CS 6476: Computer Vision, Fall 2019

PS5

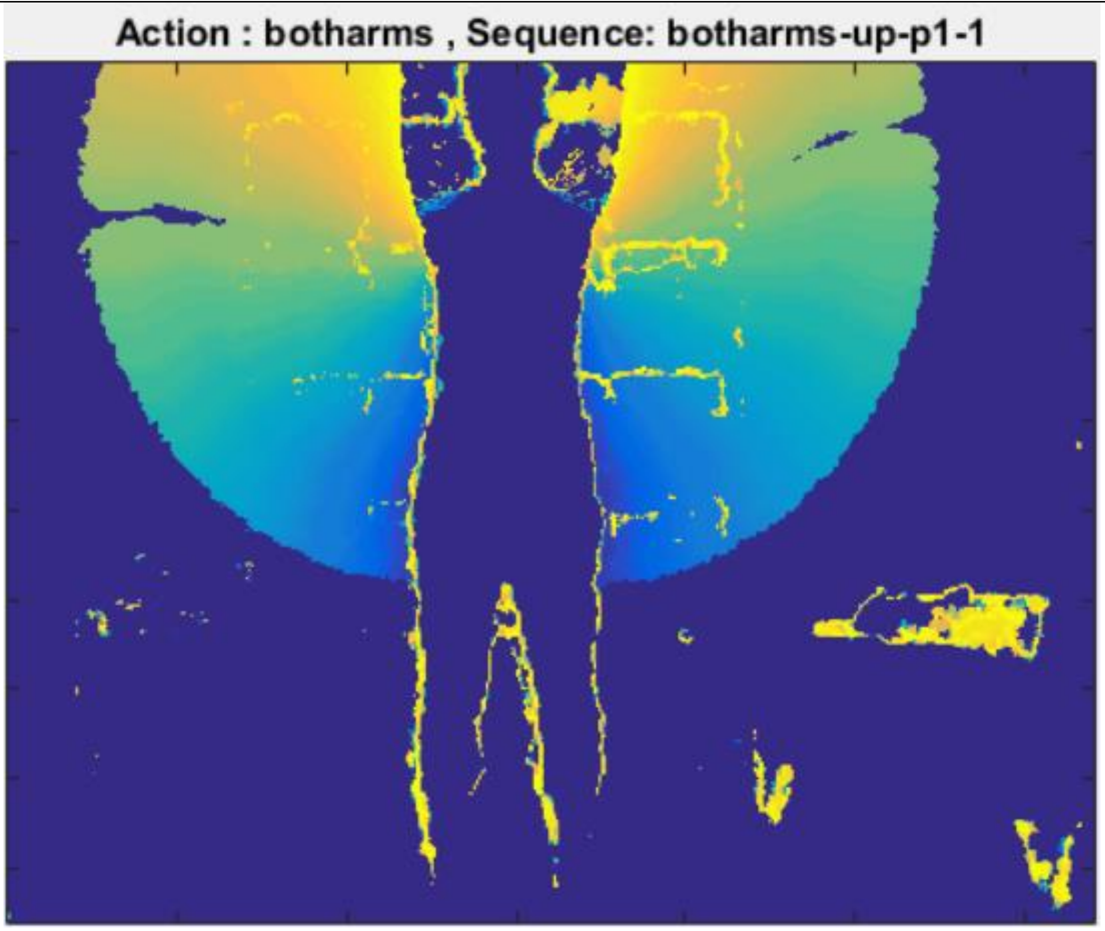
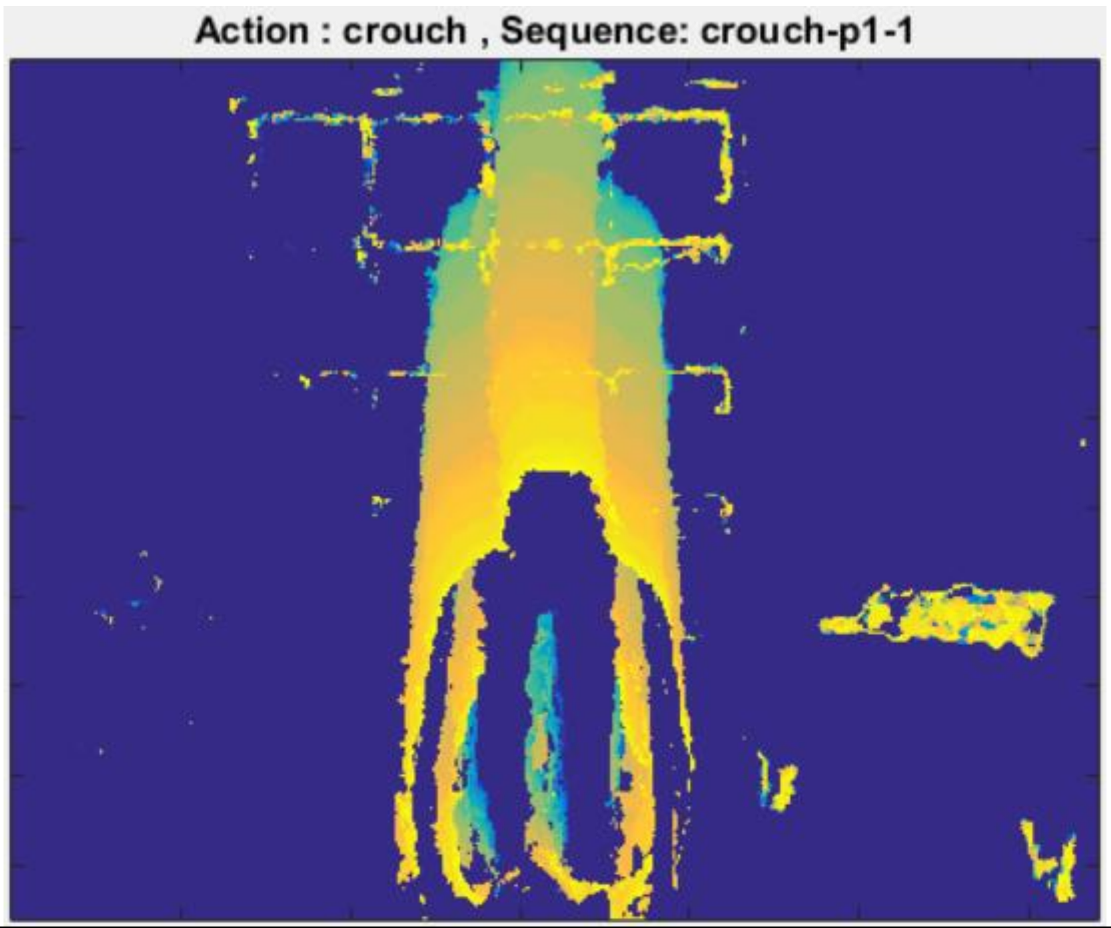
Submitted by: Disha Das

