

# EE435: Biometric Signal Processing

## Project 7: Motion Detection and Tracking in Video

Assigned: Fri 3/04/2011

Due: Fri 3/11/2011

This project builds on Project 06, in that it involves video processing and will have you create an output movie (.avi) file.

Motion segmentation typically relies on comparing frames of a video stream to some reference frame where no motion has occurred. What defines motion is up to the users, but simple noise in an image should not be considered motion. This project is an extension of the class activity where you imported several registered images into MATLAB and computed the difference between successive frames as a means of motion detection.

Obtain the [ee435\\_proj07.avi](#) video file from Short Term File Sharing (Division of Engineering and Weapons→Electrical and Computer Engineering Department→EE435). Write a MATLAB program that will take this video, analyze it, and write out a new video file that clearly shows detected motion that is present between frames. The new video file will consist of the old video with detected motion superimposed, meaning (for this project) that the new video file has a red box (2 pixel wide edges) around detected motion in EACH frame where there is motion detected. The video that is written out should run at the same speed (frames per second) and have the same dimensions as the original AVI file (i.e., 240 x 360 x 3, 170 frames).

In order to detect motion, you should base your code on the pseudocode contained on the last page of this project assignment.

Some useful functions (not a comprehensive list):

*aviinfo*—provides pertinent information about the contents of an AVI file

*aviread*—allows you to read in an AVI file into an AVI object

*avifile*—allows you to create an AVI object

*movie2avi*—allows you to write out an AVI file from a movie object

*regionprops*—determines the properties of a binary object, such as bounding box, centroid, etc.

If you use the *aviinfo* function to obtain information about this file, you'd see:

```
>> b=aviinfo('ee435_proj07.avi')  
  
b =  
    Filename: 'ee435_proj07.avi'  
    FileSize: 44070400  
    FileModDate: '19-Feb-2007 08:21:45'  
    NumFrames: 170  
    FramesPerSecond: 30  
    Width: 360  
    Height: 240  
    ImageType: 'truecolor'  
    VideoCompression: 'none'  
    Quality: 0  
    NumColormapEntries: 0
```

This shows that there are 170 frames (`b.NumFrames = 170`), each frame is 360 cols (`b.Width`) by 240 rows (`b.Height`), is stored as 30 frames per second (`b.FramesPerSecond = 30`), and has not been compressed.

If you use the *aviread* function to read in a movie object

```
>> a=aviread('ee435_proj07.avi')  
a =  
    1x170 struct array with fields:  
        cdata  
        colormap
```

Recall from Project 06 that in the movie structure, each frame is stored with a cdata component and a colormap component in the movie structure. The cdata component is the actual color image frame data, and the colormap component contains the colormap for each frame, but is NULL ([]) for true color images. To get to any of the individual image frames and view it after reading in the avi file, for example the 37<sup>th</sup> frame, you would type commands like:

```
>> fr = a(37).cdata; figure(1),imshow(fr)
```

## I. DrawBoundingBox Function

Write a function that will perform the following:

1. **DrawBoundingBox:** Takes an input color image and outputs a copy of the same image with a red rectangle drawn somewhere within it, based on the input parameters, or if the input is grayscale, will draw a white rectangle. For color images, it sets the red component values where the rectangle should be to 255, AND sets the green and blue components to 0 in the same locations. For grayscale images, it sets the rectangle locations to 255.

Usage:  $y = \text{DrawBoundingBox}(x, BB)$

### Inputs:

|           |   |
|-----------|---|
| <b>x</b>  | The input color or grayscale image .  |
| <b>BB</b> | The bounding box vector obtained from the <i>regionprops</i> function, which is of the form:<br>[ <b>top_left_col_num</b> <b>top_left_row_num</b> <b>num_cols</b> <b>num_rows</b> ] |
|           | <b>top_left_col_num</b> col coordinate of the top left corner of the bounding box   |
|           | <b>top_left_row_num</b> row coordinate of the top left corner of the bounding box   |
|           | <b>cols</b> number of cols in the bounding box  |
|           | <b>rows</b> number of rows in the bounding box  |

### Output:

|          |   |
|----------|---|
| <b>y</b> | The output <u>color</u> image w/a red box drawn in it as defined by the bounding box (if the input was color), or an output grayscale image with a white box drawn around the motion (if the input video was grayscale) or $y = []$ for an error. |
|----------|---|

### Error checking:

This program should check to ensure that the input image is either color or grayscale using only the depth (that is, the number of planes of data should be 1 or 3). If not, return  $y = []$  and output an appropriate error message.

**Note:** This function should **ONLY** be called when the input image contains detected motion; if there was no motion detected, the bounding box **DOES NOT** exist and this function **SHOULD NOT** be called.

## II. Motion Detection

Apply the steps listed in the introduction above to process the “ee435\_proj07.avi” video for motion detection. Write out your result into a new avi file using *movie2avi*, with NO compression. Use a file name that contains the group members’ last name(s), such as: “Marzougui-Wang.avi.” Demonstrate that your avi file displays the motion detection using Windows Media Player, and have the professor initial here: \_\_\_\_\_.

**Submit your m-files from Parts I and II, and demonstrate the avi file you created for Part II to the professor using Windows Media Player.**

## Pseudocode for Processing "ee435\_proj04.avi"

```
% read in the video
a=aviread('ee435_proj07.avi')
al=a; % will write processed video back into "al" frame-by-frame

%compute a background image to compare all frames to...for example,
% take average of 1st 5-10 frames. For the background, there should be no
% apparent motion in the frame.
%
% You can use a for loop, but convert the frames you use to grayscale (using
% using rgb2gray.
bkgd= ...
for k=1:170
    k
    % grab current frame
    test = a(k).cdata;

    % convert current frame to gray
    test = ...

    % compute absolute difference between this frame and background
    test = ...

    % threshold the absolute difference
    test = ...

    % erode to get rid of most of the noise...you'll need to create
    % a structuring element.
    test = ...

    % dilate-use the same structuring element
    test = ...

    % convert to double for regionprops function
    test=double(test);

    % check of any motion detected (i.e., any ones in the binary image?)
    if sum(test(:)) > 0
        % note: don't bother using bwlabel...treat the binary image as one big
        % object that you can draw a bounding box around.
        rp=regionprops(test,'BoundingBox','Centroid');

        % remember: for the next step, DON'T CALL DrawBoundingBox if no motion is
        % detected.
        al(k).cdata=DrawBoundingBox(a(k).cdata,rp.BoundingBox);
    end

    % display each frame's result as you process it
    imshow(al(k).cdata,[],title(['k = ' num2str(k)]));
end
% check out the movie you created
movie(al,1,30) % show it 1 time, 30 frames/sec
% for fun, try this:
movie(al,-1,30);

% write out your result as an avi video file (use your own names)
movie2avi(al,'Lado-Turner.avi','FPS',30, 'COMPRESSION', 'none');
```