Advanced Vision Assignment 2

Seyed Behzad Tabibian and Gintautas Sasnauskas

1 Introduction

For the second assignment of Advanced Vision course we implemented a system, which classifies motion images into three categories. Motion images in the dataset represent a hand showing one of three signs in rock-paper-scissors game.

Solution for this problem was implemented in Python language using *OpenCV* library for processing images and *MLPy* library for classification.

2 Implementation

2.1 Hand recognition

When presented a motion sequence, moving hand is first detected in each image of the sequence. This is done in a number of steps.

At first image background is subtracted from the image. This allows us to only see what has changed in the image comparing to the background. Example of image from which background was subtracted is shown in Figure 1.

Obtained image is then smoothened using median blur in order to reduce noise. Such image is then transformed into black and white image according to threshold based on a colour difference between a typical hand in the data set and the background of the image. Such colour is found to lie between RGB values of $(0, 0, 27)$ and $(100, 100, 140)$. Image obtained by thresholding is shown in Figure 2.

To improve precision of the background subtraction and to account for possible lighting changes between different sets of images, first image of the sequence is also subtracted from the image. This removes all static features from the motion sequence. Such static features are often artifacts caused by changed lighting conditions between different the times different motion images were made. Both, the first image which is subtracted from the image of interest and the image of interest itself, are preprocessed in the same way.

Figure 1: Image obtained by performing background subtraction.

Figure 2: Image obtained by thresholding preprocessed image.

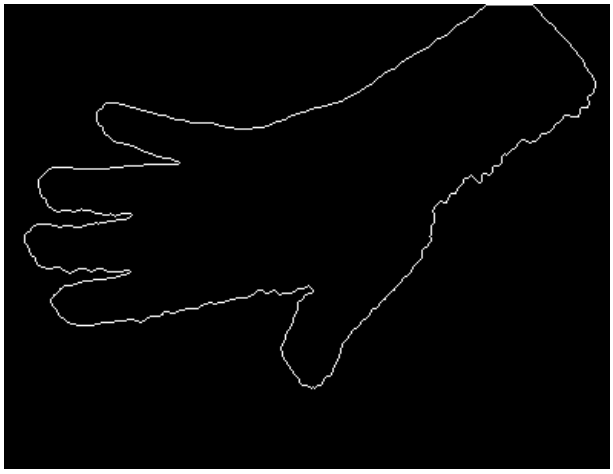


Figure 3: Image obtained by using simple chain approximation algorithm.

$Image_n^* = f(Image_n - Background) - f(Image_1 - Background), n \geq 2$

Dilation is then performed on such image. This helps further smoothen the detected hand and also helps eradicate any noise which appears not attached to the hand. Furthermore, dilation helps to connect parts of the same object which might appear disjoint because of shadows. This is often the case in images of hand showing "scissors" sign, where two fingers do not appear connected in the processed image because of a shadow between fingers.

Contours of the objects are then detected using simple chain approximation algorithm. Such algorithm scans binary image vertically as well as horizontally looking for chains of white pixels. It flips all the white pixels except from the pixels which represent both ends of each such chain. Using this technique we are able to obtain contours of all objects in the image. Contours of object obtained by using this method is showin in Figure 3.

Largest object in the image is then chosen. If area of such object is above given threshold, the object is chosen as representation of the hand in the image, this helps not include initial and last images of the hand in the sequence where it is not fully visible in the picture.

Summary of the algorithms and parameters used in the Hand Recognition module is presented in Table 1.

2.2 Feature extraction

Two sets of features are extracted from the motion picture image to be used in the classifier:

Preprocessing:	Smoothing with Median blur Background subtraction
Background subtraction:	Using background image provided
Thresholding:	on RGB image, $(0, 0, 27)$ and $(100, 100, 140)$
First image subtraction:	subtracting first image of the sequence
Mathematical Morphology	Dilation using Cross Kernel of size 11×11
Contour Extraction:	Simple Chain Approximation
Area measure:	discarding contours of small size

Table 1: Summary of Classification Module parameters and algorithms

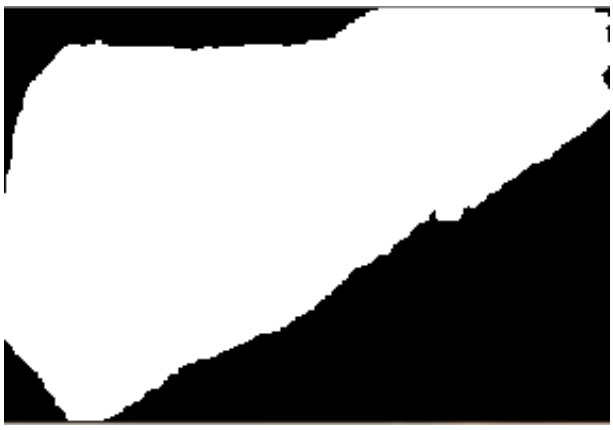


Figure 4: Motion history image for "rock" sign.

