = shifted_image - base_image;
normal_flow_vectors = zeros(2,168);
normal_flow_start_points = zeros(2,168);
num_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
      num_of_vectors =num_of_vectors + 1;
      c = sqrt(fx(y,x)^2 + fy(y,x)^2);
      magnitude = -ft(y,x)./c;
      normal_flow_vectors(1,num_of_vectors) = magnitude * fx(y,x)./c;
      normal_flow_vectors(2,num_of_vectors) = magnitude * fy(y,x)./c;
      normal_flow_start_points(1,num_of_vectors) = x;
      normal_flow_start_points(2,num_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,:),n
ormal_flow_vectors(2,:));
title('Normal flow vectors');
hold off;

computeMoments.m
function [m] = computeMoments(Img,x,y,p,q)
  m=0;
  for r = 1:size(Img,1)
    for c = 1:size(Img,2)
       m = m + (Img(r,c)*((c-x)^p)*((r-y)^q));
    end
  end
end

getCentroid.m
function [centroidX,centroidY] = getCentroid(Img)
   m00 = computeMoments(Img,0,0,0,0);
   m10 = computeMoments(Img,0,0,1,0);
   m01 = computeMoments(Img,0,0,0,1);
   centroidX=m10/m00;
   centroidY=m01/m00;
End

similitudeMoments.m
function[N] = similitudeMoments(Img)
   [centroidX,centroidY] = getCentroid(Img);
   m00 = computeMoments(Img,0,0,0,0);
   moments = [0 2; 0 3; 1 1; 1 2; 2 0; 2 1; 3 0];
   N = zeros(1,size(moments,1));
   for r=1:size(moments,1)
      p = moments(r,1);
      q = moments(r,2);
      momentVal = computeMoments(Img,centroidX,centroidY,p,q);
      N(1,r) = momentVal/(m00^(((p+q)/2)+1));
   end
end
```