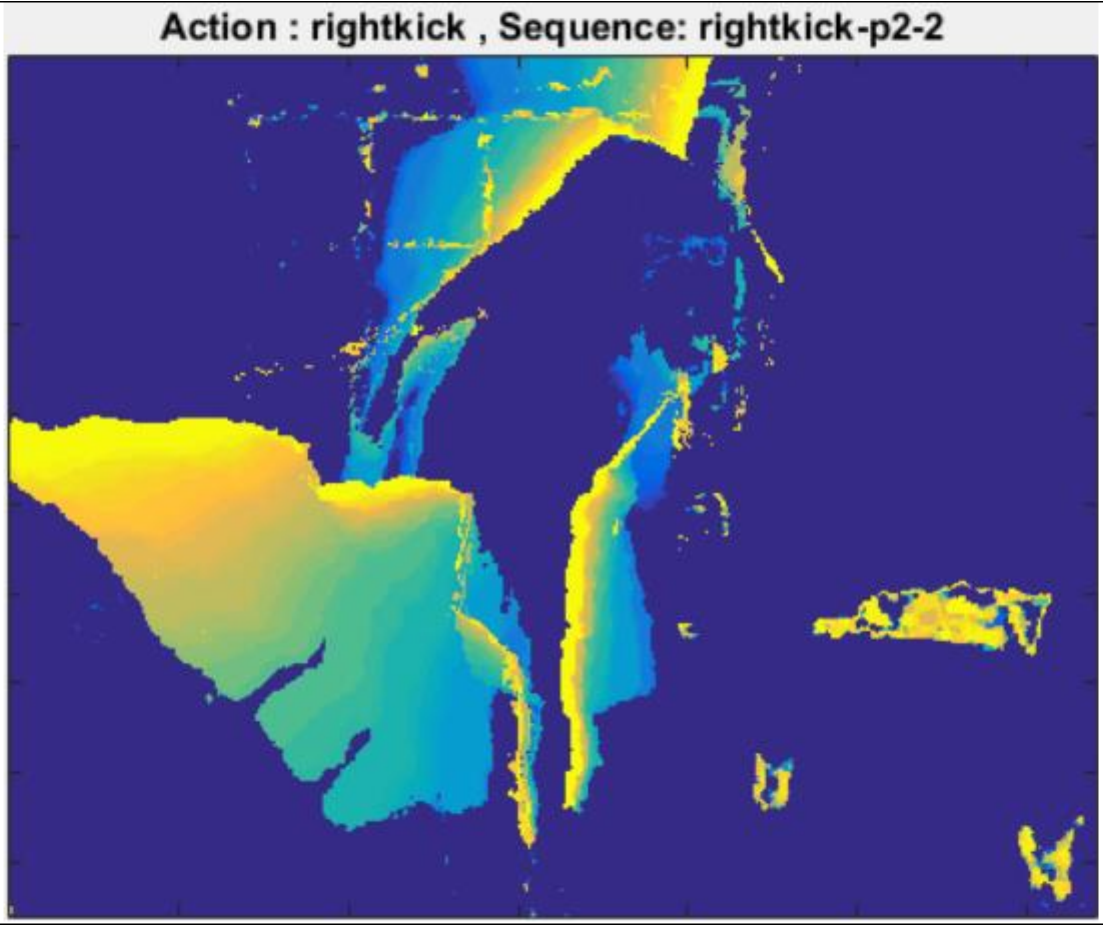
GTID: 903542819

Date: 11/21/2019

1 Programming

1.



In order to generate MHIs for each sequence, the foreground from images is extracted by simple background subtraction using a threshold. Tau value is set to the number of frames in a sequence. All the MHIs are computed by calling 'computeMHI.m' function from 'generateAllMHIs.m' script and stored in 'allMHIs.mat' file. Simple background subtraction is prone to noise as we can see above. We can notice window frames and several