ECE 4984/5554: Computer Vision, Fall 2015

PS5

Christopher Mobley

Due: Monday, 02 DEC 15

1 Programming problems [100 points]

What to implement 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).

30 points. Write a function *computeMHI.m(py)* 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.) Please submit the Motion History Images of all the sequences in a file called allMHIs.mat(npy). This file should contain a variable called *allMHIs* which is a matrix of size *MxNx20*.

Figures 1 through 3 show motion history images returned by computeMHI() for directories botharms-up-p1-1, crouch-p1-1, and rightkick-p1-1 respectfully.

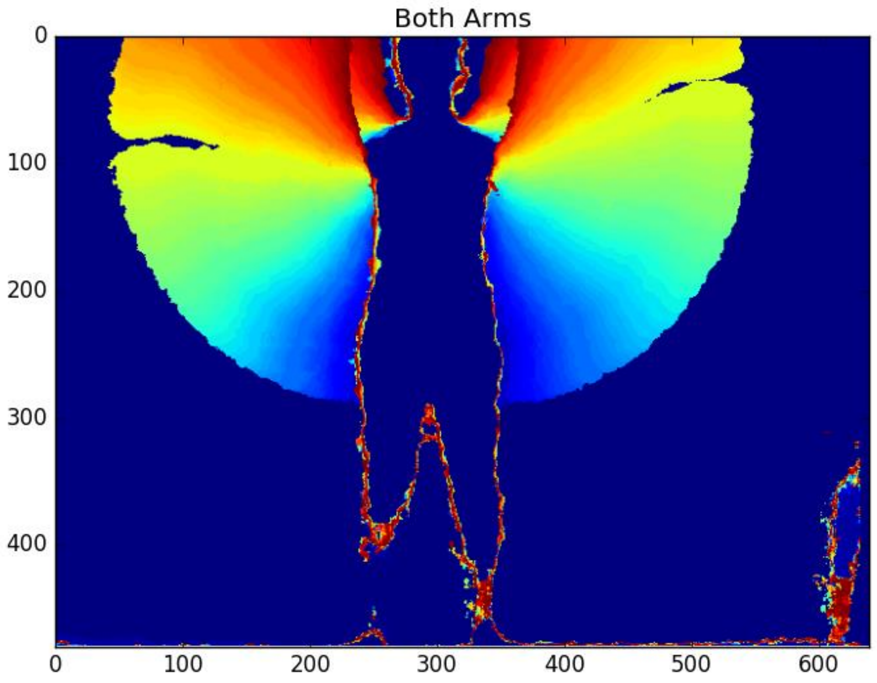

Figure 1. Both Arms MHI

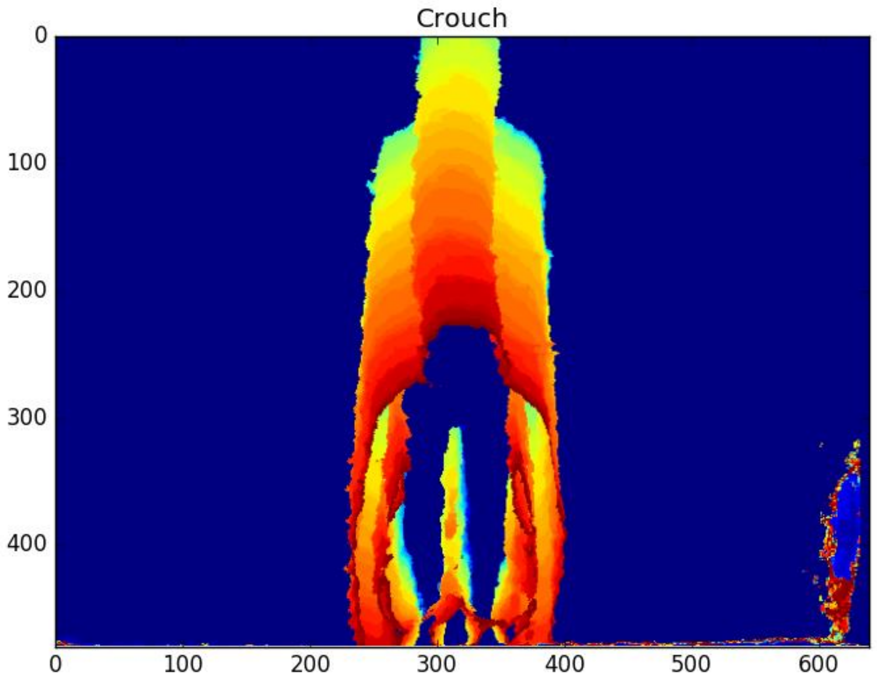

Figure 2. Crouch MHI

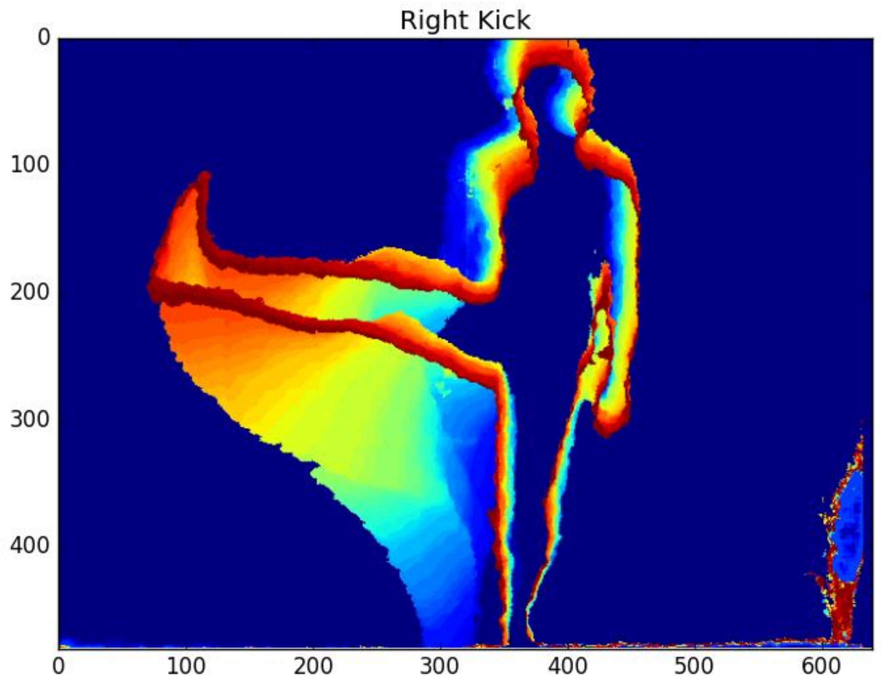

Figure 3. Right Kick MHI

15 points. Write a function *huMoments.m(py)* that takes a Motion History Image matrix and returns the 7-dimensional Hu moments vector:

function [moments] = huMoments(H)

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

20 pts. Write a function *predictAction.m(py)* 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.

20 points. Write a script *showNearestMHIs.m(py)* 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.

Figures 4 through 8 shows the botharms-up-p1-1 MHI, as well as its four nearest neighbors calculated by normalized Euclidean distance in *showNearestMHIs.py*. While Figures 9 through 13 show the rightkick-p1-1 MHI, as well as its four nearest neighbors calculated by normalized Euclidean distance in *showNearestMHIs.py*.