1. Hu Moments;
2. Changes in the area of the extracted object over time.

Hu Moments

Hu set of invariant moments is used as the first set of features. In order to calculate Hu moments, motion history image for a given video clip is first built[1]. Motion history image is built from black and white images reconstructed from contour images. This is done by combining them into one image by superimposing them. Example motion history images for "rock", "paper" and "scissors" signs are shown in Figure 4, Figure 5 and 6 respectively.

Hu moments are then extracted from motion history image and used as features of the motion image.

[1]this is done by using modified version of sample code from OpenCV. Original code is available at `https://code.ros.org/trac/opencv/browser/trunk/opencv/samples/python/motempl.py`

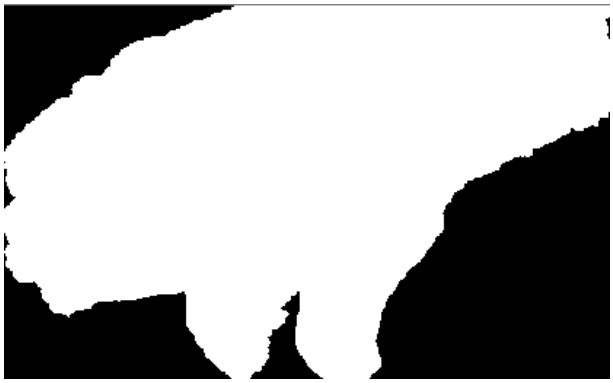


Figure 5: Motion history image for "paper" sign.

Figure 6: Motion history image for "scissors" sign.

Temporal function of the object size

In addition to Hu moments, additional set of five features is generated by temporarily dividing provided video clip into five equal segments and calculating average area taken by the hand in each of these segments. This way we can see how area of the hand in the image changes over time and represent such changes as features usable by the classifier.

While being a simple method, it explains large amount of variance between different signs.

2.3 Classifier

To classify different types of image sequences MLpy library was used. MLpy is a machine learning library for python which provides a variety of algorithms and tools to perform machine learning on data.

The algorithm used in this project is Linear Support Vector Machine which was trained over 5 principal components of the input data. Methods of producing such data was further described in Section **??**.

Principal Component Analysis

Input vector consists of 12 features for every image sequence - 7 Hu moments and 5 area measures over time, therefore dataset lies in 12 dimensional space. principal component analysis (PCA) was performed on the input dataset to reduce dimensionality of the dataset. Prior to PCA all the features are normalised the zero mean and unit standard deviation. Figure 7 shows that first 5 principal components can explain most of the variance of the dataset. Therefore, we can deduct that input data sits on a five dimensional manifold.

Using these components we project entire dataset as well as any new input data into 5 dimensional subspace.

Support Vector Machine

Projected data is then used to train a linear support vector machine (SVM). The algorithm used is an implementation of multi-class support vector classification by Crammer and Singer. This algorithm is also available in MLpy library.

Logistic regression was also tested. However, as results in Section **??** shows SVM outperformed logistic regression and therefore was chosen to be used in the project.

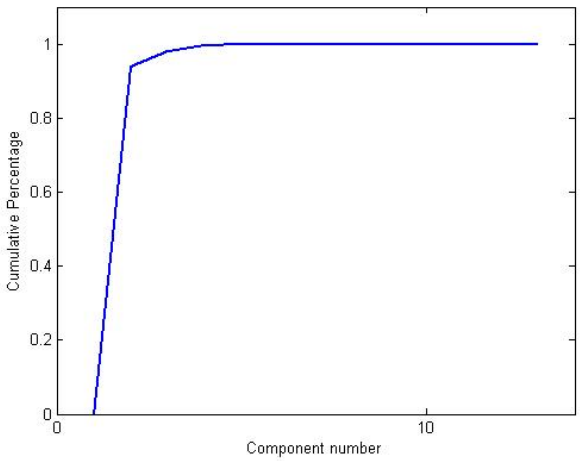


Figure 7: Cumulative Percentage PCs

Preprocessing:	1. Feature Normalization, zero mean standard deviation 2. Principal Component Analysis with 5 Components
Training Algorithm	Multi-class SVM
Performance Measure	Leave-One-Out Cross-Validation

Table 2: Summary of classification module parameters and algorithms

Cross Validation

In order to evaluate performance of the classifier we utilised Leave-One-Out Cross-Validation. At each step the algorithm takes one instance of the data and performs training on the remaining data. Accuracy of prediciting left out data point is used as an evaluation metric. Using the trained classifier test instances are checked and results averaged for entire dataset.