disjointed blobs all over the image. This can be reduced by the methods we'll discuss in the Q2 (Extra Question).

2.

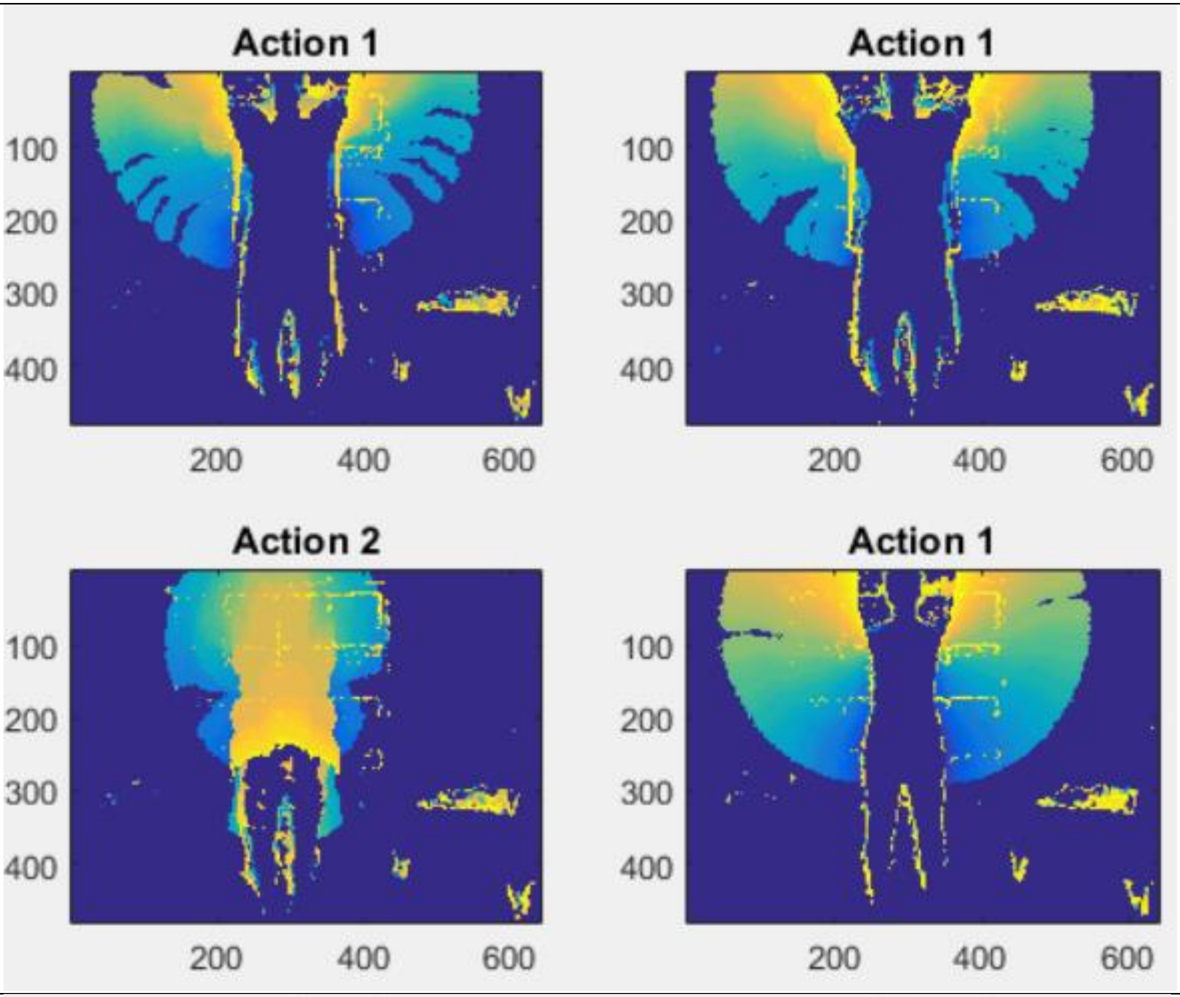
In this part, huMoments are calculated for the above MHIs using central moments. The file 'calculateHuMoments.m' calls the function 'huMoments.m' for every MHI. These 7d vectors are stored in 'huVectors.mat' file. In the extra question, we will make use of scale invariant moments and check if they make any improvements over results emerging from using central moments.