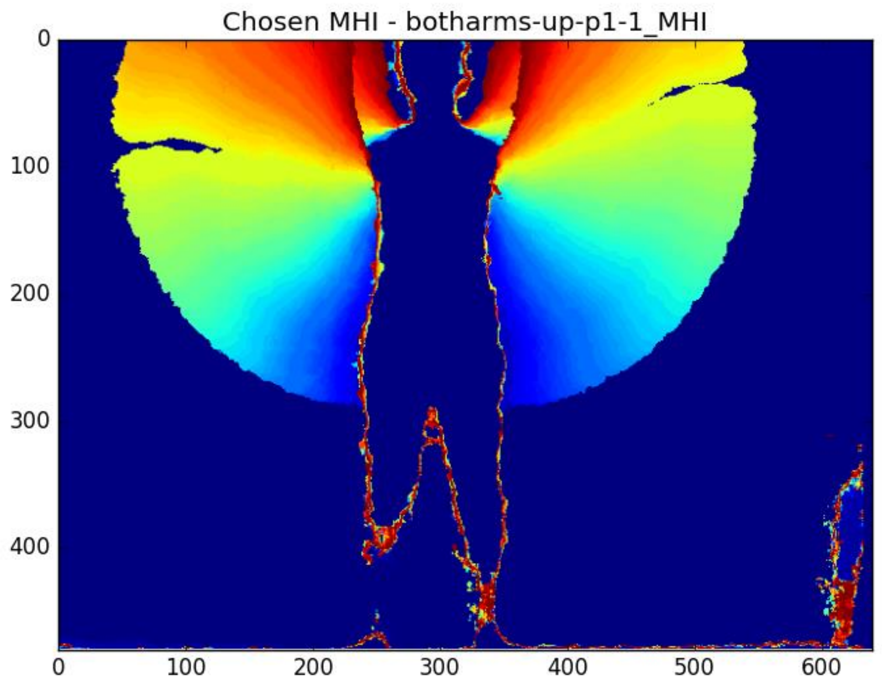

Figure 4. Chosen MHI – Both Arms

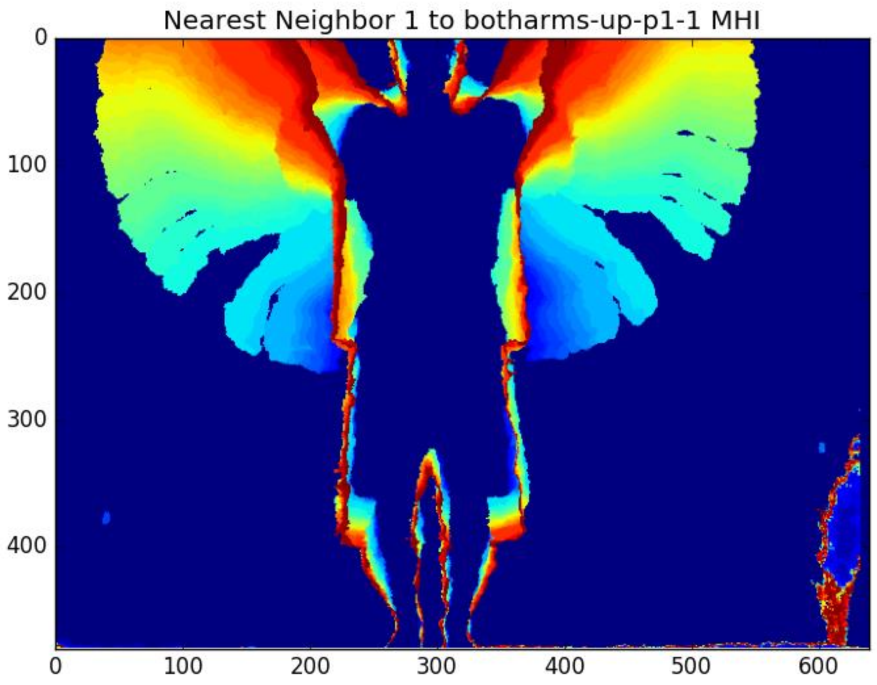

Figure 5. Nearest Neighbor 1 to Both Arms MHI

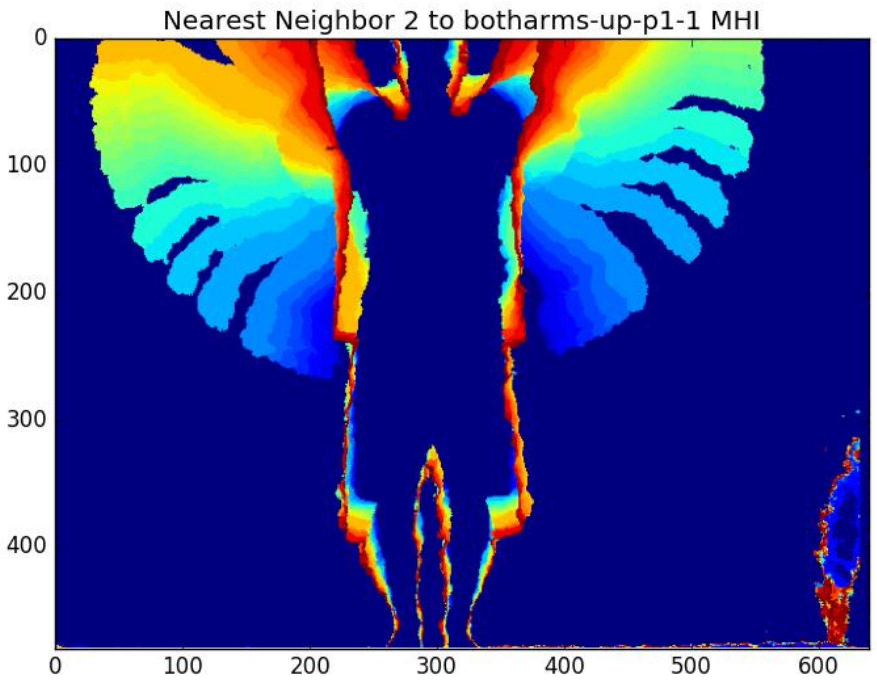

Figure 6. Nearest Neighbor 2 to Both Arms MHI

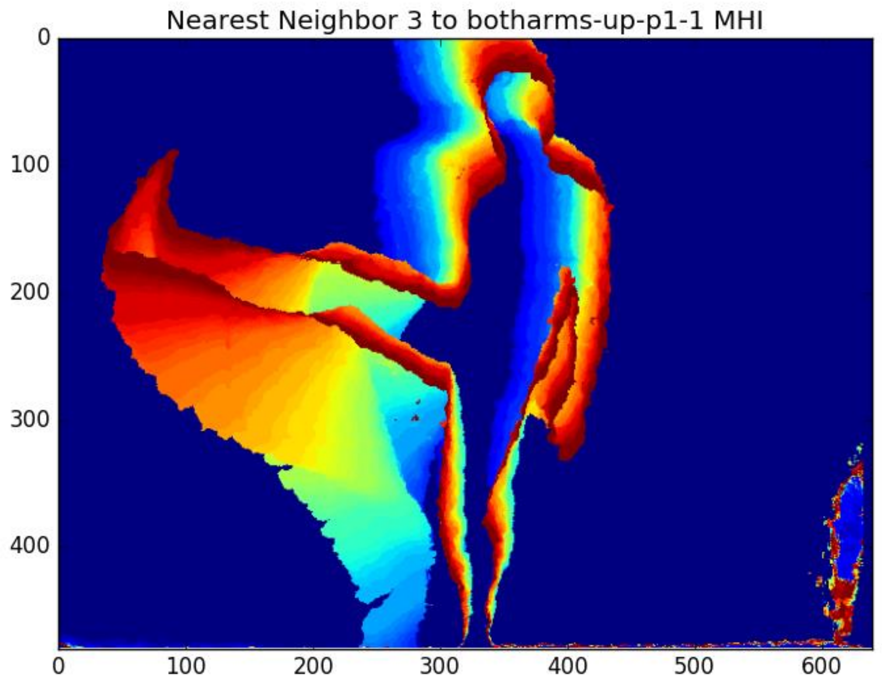

Figure 7. Nearest Neighbor 3 to Both Arms MHI

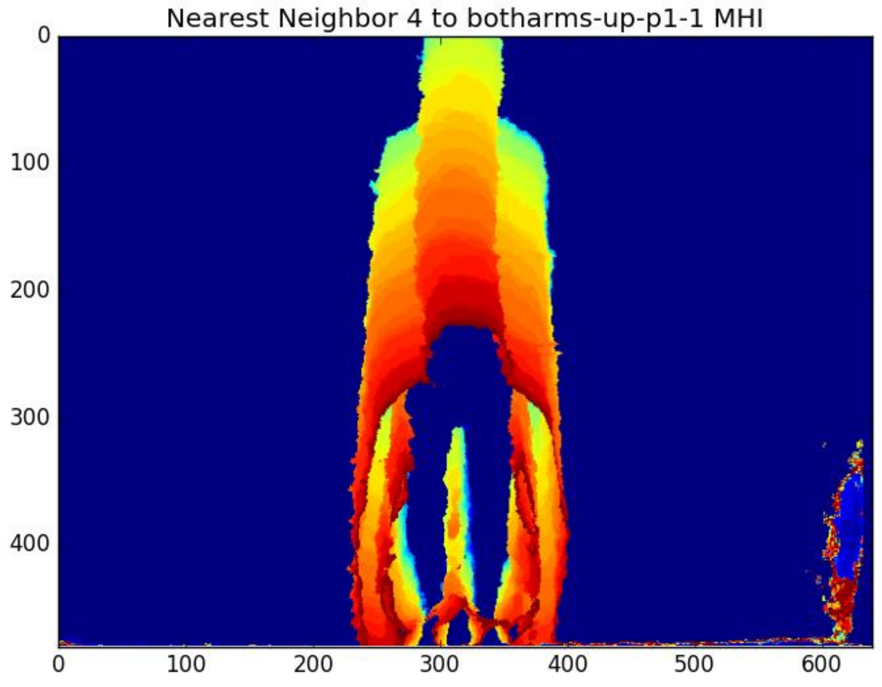