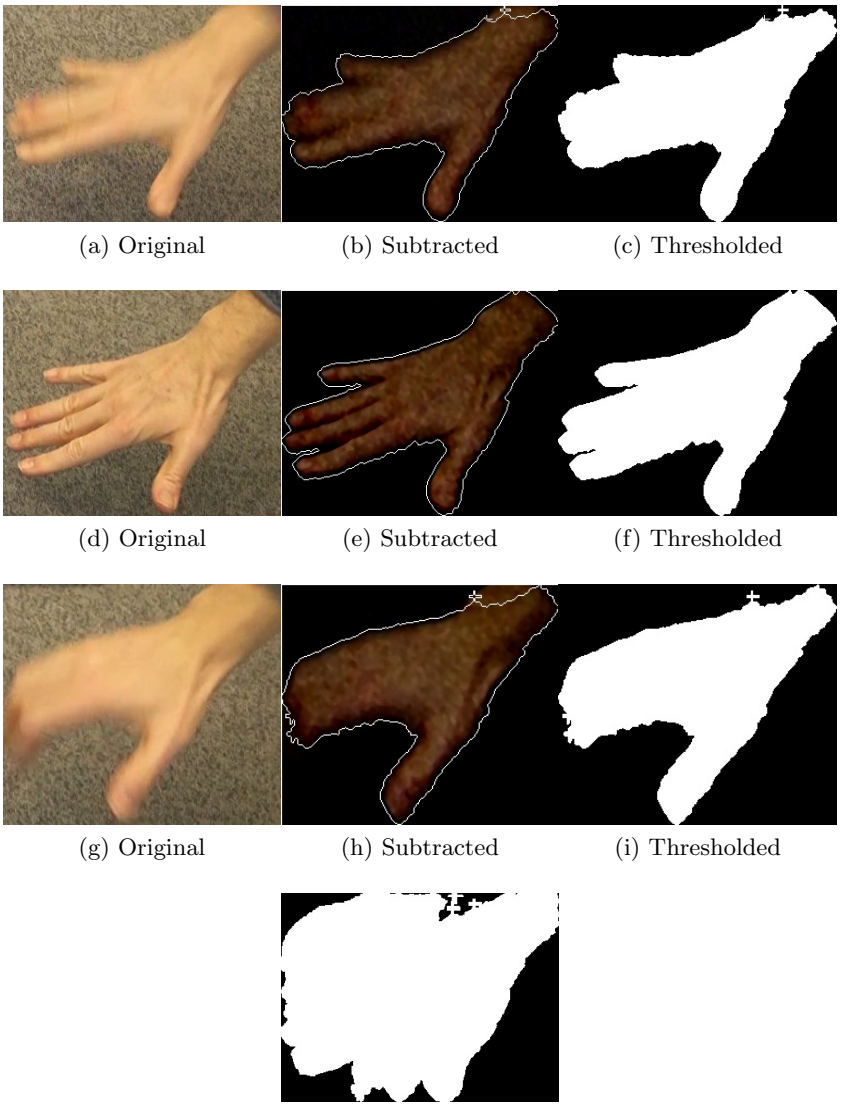
Summary of the algorithms and parameters used in the classification module is presented in table 2.

3 Results

Figures 8, Figure 9 and Figure 10 show different images sampled from each *rock*, *paper*, *scissors* signs respectively. Each set includes an original image, background-subtracted image and thresholded image after morphology is performed on the data. The line on the subtracted image demonstrates the detected boundary of the hand.

From the images it can be seen that boundary does not correspond exactly with the

true boundary of the hand. This is the effect of dilation using cross kernel. In case of *scissors* sign dilation helps connect two fingers which often appear disjoint due to a shadow between them.



