Daniela Florit

To run the code:

To run the code use the command python PA3_Daniela_Florit.py
Note inside the PA3_Daniela_Florit.py file that it's running a function called allExperiments().
This function runs all 14 experiments. I would not recommend doing this. Select only the ones you want to test. The ones resulting in the best metrics are experiments 10 and 14.
Please also note that this program assumes the folder with the video files is exactly as downloaded from the website. If it's not, please adjust the path in line 14 of PA3_Daniela_Florit.py

PA3 summary:

For this project, I tried three different feature extractors, all of which were based on optical flow. The general idea was to find good features to track (using ShiTomasi corner detection) and track them through the video frames [1], then find the difference between the coordinate of the original feature location, and the final location. From this, I was able to obtain the magnitude (distance) and direction (angle) of the optical flow for all selected features. The max number of features to track was experimentally found for optimality (using 100 for all experiments shown). The "allFlowFeatureVector" consisted on the magnitudes and angles corresponding to all found features. This should be the best feature vector since it had the most features. In the cases where the length of the feature vectors was less than the max number of features, the vectors were resized by filling them with zeros. The "meanFlowFeatureVector" feature extractor takes all the magnitudes and angles and returns a feature vector of size 2, containing only the mean distance and the mean angle. This was used to represent all the flow in the image. However, results were poor, mainly because it did not correctly represent all the flow in the image, and because it was only two features. The "minMaxFlowFeatureVector" consisted on a vector of size 6, including the mean distance, mean angle, max distance, max angle, min distance, and mean angle. This improved the accuracy of the prediction, however, still was lower than the accuracy obtained by using the "allFlowFeatureVector"

For the classifier, three different approaches were used. The first two were implementations of Support Vector Machine. First, I used the sklearn library's svm.SVC() class, which implements SVM one-vs-one approach [2]. For the parameters, I maintained the defaults, and modified only kernel, and degree. Tests were run using different values for kernel, including 'linear,' 'poly,' 'sigmoid,' and the default 'rbf.' The degree parameter is changed only when the kernel is set to 'poly.' Several values for degree were used, and the best one was found to be in the range of 2-4, giving results similar to those obtained by setting the kernel to 'linear.' These settings were found to be the most optimal. The second classifier was sklearn's svm.LinearSVC, which uses the one-versus-rest approach [3]. For this classifier, the default parameters were kept.
Finally, I used the sklearn library's neural networks class MLPClassifier() [4]. I trained and tested my data with this, maintaining the default parameters and changing only the alpha, and the number of hidden layers.

For the evaluation section, different tests were run in order to compare the performances using different parameters. A table with a description of all the tests run is included at the end.
For each of these, the Leave-one-out (LOO) cross-validation approach was used, which consists on training the classifier with all videos but one, and then testing with that video, then repeating this iterating through all videos. The program outputs the overall accuracy of each experiment once it finished. Also the precision, specificity and sensitivity for each class is shown. Note than classes have been represented as int values from 0 to 9. The most successful experiment was # 10, which uses an SVM with a linear kernel and reached a 25% accuracy. Also, experiment # 14, which uses a Neural Network with 150 hidden layers, and reached a 25% as well. The Precision, Specificity and Sensitivity for each class in those experiments is shown below.

Experiments:

Experiment	Classifier	Feature vector	Max # corners	For neural networks		For SVM		Accuracy
				alpha	# hidden layers	kernel	degree	
1	SVM	'all'	100	-	-	'rbf'	-	0.02
2		'mean'	100	-	-	'rbf'	-	0.15
3		'minMax'	100	-	-	'rbf'	-	0.03
4	LinearSVM	'all'	100	-	-	-	-	0.16
5		'mean'	100	-	-	-	-	0.12
6		'minMax'	100	-	-	-	-	0.18
7	NN	'all'	100	1e-5 (default)	15 (default)	-	-	0.16
8		'mean'	100			-	-	0.24
9		'minMax'	100			-	-	0.11
10	SVM	'all'	100	-	-	'linear'	-	0.25
11	SVM	'all'	100	-	-	'poly'	3 (default)	0.18
12	SVM	'all'	100	-	-	'poly'	6	-
13	NN	'all'	100	1e-5 (default)	50	-	-	0.23
14	NN	'all'	100		150	-	-	0.25