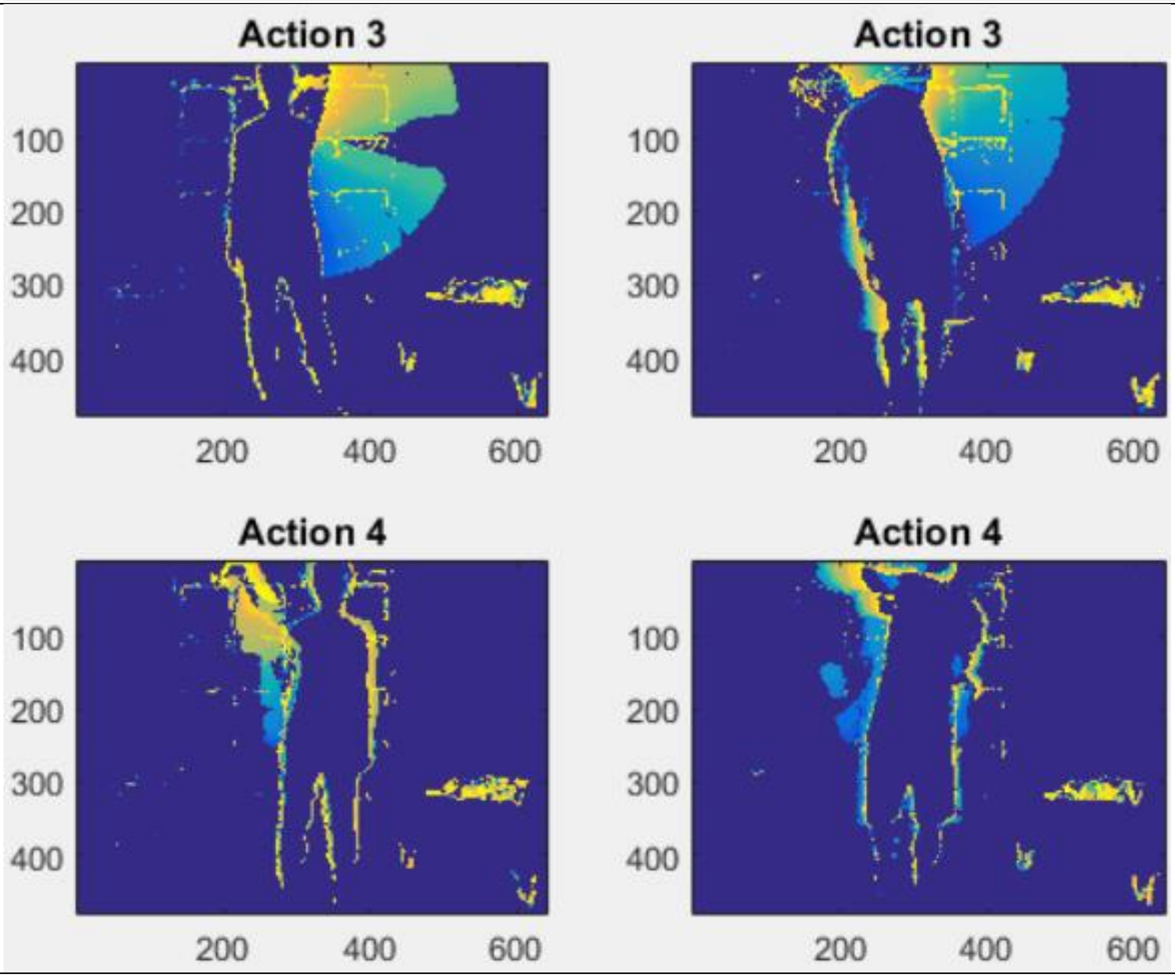
3.