(a) Original (b) Subtracted (c) Thresholded

(d) Original (e) Subtracted (f) Thresholded

(g) Original (h) Subtracted (i) Thresholded

(j) Motion History Image

Figure 8: Images for paper action

Cross validation test results showed that best accuracy over validation data set was achieved by using combination of Hu moments and temporal area of the extracted object as features of the motion image and multi-class SVM for classification. Results for using different feature sets are shown in Table 4, for using different number of principal components are shown in Table 5 and for using different classifiers in Table 6. Final 5

(a) Original (b) Subtracted (c) Thresholded

(d) Original (e) Subtracted (f) Thresholded

(g) Original (h) Subtracted (i) Thresholded

(j) Motion History Image

Figure 9: Images for rock action

(a) Original (b) Subtracted (c) Thresholded

(d) Original (e) Subtracted (f) Thresholded

(g) Original (h) Subtracted (i) Thresholded

(j) Motion History Image

Figure 10: Images for scissors action

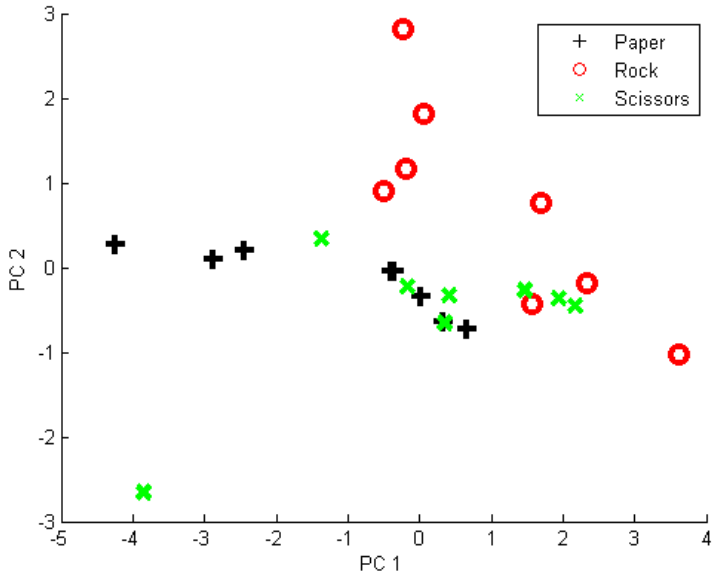


Figure 11: Scatter plot shows dependency between first two principal components used and actual class of an object.

top Principle Components that are used to train the SVM are demonstrated in table 3.

	PC 1	PC 2	PC 3	PC 4	PC 5	Class Label
1	2.456	-2.448	0.201	0.001	0.4	paper
2	1.939	-0.359	-0.044	-0.048	0.837	paper
3	2.613	0.328	-0.634	-0.641	-1.164	paper
4	1.873	0.011	-0.35	-0.593	0.839	paper
5	2.513	0.665	-0.724	-1.003	-0.679	paper
6	1.887	-0.395	-0.044	0.648	0.409	paper
7	2.479	-4.245	0.268	-0.381	-0.211	paper
8	3.581	-2.873	0.101	0.348	0.499	paper
9	-0.85	1.565	-0.433	0.655	-0.383	rock
10	-4.475	-0.496	0.911	-0.458	-0.031	rock
11	-1.49	-0.228	2.82	-1.773	-0.293	rock
12	-2.458	0.063	1.808	0.93	-0.107	rock
13	-2.533	-0.181	1.169	1.133	-0.372	rock
14	-0.741	2.338	-0.184	-0.644	0.719	rock
15	-1.903	1.683	0.757	-0.435	0.558	rock
16	0.235	3.613	-1.023	-0.887	-0.137	rock
17	0.125	2.17	-0.455	0.559	0.001	scissors
18	1.088	-0.178	-0.219	0.903	-0.474	scissors
19	-7.689	-3.842	-2.651	-0.522	0.093	scissors
20	0.205	-1.371	0.342	0.888	-0.455	scissors
21	-1.051	1.938	-0.365	1.315	0.211	scissors
22	1.565	0.412	-0.332	-0.051	-0.071	scissors
23	0.091	1.472	-0.265	0.045	0.415	scissors
24	0.541	0.358	-0.652	0.009	-0.602	scissors

Table 3: Set of 5 Principle components used for each class

Features	Training set	Validation set
Temporal area of object only	73.9%	54.2%
Hu moments only	79.3%	58.3%
Both	91.8%	87.5%

Table 4: Cross validation results when using different sets of features with optimal number of principal components and optimal classifier.