References:

[1] http://docs.opencv.org/trunk/d7/d8b/tutorial_py_lucas_kanade.html
[2] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[3] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC
[4] http://scikit-learn.org/stable/modules/neural_networks_supervised.html

Experiment 10
Experiment 10 uses SVM classifier with 'all' feature vector with 'linear' kernell and all other default parameters

Feature vector used: all
Classifier used: SVM
Max number of features (corners) extracted from video: 100
Relevant parameters:
kernel used: linear

Parameters for classifier:
{'kernel': 'linear', 'C': 1.0, 'verbose': False, 'probability': False, 'degree': 3, 'shrinking': True, 'max_iter': -1, 'decision_function_shape': None, 'random_state': None, 'tol': 0.001, 'cache_size': 200, 'coef0': 0.0, 'gamma': 'auto', 'class_weight': None}

Printing Metrics
The accuracy for this test is: 0.258751625851

Metrics for : Diving
precision: 0.000142836737609
sensitivity: 0.000142836737609
specificity: 0.948529411765

Metrics for : Golf-Swing
precision: 0.4
sensitivity: 0.444444444444
specificity: 0.904

Metrics for : Kicking
precision: 0.210526315789
sensitivity: 0.2
specificity: 0.878048780488

Metrics for : Lifting
precision: 0.222222222222
sensitivity: 0.333333333333
specificity: 0.948905109489

Metrics for : Riding-Horse
precision: 0.111111111111
sensitivity: 0.25
specificity: 0.81679389313

Metrics for : Running
precision: 9.99900009999e-05
sensitivity: 7.69171602184e-05
specificity: 0.923076923077

Metrics for : SkateBoarding
precision: 0.142857142857
sensitivity: 0.166666666667
specificity: 0.908396946565

Metrics for : Swing-Bench
precision: 0.285714285714
sensitivity: 0.1
specificity: 0.959349593496

Metrics for : Swing-Side
precision: 0.47619047619
sensitivity: 0.769230769231
specificity: 0.915384615385

Metrics for : Walking
precision: 0.666666666667
sensitivity: 0.272727272727
specificity: 0.97520661157

Experiment 14

Experiment 14 uses 'NN' classifier with 'all' feature vector with 150 hidden layers and all other default parameters

Feature vector used: all
Classifier used: NN
Max number of features (corners) extracted from video: 100
Relevant parameters:
    alpha used: 1e-05
    number of hidden layers 150

Printing Metrics
The accuracy for this test is: 0.258746442333

Metrics for : Diving
precision: 0.142857142857
sensitivity: 0.142857142857
specificity: 0.955882352941

Metrics for : Golf-Swing
precision: 0.6
sensitivity: 0.5
specificity: 0.952

Metrics for : Kicking
precision: 0.0625
sensitivity: 0.05
specificity: 0.878048780488

Metrics for : Lifting
precision: 0.000999000999001
sensitivity: 0.000166638893518
specificity: 0.992700729927

Metrics for : Riding-Horse
precision: 0.153846153846
sensitivity: 0.166666666667
specificity: 0.916030534351

Metrics for : Running
precision: 0.047619047619
sensitivity: 0.0769230769231
specificity: 0.846153846154

Metrics for: SkateBoarding
precision: 0.153846153846
sensitivity: 0.166666666667
specificity: 0.916030534351

Metrics for: Swing-Bench
precision: 0.333333333333
sensitivity: 0.35
specificity: 0.886178861789

Metrics for: Swing-Side
precision: 0.384615384615
sensitivity: 0.384615384615
specificity: 0.938461538462

Metrics for: Walking
precision: 0.391304347826
sensitivity: 0.409090909091
specificity: 0.884297520661