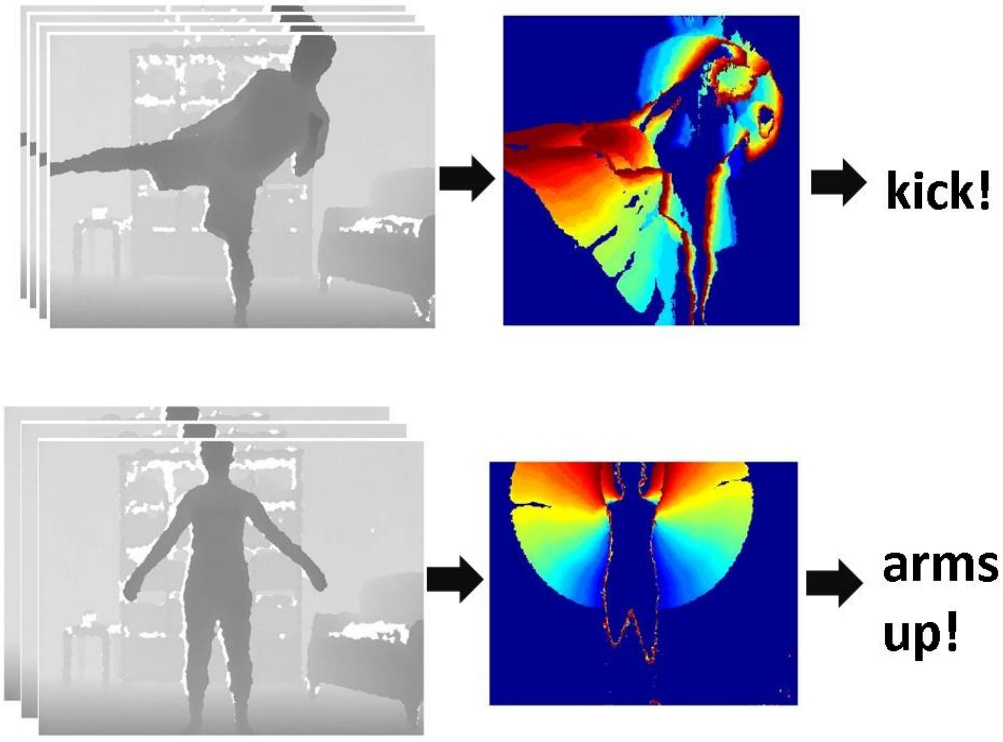
CS 376 Computer Vision
Problem Set 5
Out: Monday, April 25,
Due: Sunday, May 8, 11:59 PM

Please see the end of this document for submission instructions. Visit us during office hours to discuss any questions on the assignment. Or, if sending a question via email, please submit to cv-spring2011@cs.utexas.edu with CS376 in the subject line, and please be specific.

I. Programming problem: Human action recognition [100 points]



For this problem, you will implement an action recognition method using "motion history images". Given a video sequence, the goal is to predict which of a set of actions the subject is performing.

The basic idea is to use a sequence of depth images to segment out the foreground person, and then for each sequence, compute its motion history image (MHI). This MHI serves as a "temporal template" of the action performed, and can be further compressed into a set of 7 Hu moments. Once the Hu moments have been computed for all labeled training examples, we can categorize a novel test example by using nearest neighbor classification.

Video data

You can access the video data here: **/v/filer4b/v26q010/pset5data/**. There are 5 directories, each of which contains 4 sequences for one of the action categories. The 5 action categories are: "botharms", "crouch", "leftarmup", "punch", "righkick". Each directory under any one of these 5 main directories contains the frames for a single sequence. For example, /v/filer4b/v26q010/pset5data/punch/punch-p1-1/ contains one sequence of frames for "punch".

The data are stored as .pgm images. Each pgm is a grayscale image, where the intensity is proportional to depth in the scene. Note that the image frames are named sequentially, so that if you use Matlab's "`dir`" command to gather all pgm's in a directory and then loop over the image list, they will be in the correct order. **See the provided script "`readingDataExample.m`"** in /v/filer4b/v26q010/pset5data for an example of how to loop over the videos and read in the files.

The data takes about 530 M of disk space. You can point to the image files directly in your code, rather than copy the data to your home directory.

Approach

The main steps are as follows:

- Load the depth map pgms in a given sequence, and perform background subtraction to identify pixels on the foreground person. You should choose reasonable threshold(s) on depth based on examining an example or two. Global parameter settings should be sufficient for this data.

- From frame to frame in the sequence, compute the binary foreground silhouette difference image $D(x,y,t)$.