Number of PC	Training set	Validation set
3	87.3%	66.7%
4	91.8%	79.2%
5	91.8%	87.5%
6	95.7%	73.3%
7	96.0%	83.3%

Table 5: Cross validation results when using different number of principal components with set of features and optimal classifier.

Classifier	Training set	Validation set
Logistic regression	88.6%	66.7%
Multi-class SVM	91.8%	87.5%

Table 6: Cross validation results when using different classifiers with optimal set of features and optimal number of principal components.

4 Source code

'Python Script for general hand recognition and feature extraction'

```
import cv
import os
import fnmatch
import time
import av_utils
import mhi_update
import classify

def getBiggestCountour(Seq):
    max_area =0
    best_area=Seq
    seq=Seq
    while seq:
        area=cv.ContourArea(seq)
        if (area>max_area):
            max_area=area
            best_area=seq
        seq=seq.h_next()
    return best_area,max_area

def getHuMoments(Seq,binary=1):
    moments=cv.Moments(Seq,binary)
    return cv.GetHuMoments(moments)
```

```
25
    Thresh = 3
27
    bg = './data/background2.jpg'
29
    if os.path.exists('./output.txt'):
31      os.remove('./output.txt')

33  pathes = [('./data/train/1-1/',1),('./data/train/1-2/',1),('./
        data/train/1-3/',1)
              ,('./data/train/1-4/',2),('./data/train/1-5/',2),('./
                 data/train/1-6/',2)
35            ,('./data/train/1-7/',3),('./data/train/1-8/',3),('./
                 data/train/1-9/',3)
              ,('./data/train/2-1/',3),('./data/train/2-2/',1),('./
                 data/train/2-3/',2)
37            ,('./data/train/3-9/',1),('./data/train/2-5/',1),('./
                 data/train/2-6/',2)
              ,('./data/train/3-1/',2),('./data/train/3-2/',2),('./
                 data/train/3-3/',2)
39            ,('./data/train/3-4/',3),('./data/train/3-5/',3),('./
                 data/train/3-6/',3)
              ,('./data/train/3-7/',1),('./data/train/3-8/',1),('./
                 data/train/2-4/',3)]
41  index=0
    pred=-1
43  for path, type in pathes:
        index=index+1
45      mhi=mhi_update.mhi()
        time = 10
47      dirList=os.listdir(path)
        cv.NamedWindow('image')
49
        av_utils.av_debug('background image loaded from: '+ bg)
51      bg_img = cv.LoadImage(bg)
        img_size = cv.GetSize(bg_img)
53      img_depth = bg_img.depth
        img_channel = bg_img.channels
55
        av_utils.av_debug('image info, size:' + str(img_size) + ',
            ' +
57                          'depth:' + str(img_depth) + ', ' +
```

```
                    'channel:' + str(img_channel))
59
    bg_gray_img = cv.CreateImage(img_size, img_depth, 1)
61
    cv.ShowImage('image', bg_img)
63  cv.WaitKey(time)

65  storage = cv.CreateMemStorage(0)
    bounding_rects = list()
67
    last_bw_sub = False
69  last_bw_sub_set = False

71  maxareaseq = []
    rectimgs = []
73  indexj = 0

75  for fname in dirList:
        indexj = indexj + 1
77      if not fnmatch.fnmatch(fname, '*.jpg'):
            continue
79
        fname = path + fname
81      img = cv.LoadImage(fname)
        img_sub = cv.CreateImage(img_size, img_depth,
            img_channel)
83      cv.Sub(img, bg_img, img_sub)
        cv.Smooth(img_sub, img_sub, cv.CV_MEDIAN, 5)
85
        img_bw_sub = cv.CreateImage(img_size, cv.IPL_DEPTH_8U,
            1)
87
        cv.ShowImage('image', img_sub)
89      cv.InRangeS(img_sub, (0, 0, 27), (100, 100, 140),
            img_bw_sub)

91      if not last_bw_sub_set:
            last_bw_sub_set = True
93          last_bw_sub = img_bw_sub
            continue
95      else:
            cv.Sub(img_bw_sub, last_bw_sub, img_bw_sub)
97
```

```
        img_bw_sub_dilated=cv.CreateImage(img_size, cv.
            IPL_DEPTH_8U, 1)
 99     kernel=cv.CreateStructuringElementEx(11,11,6,6,cv.
            CV_SHAPE_CROSS)
        cv.Dilate(img_bw_sub,img_bw_sub_dilated,kernel)
101
        img_bw_sub_dilated_preContours = cv.CreateImage(
            img_size,
103                                                         cv.
                                                            IPL_DEPTH_8U
                                                            , 1)
        cv.Copy(img_bw_sub_dilated,
            img_bw_sub_dilated_preContours)
105
        seq = cv.FindContours(img_bw_sub_dilated_preContours,
            storage,
107                           cv.CV_CHAIN_APPROX_SIMPLE)

109     if (len(seq) != 0):
            cnt_biggest, area = getBiggestCountour(seq)
111         if (area > 15000):
                maxareaseq.append(area)
113
                rect = cv.BoundingRect(cnt_biggest)
115             bounding_rects.append(rect)
                cv.DrawContours(img_sub, cnt_biggest,
117                             cv.RGB(255,255,255),
                                cv.RGB(255,255,255), 0)
119