The function 'predictAction.m' takes a Hu vector as test, takes Nx7 vector of Hu moments as training set and a vector of labels, trainMoments. It makes use of Normalized Euclidian distance to calculate nearest neighbours of test set. This is executed by calling the function 'normDist.m', which is included here as a helper function. The function gives an output of the label of the action that it predicts.

4.

In the examples given below, the first figure corresponds to the top 4 matches for **sequence number 2 (Action 1)** and the second one to **sequence number 12 (Action 3)**. In figure 1, we find a sequence from Action 2 ranked above a sequence from Action 1, even though it returns all the right matches. In figure 2, two sequences from Action 4 are returned. In the extra question, we will see how this can be remedied.


Action 1
Action 1
Action 2
Action 1
100
200
300
400
200
400
600


Action 3
Action 3
Action 4
Action 4
100
200
300
400
200
400
600

5.
In this script, we display a Confusion Matrix, the recognition rate per class and the overall recognition rate.

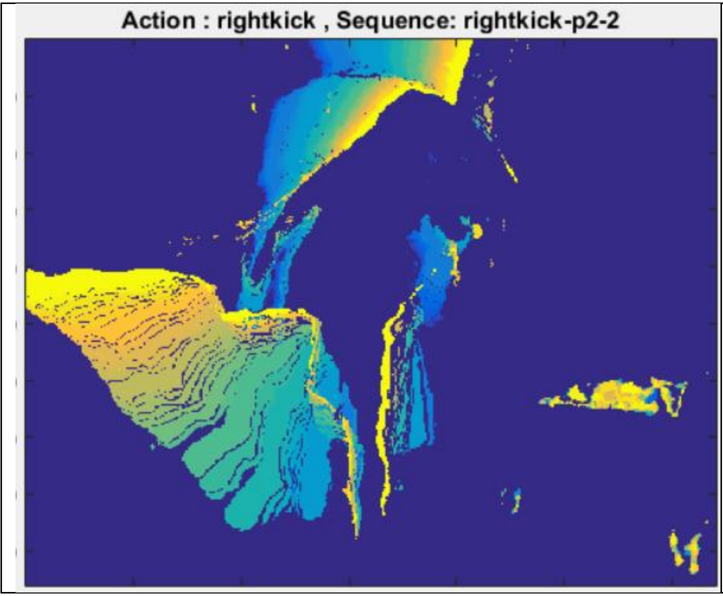
confusionMatrix =	**Overall Recognition Rate =**	**Class Recognition Rates =**
4 0 0 0 0	0.8000	1.0000
0 1 1 2 0		0.2500
0 0 4 0 0		1.0000
0 0 0 4 0		1.0000
0 1 0 0 3		0.7500

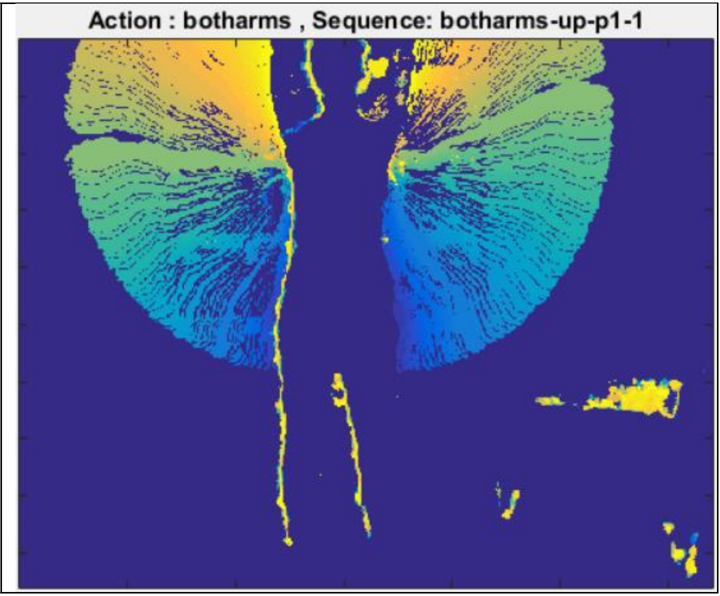
Overall Accuracy is 80%.

From the above data, we see that the most confused Action is Action 2, crouch. The second most confused action is Action 5. The crouch action seems to be similar to the punch action. In the extra question, we will see how certain improvements could affect changes in the Confusion matrix and improve the Recognition Rates.