Figure 8. Nearest Neighbor 4 to Both Arms MHI

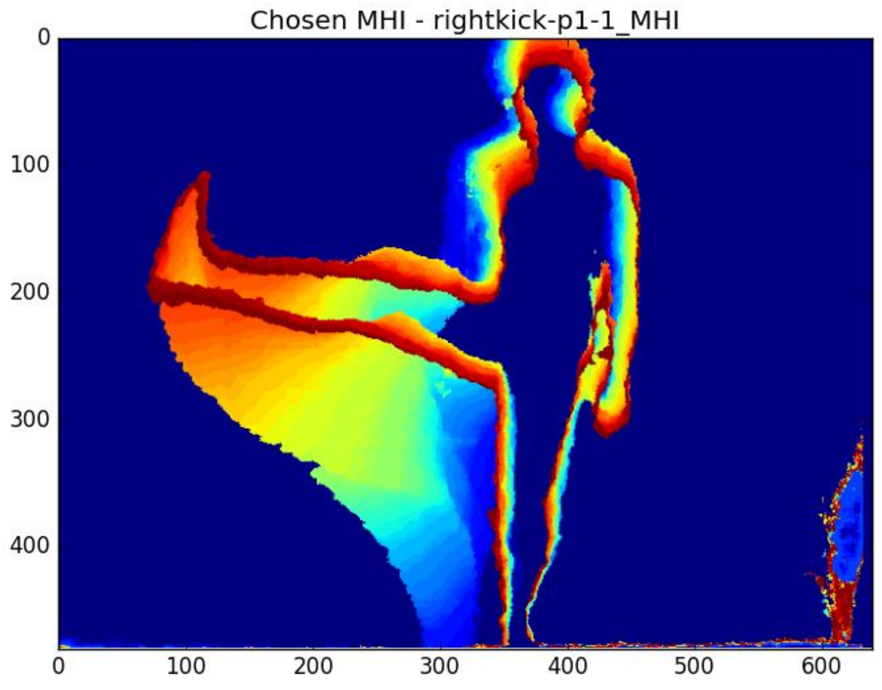

Figure 9. Chosen MHI – Right Kick

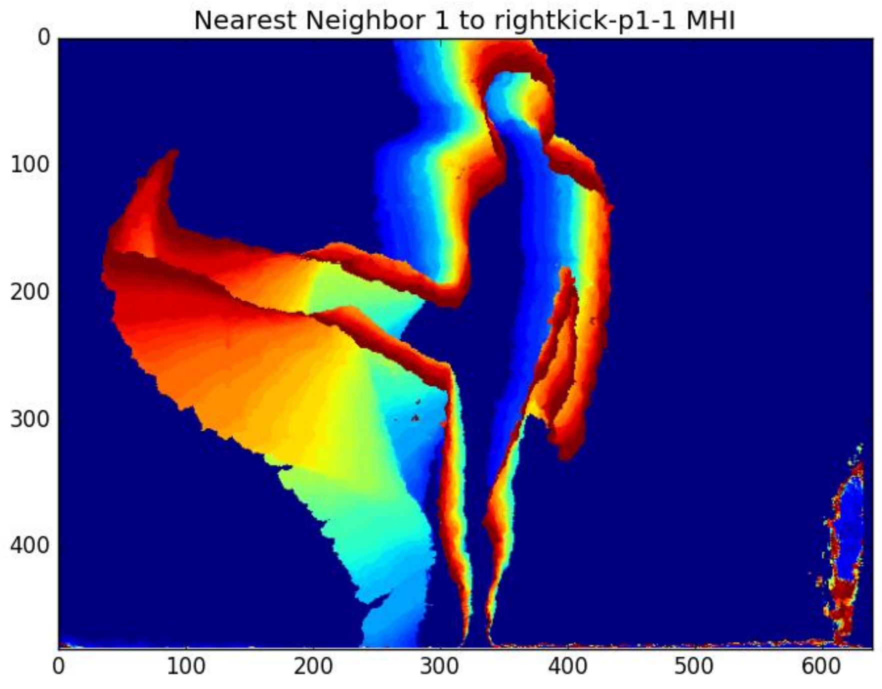

Figure 10. Nearest Neighbor 1 to Right Kick MHI

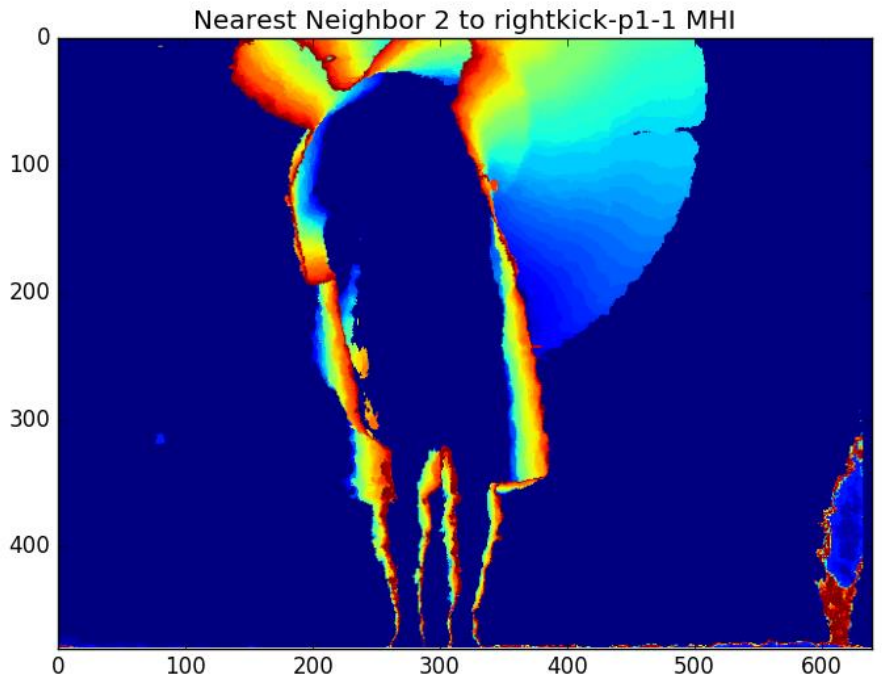

Figure 11. Nearest Neighbor 2 to Right Kick MHI

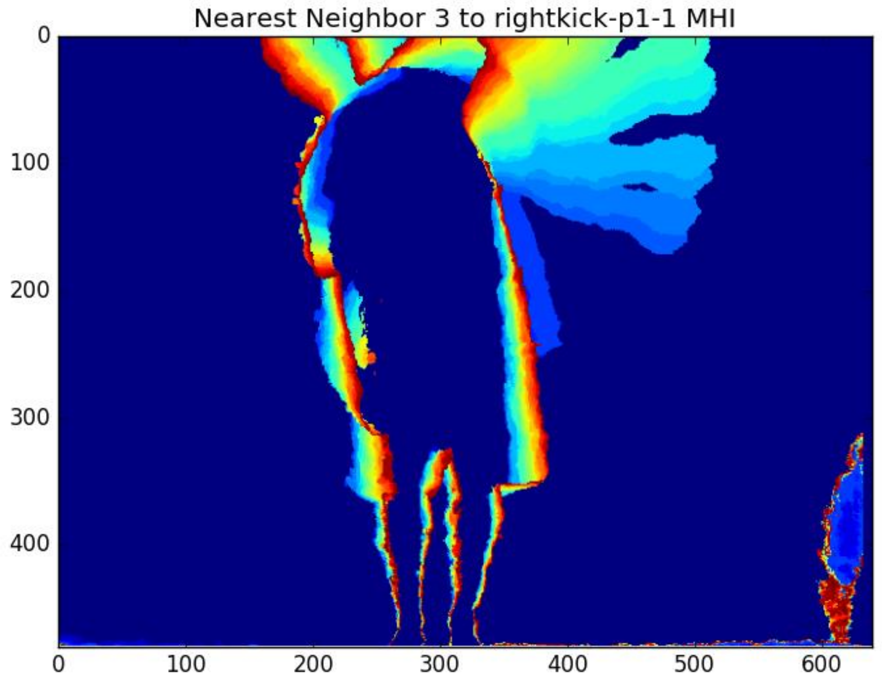

Figure 12. Nearest Neighbor 3 to Right Kick MHI