121             cv.Rectangle(img_sub, (rect[0], rect[1]),
                                      (rect[0] + rect[2], rect
                                          [1] + rect[3]),
123                                   cv.RGB(255,255,255))

125             tempimg = cv.CreateImage(img_size, img_depth,
                    img_channel)
                cv.DrawContours(tempimg, cnt_biggest,
127                             cv.RGB(255,255,255),
                                cv.RGB(255,255,255), 0, -1)
129             rectimgs.append(cv.GetSubRect(tempimg, rect))

131             cv.ShowImage('image', img_sub)
```

```
                cv.ShowImage('image_temp', img_bw_sub)
133
135             largest_bound = (bounding_rects[0][0],
                                 bounding_rects[0][1],
137                              bounding_rects[0][0] +
                                    bounding_rects[0][2],
                                 bounding_rects[0][1] +
                                    bounding_rects[0][3])
139
                for i in bounding_rects:
141                 if i[0] < largest_bound[0]:
                        largest_bound = (i[0], largest_bound
                            [1],
143                                      largest_bound[2],
                                            largest_bound[3])
                    if i[1] < largest_bound[1]:
145                     largest_bound = (largest_bound[0], i
                            [1],
                                         largest_bound[2],
                                            largest_bound[3])
147                 if (i[2] + i[0]) > largest_bound[2]:
                        largest_bound = (largest_bound[0],
                            largest_bound[1],
149                                      (i[0] + i[2]),
                                            largest_bound[3])
                    if (i[3]+i[1]) > largest_bound[3]:
151                     largest_bound = (largest_bound[0],
                            largest_bound[1],
                                         largest_bound[2], (i
                                            [1] + i[3]))
153
                bounding_box = (largest_bound[0], largest_bound
                    [1],
155                             largest_bound[2] -
                                    largest_bound[0],
                                largest_bound[3] -
                                    largest_bound[1])
157             boxsize = (bounding_box[2], bounding_box[3])

159         if (cv.WaitKey(time) == 115):
                cv.SaveImage('./results/'+str(index)+str(indexj
                    )+str('bw.jpg'),
```

```
161                                 cv.GetSubRect(tempimg, rect))
                    cv.SaveImage('./results/'+str(index)+str(indexj
                        )+str('orig.jpg'),
163                                 cv.GetSubRect(img, rect))
                    cv.SaveImage('./results/'+str(index)+str(indexj
                        )+str('sub.jpg'),
165                                 cv.GetSubRect(img_sub, rect))

167

169         motion = cv.CreateImage(boxsize, img_depth, 1)
            frame = 0
171         timestamp = 0
            frames = len(rectimgs)
173         centre = frames / 2
            for rectimg in rectimgs:
175             tempimg = cv.CreateImage(boxsize, img_depth, 3)
                cv.Resize(rectimg, tempimg)
177             mhi.update_mhi(tempimg, motion, 25)
            hu_moments=getHuMoments(cv.GetMat(mhi.mhi),0)
179         hu_array=list(hu_moments)

181
            for i in range(5):
183             area = 0
                cnt = 0
185             for ii in range(len(maxareaseq) / 5):
                    j = len(maxareaseq) / 5 * i + ii
187                 if j < len(maxareaseq):
                        cnt = cnt + 1
189                     area = area + maxareaseq[j]

191             area = area / cnt
                hu_array.append(area)
193         if(type==-1):
                if(pred==-1):
195                 model,pca_model,mean,stan= classify.script()
                    classify.pred_new_instance(model,pca_model,mean,
                        stan,hu_array)
197                 pred=1
                else:
199                 classify.pred_new_instance(model,pca_model,mean,
                        stan,hu_array)
```

```
201         else:
                hu_array.append(type)
203             av_utils.write_array_to_file(hu_array)
                cv.ShowImage('image_mhi',mhi.mhi)
205             if(cv.WaitKey(time) ==115):
                    out=cv.CreateImage(boxsize, cv. IPL_DEPTH_32F, 3)
207                 cv.CvtColor(mhi.mhi,out,cv.CV_GRAY2BGR)
                    cv.CvtScale(out,out,255)
209                 cv.SaveImage('./results/'+str(index)+str('mhi.jpg')
                        ,out)
    if(pred==-1):
211         classify.script()
```