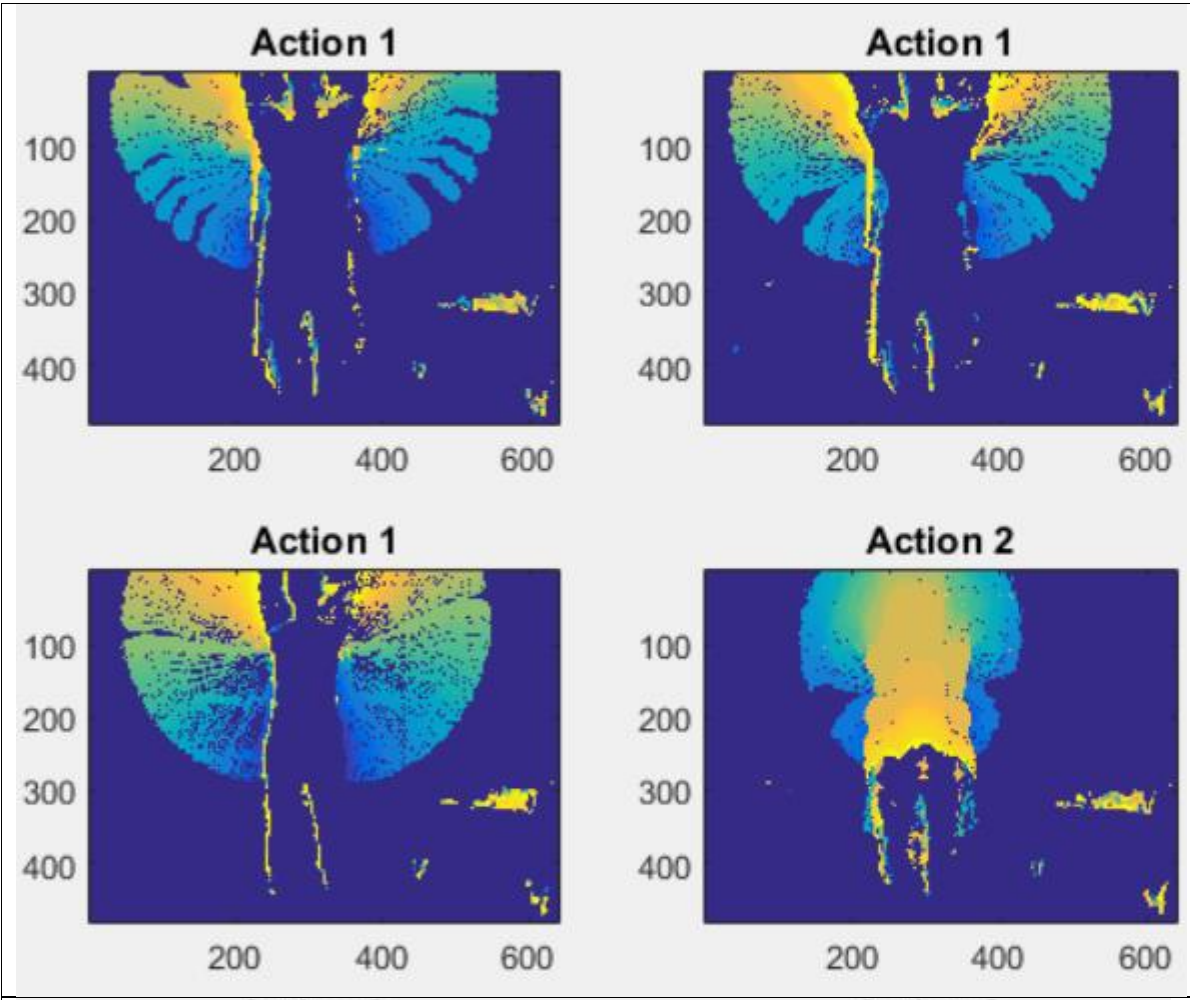
2 Optional: Extra Credit

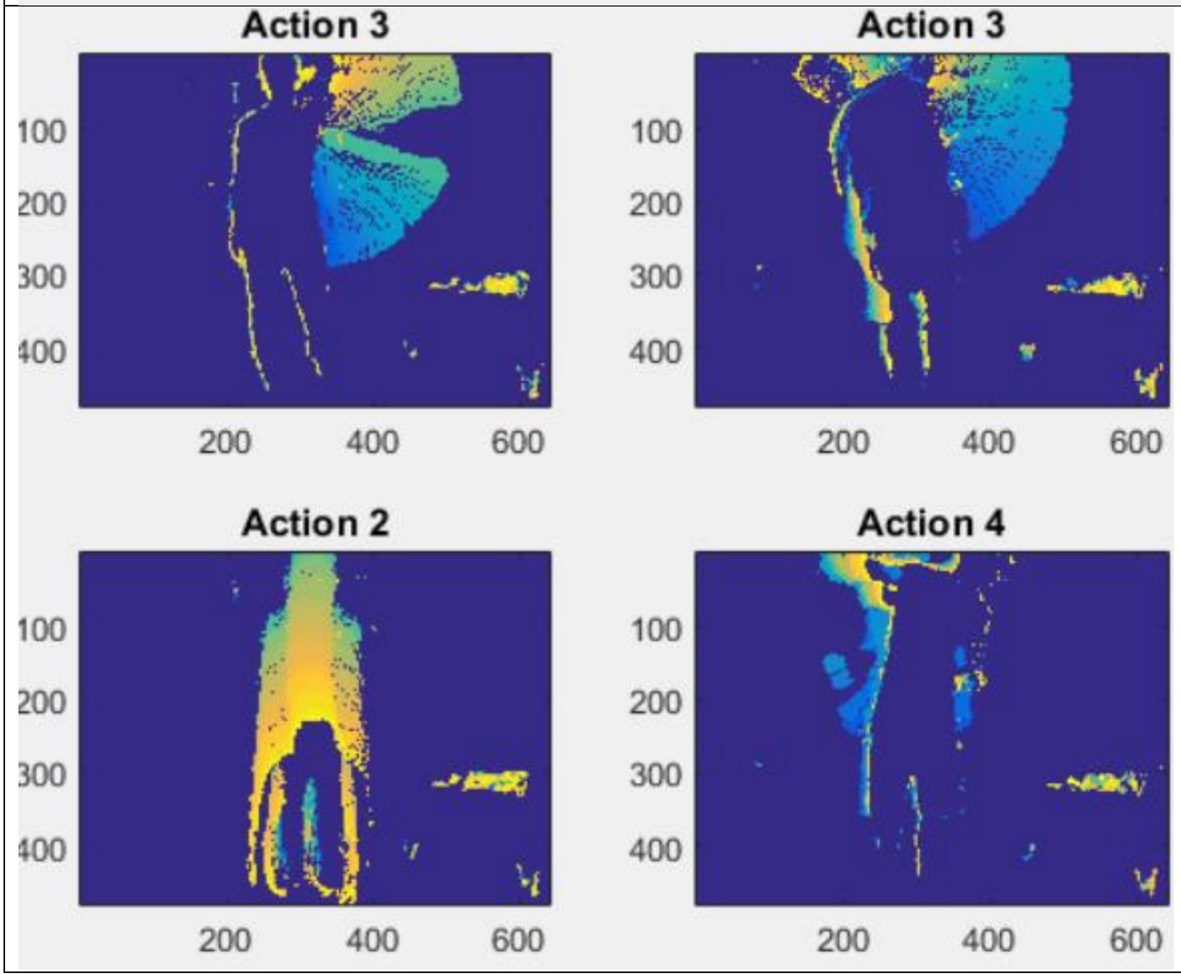
Implementing Gaussian filter for Background subtraction: In 'computeMHI.m', we make use of a Gaussian filter to blur the original image and subtract it from previous image. This results in relatively less noise and clearer boundaries of the foreground object. Some of the resulting MHIs are shown below:





Now we calculate the Hu moments using these MHIs and run 'showNearestMHIs.m'. The resulting images are compared to our earlier result. In figure 1, the rank of Action 2 has gone down by one, which is an improvement over the previous result. Not much of an improvement is observed in figure 2.