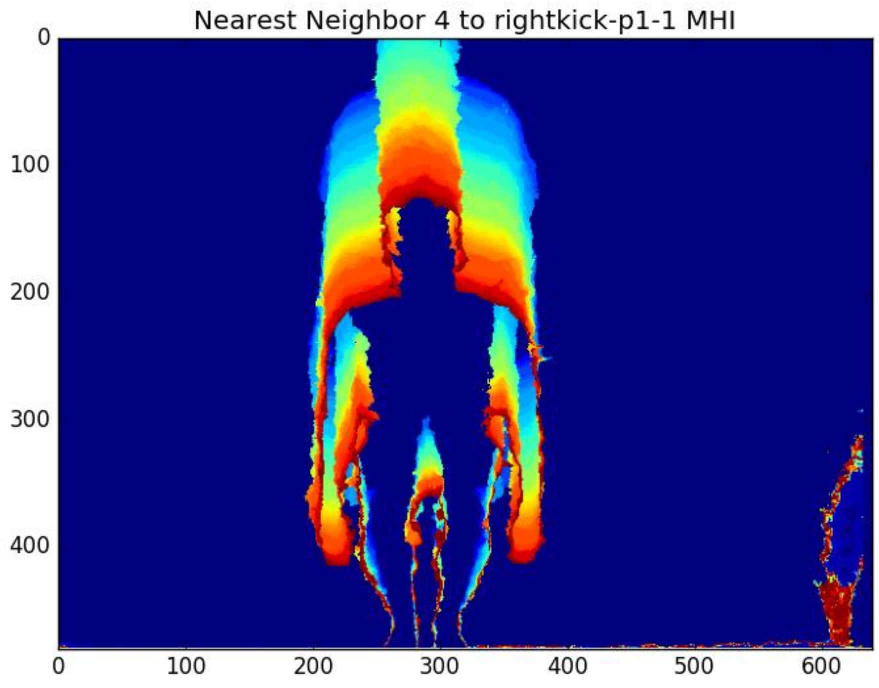

Figure 13. Nearest Neighbor 4 to Right Kick MHI

15 points. Finally, write a script *classifyAllActions.m(py)* 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.

Tables 1 and 2 show the mean recognition rate and confusion matrix respectively. Leftarmup while performing with a recognition rate of 100%, was the most confused class among the other classes, dropping the mean recognition rate of Crouch and Punch by 25% and Rightkick by 50%. However, this make sense since leftarmup MHI has a singular arch which was present in each of the failure MHIs. The normalized Euclidean distance proved to be a decent algorithm for labeling the MHI. However, for improved result one should look at the efficiency of k-mean or other machine learning algorithms like SVMs or Decision Trees.

Table 1. Mean recognition rate

	Mean Recognition Rate
Botharms	1
Crouch	0.5
Leftarmup	1
Punch	0.75
Rightkick	0.5

Table 2. Confusion matrix

	Botharms	**Crouch**	**Leftarmup**	**Punch**	**Rightkick**
Botharms	4	0	0	0	0
Crouch	0	2	1	1	0
Leftarmup	0	0	4	0	0
Punch	0	0	1	3	0
Rightkick	0	0	2	0	2

Note: I discussed this assignment with Murat Ambarkutuk and Orson Lin