'Python Script for Classification, training and evaluation'

```
import numpy
import io
import os
import mlpy
import random

def preproccessing(data,y):
    y=numpy.array(y,dtype=numpy.int)-1
    mu=numpy.mean(data,0)
    stan=numpy.std(data,0)
    data = data-numpy.tile(mu,(numpy.size(data,0),1))
    data=data/numpy.tile(stan,(numpy.size(data,0),1))

    randperm=random.sample(range(0,numpy.size(data,0)),numpy.
        size(data,0))
    data=data[randperm,:]
    y=y[randperm,:]

    pca=mlpy.PCA()
    pca.learn(data)
    data_pca=pca.transform(data,k=5)
    return (data_pca,y,pca,mu,stan)
def train(x,y):
    logReg=mlpy.LibLinear(solver_type='mcsvm_cs')
    logReg.learn(x,y)
    return logReg
def evaluate(y_model,y_true,confusion=None):
    if(confusion!=None):
        confusion[y_model,y_true]=confusion[y_model,y_true]+1
    return numpy.size(numpy.nonzero(y_model==y_true))/float(
        numpy.size(y_true)),confusion
def pred_new_instance(model,pca_model,mean,stan,x):
    x = x-mean
    x= x/stan
    pcad=pca_model.transform(x,5)
    y=model.pred(pcad)
    print('label is '+str(y))


def cross_validation(data,y):
    idx=mlpy.cv_kfold(numpy.size(data,0), numpy.size(data,0),
        strat=None, seed=4)
```

```
        corrects=0
41      falses=0
        confusion=numpy.zeros((3,3))
43      testavg = 0
        testavgcount = 0
45      test_res=0
        for tr,ts in idx:
47          X_tr=data[tr,:]
            Y_tr=y[tr]
49          X_ts=data[ts,:]
            Y_ts=y[ts]
51
            model=train(X_tr,Y_tr)
53          y_training_pred=model.pred(X_tr)
            y_trained=y_training_pred.copy()
55          y_dat=model.pred(X_ts)

57          acc,conf=evaluate(y_trained,Y_tr)
            testavg = testavg + acc
59          acc,confusion=evaluate(y_dat,Y_ts,confusion)
            test_res=test_res+acc
61          testavgcount = testavgcount + 1

63      return confusion, testavg/float(testavgcount), test_res/
            float(testavgcount)
    def learn(data, y):
65
        (X,y,pca_model,mean,stan)=preproccessing(data,y)
67      model = train(X,y)

69      confusion, training_error,test_error=cross_validation(X,y)
        print('average training error: '+str(training_error))
71      print('test error: '+str(test_error))
        print('confusion matrix\n'+str(confusion))
73      return model,pca_model,mean,stan
    def write_features(x,y):
75       write=file('features.txt','a');
         p=(y, x)
77       for ii in range(0,numpy.size(x,0)):
             i=x[ii,:]
79           j=y[ii]
             for k in i:
81               write.write(str(k)+',')
```

```
            write.write(str(j))
83          write.write('\r\n')
        write.close()
85
def script():
87      [X,y]=read_file()
        #x_dat,y=preproccessing(X,y)
89      #write_features(x_dat,y)
        return learn(X,y)
91
def read_file():
93      read=file('output.txt','r')
        full_data=None
95      while(1):
            data=read.readline()
97          if (data==''):
                break
99          arr=data.split(',')
            set=numpy.array([])
101         for i in arr:
                if(i=='\r\n'):
103                 continue
                set=numpy.append(set,float(i))
105
            if(full_data!=None):
107             full_data=numpy.vstack((full_data,set))
            else:
109             full_data=set
        return (full_data[:,0:-1],full_data[:,-1])
111 #script()
```

