Instructions

1. Answer sheets must be submitted on Scholar. Hard copies will not be accepted.

2. Please submit your answer sheet containing the written answers in a file named: FirstName_LastName_PS5.pdf.

3. Please submit your code and input/output images in a zip file named: FirstName_LastName_PS5.zip. Please do not create subdirectories within the main directory.

4. You may collaborate with other students. However, you need to write and implement your own solutions. Please list the names of students you discussed the assignment with.

5. For the implementation questions, make sure your code is bug-free and works out of the box. Please be sure to submit all main and helper functions. Be sure to not include absolute paths. Points will be deducted if your code does not run out of the box.

6. Display plots in your answer sheet wherever required. Points will be deducted for not following this protocol.

1 Programming problem [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 labelled training examples, we can categorize a novel test example by using nearest neighbour classification.

**Video data** You can access the video data here (unzipped data occupies about 530MB of disk space): https://filebox.ece.vt.edu/~F13ECE5554/resources/PS5_material/PS5_Data.zip

There are 5 directories within the zip file, each of which contains 4 sequences for one of the action categories. The 5 action categories are: *botharms*, *crouch*, *leftarmup*, *punch*, *rightkick*. Each directory under any one of these 5 main directories contains the frames for a single sequence. For example, *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` for an example of how to loop over the videos and read in the files.

**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 $\tau$-frame sequence to compute its Motion History Image, $H_\tau$:

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

where $t$ varies from 1 to $\tau$.

- 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 neighbour 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 neighbour 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 (available here). For additional background on the properties of Hu moments, see Visual Pattern Recognition by Moment Invariants by M. K. Hu (available here).

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 write-up report on the results, briefly explain, and show images where appropriate. Your code must access the depth maps from sub-folders, named after each action, in your current working directory (i.e. unzip the provided data in your current working directory).

1. **30 points.** Write a function computeMHI.m that takes a directory name for a sequence, and returns the Motion History Image:
   
   ```matlab
   function [H] = computeMHI(directoryName)
   ...
   end
   ```

   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.) Please submit the Motion History Images of all the sequences in a file called allMHIs.mat. This file should contain a variable called allMHIs which is a matrix of size MxNx20.

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

   Please submit the Hu moments vectors of all the sequences in a file called huVectors.mat. This file should contain a variable called huVectors which is a matrix of size 20x7.

3. **20 pts.** Write a function predictAction.m that predicts the label of a test sequence using nearest neighbour classification:
   
   ```matlab
   function [predictedLabel] = predictAction(testMoments, trainMoments, trainLabels)
   ...
   end
   ```

   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 neighbour recognition performance. This script should report the mean recognition rate per class, and display a 5 x 5 confusion matrix. In your write-up, discuss the performance and the most confused classes.
2. **OPTIONAL: Extra credit (max 20 points)**

1. 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 1, and why.

---

This assignment is adapted from PS5 of Kristen Grauman’s CS 376: Computer Vision at UT Austin.