- Use all difference images in a τ-frame sequence to compute its Motion History Image, H_τ:

$$H_\tau(x,y,t) = \begin{cases} \tau & \text{if } D(x,y,t) = 1 \\ \max\ (0, H_\tau(x,y,t-1) - 1) & \\ \text{otherwise} & \end{cases}$$

  where t varies from 1 to τ.

- Normalize the Motion History Image (MHI) by the maximum value within it.

- Use the MHI to compute a 7-dimensional vector containing the 7 *Hu moments*. This vector is the final representation for the entire video sequence, and describes the global "shape" of the temporal template in a translation- and rotation-invariant manner.

- Having computed a descriptor vector for each video sequence, evaluate the nearest neighbor classification accuracy using *leave-one-out cross-validation*. That is, let every instance serve as a "test" case in turn, and classify it using the remaining instances.

- For the nearest neighbor classifier, use the normalized Euclidean distance (i.e., where the distance per dimension is normalized according to the sample data's variance).

- Evaluate the results over all sequences based on the mean recognition rate per class and the *confusion matrix*.

See the paper "The Representation and Recognition of Action Using Temporal Templates" by J. Davis and A. Bobick for more background on computing the MHI. For additional background on the properties of Hu moments, see "Visual Pattern Recognition by Moment Invariants" by M.-K. Hu. Both are linked from the course page.

What to implement, display, and discuss in the writeup

Write code for each of the following (along with any helper functions you find useful), and in your pdf writeup report on the results, briefly explain, and show images where appropriate.

1. **[30 points]** Write a function `computeMHI.m` that takes a directory name for a sequence, and returns the Motion History Image:

```
function [H] = computeMHI(directoryName)
```

In some other script, apply this function to compute Motion History Images for all the data, and display three examples in a figure in the pdf, titled with the action category each one belongs to. (You will need to debug your background subtraction procedure to get this working.)

2. **[15 points]** Write a function `huMoments.m` that takes a Motion History Image matrix and returns the 7-dimensional Hu moments vector:

```
function [moments] = huMoments(H)
```

3. **[20 points]** Write a function `predictAction.m` that predicts the label of a test sequence using nearest neighbor classification:

```
function [predictedLabel] =
predictAction(testMoments, trainMoments, trainLabels)
```

where `predictedLabel` is an integer from 1 to 5 denoting the predicted action label, `testMoments` is a 7-dimensional Hu moment descriptor representing the test sequence, `trainMoments` is an Nx7 matrix of Hu moment descriptors for N training instances, and `trainLabels` is an Nx1 vector denoting the action category labels for all N training instances. Use the normalized Euclidean distance.

4. **[20 points]** Write a script `showNearestMHIs.m` that displays the top `K` most similar Motion History Images to an input test example, based on the normalized Euclidean distance between their associated Hu moment descriptors. (Note that you *display* MHIs but still refer to distance in terms of the videos' Hu moment vectors.) In the pdf writeup, display the results for two selected test examples, for `K=4`.

5. **[15 points]** Finally, write a script `classifyAllActions.m` that performs leave-one-out cross validation on all the provided videos to determine the overall nearest neighbor recognition performance. This script should report the mean recognition rate per class, and display a 5 x 5 confusion matrix. In your writeup, discuss the performance and the most confused classes.

Useful Matlab functions: `imagesc, std, sort, dir`

II. OPTIONAL : Extra credit (up to 20 points total)

- Using ideas from any previous lectures, enhance the approach to try and improve the recognition results. For example, you might incorporate a different classifier, distance function, or descriptor. You might exploit the depth map for more than simply background subtraction, enhance the silhouette computation,… Explain clearly how you've extended the method, and report on the results. Explain in what ways things change relative to your implementation for Section I, and why.

Submission instructions: what to hand in

Electronically:

- Your well-documented Matlab code .m files.
- A pdf file containing
  - Your name and CS login ID at the top.
  - A brief explanation of your implementation strategy for Section I: paragraph or two describing in English how you implemented the programs.
  - Your image results and accompanying explanations for Section I.
  - (optional): Any results and descriptions for extra credit. For max credit explain what you implemented and also interpret the results.

Submit all the above with one call to turnin:

1. **`>> turnin --submit shalini pset5`** `file.pdf codeFileXYZ.m codeFileABC.m etc.`

Hardcopy: Also drop a hardcopy in the CS homework dropbox in PAI 5.38. Attach a cover page with the course number CS376 and your name, to make it easy to find in the dropbox. Please save paper by concatenating shorter functions into a single page before printing.