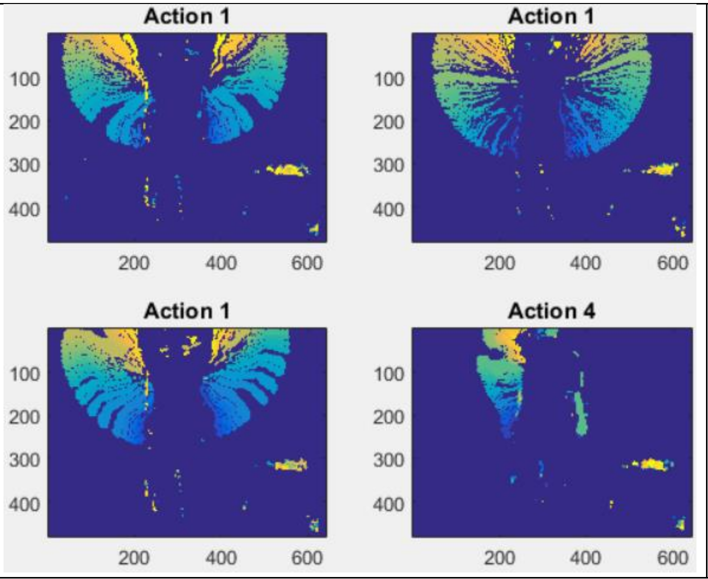


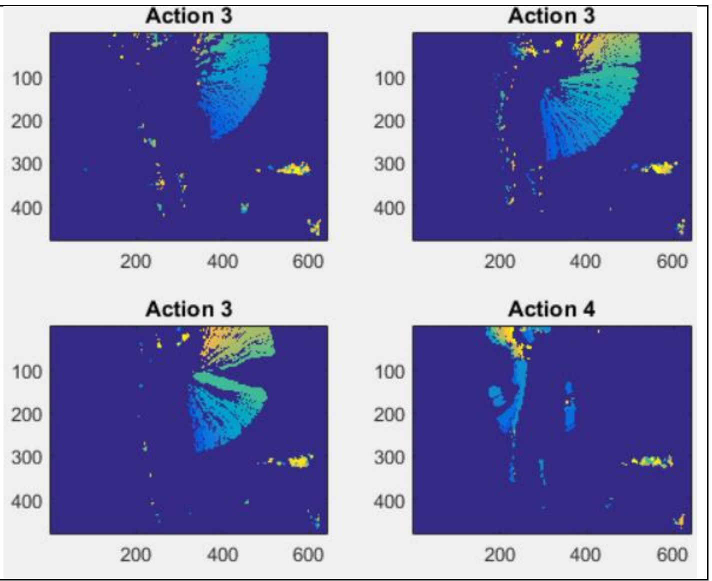
From the confusion matrix, it can be seen that the recognition rates of Actions 2 and 5 have improved. However, we see a reduction in improvement rate of Action 4. The overall Recognition rate has increased by 5%. Therefore using Gaussian blurring before background subtraction definitely improves upon our results.

confusionMatrix =	overallRecRate =	classRecRate =

4 0 0 0 0
0 2 1 1 0
0 0 4 0 0
0 1 0 3 0
0 0 0 0 4
	0.8500	1.0000
		0.5000
		1.0000
		0.7500
		1.0000

Using dilation and erosion to remove blobs: Using this in ‘computeMHI.m’, we generate the MHIs and compute Hu vectors. The results from ‘showNearestMHIs.m’ are given below. We see that there’s improvement in the results for Figure 2. We get all the matching actions for Action 3. The last MHI returned is also similar to Action 3.





Looking at the confusion matrix, we see that the results remain pretty much the same except for Action 4, for which it worsens. The resulting Overall recognition rate is reduced by 5%. Using dilation and erosion seems to give worse results because they eliminate larger blobs that may be relatively important.

confusionMatrix =	**overallRecRate =**	**classRecRate =**
4 0 0 0 0 **0 2 1 1 0** **0 0 4 0 0** **0 2 0 2 0** **0 0 0 0 4**	**0.8000**	**1.0000** **0.5000** **1.0000** **0.5000** **1.0000**

Using scale-invariant moments: We use scale invariant moments while calculating Hu moments. He resulting confusion matrix shows a huge degradation in the results.

confusionMatrix =	**overallRecRate =**	**classRecRate =**
2 1 1 0 0 **2 1 0 0 1** **1 0 2 0 1** **0 0 0 4 0**	**0.6500**	**0.5000** **0.2500** **0.5000** **1.0000**

0 0 0 0 4		1.0000

Using scale-invariant moments with Gaussian blurring: However, using Scale-invariant moments along with Gaussian blurring improves upon the previous result.

confusionMatrix =	**overallRecRate =**	**classRecRate =**
4 0 0 0 0	**0.8000**	**1.0000**
0 2 1 0 1		**0.5000**
0 1 3 0 0		**0.7500**
1 0 0 3 0		**0.7500**
0 0 0 0 4		**1.0000**

Using Normalized Euclidian distance with scale invariant moments and Gaussian blurring: The Gaussian blurring is tweaked a bit so that it yields a result different from the above case. Normalized Euclidian distance is the default method of computing distances between Hu Features. This yields a recognition rate of 75%.

confusionMatrix =	**overallRecRate =**	**classRecRate =**
4 0 0 0 0	**0.7500**	**1.0000**
0 1 2 0 1		**0.2500**
0 1 2 0 1		**0.5000**
0 0 0 4 0		**1.0000**
0 0 0 0 4		**1.0000**

Using Euclidian distance instead of Normalized Euclidian Distance over the above setting: Now the setting remains the same, except that we use Euclidian distance to calculate distances between Hu feature vectors. Surprisingly, this gives a better result, infact the best so far!

confusionMatrix =	**overallRecRate =**	**classRecRate =**
4 0 0 0 0	**0.9000**	**1.0000**
0 3 0 0 1		**0.7500**
0 0 4 0 0		**1.0000**
0 1 0 3 0		**0.7500**
0 0 0 0 4		**1.0000**

Conclusion: An improvement of 10% over recognition rate was achieved by incorporating Gaussian blurring for background subtraction, scale-invariant moments for Hu vector calculation and using Euclidian distance to compute nearest neighbours.