'Python Script for general utility functions'

```
write=None
def av_debug(msg):
    print msg
def write_array_to_file(data):
    global write
    if(write==None):
        write=file('output.txt','a');
    for i in data:
        write.write(str(i)+',')
    write.write('\r\n')
    write.flush()
```

'Python Script for Motion History Image construction based on OpenCV sample'

```
import urllib2
import sys
import time
from math import cos, sin
import cv
class mhi():
    def __init__(self):
        self.CLOCKS_PER_SEC = 1
        self.MHI_DURATION = 5

        self.MAX_TIME_DELTA = 0.5
        self.MIN_TIME_DELTA = 0.05
        self.N = 100

        self.buf = range(100)
        self.last = 0
        self.mhi = None # MHI
        self.orient = None # orientation
        self.mask = None # valid orientation mask
        self.segmask = None # motion segmentation map
        self.storage = None # temporary storage
        self.counter=0
    def update_mhi(self,img, dst, diff_threshold):
        last=self.last
        mhi=self.mhi
        storage=self.storage
        mask=self.mask
        orient=self.orient
        segmask=self.segmask
        counter=self.counter
        buf=self.buf
        N=self.N
        MIN_TIME_DELTA=self.MIN_TIME_DELTA
        MAX_TIME_DELTA = self.MAX_TIME_DELTA
        CLOCKS_PER_SEC=self.CLOCKS_PER_SEC
        MHI_DURATION =self.MHI_DURATION

        timestamp = counter#time.clock() / CLOCKS_PER_SEC # get
            current time in seconds
        counter=counter+0.5
        size = cv.GetSize(img) # get current frame size
```

```
        idx1 = last
42      if not mhi or cv.GetSize(mhi) != size:
            for i in range(N):
44              buf[i] = cv.CreateImage(size, cv.IPL_DEPTH_8U,
                    1)
                cv.Zero(buf[i])
46          mhi = cv.CreateImage(size,cv. IPL_DEPTH_32F, 1)
            cv.Zero(mhi) # clear MHI at the beginning
48          orient = cv.CreateImage(size,cv. IPL_DEPTH_32F, 1)
            segmask = cv.CreateImage(size,cv. IPL_DEPTH_32F, 1)
50          mask = cv.CreateImage(size,cv. IPL_DEPTH_8U, 1)

52      cv.CvtColor(img, buf[last], cv.CV_BGR2GRAY) # convert
            frame to grayscale
        idx2 = (last + 1) % N # index of (last - (N-1))th frame
54      last = idx2
        silh = buf[idx2]
56      cv.AbsDiff(buf[idx1], buf[idx2], silh) # get difference
            between frames

58      cv.Threshold(silh, silh, diff_threshold, 1, cv.
            CV_THRESH_BINARY) # and threshold it
        cv.UpdateMotionHistory(silh, mhi, timestamp,
            MHI_DURATION) # update MHI
60      cv.CvtScale(mhi, mask, 255./MHI_DURATION,
                    (MHI_DURATION - timestamp)*255./
                        MHI_DURATION)
62      cv.Zero(dst)
        cv.Merge(mask, None, None, None, dst)
64      cv.CalcMotionGradient(mhi, mask, orient, MAX_TIME_DELTA
            , MIN_TIME_DELTA, 3)
        if not storage:
66          storage = cv.CreateMemStorage(0)
        seq = cv.SegmentMotion(mhi, segmask, storage, timestamp
            , MAX_TIME_DELTA)
68      for (area, value, comp_rect) in seq:
            if comp_rect[2] + comp_rect[3] > 100: # reject very
                small components
70              color = cv.CV_RGB(255, 0,0)
                silh_roi = cv.GetSubRect(silh, comp_rect)
72              mhi_roi = cv.GetSubRect(mhi, comp_rect)
                orient_roi = cv.GetSubRect(orient, comp_rect)
74              mask_roi = cv.GetSubRect(mask, comp_rect)
```

```
                angle = 360 - cv.CalcGlobalOrientation(
                    orient_roi, mask_roi, mhi_roi, timestamp,
                    MHI_DURATION)
76
                count = cv.Norm(silh_roi, None, cv.CV_L1, None)
                    # calculate number of points within
                    silhouette ROI
78              if count < (comp_rect[2] * comp_rect[3] * 0.05)
                    :
                    continue
80
                magnitude = 30.
82              center = ((comp_rect[0] + comp_rect[2] / 2), (
                    comp_rect[1] + comp_rect[3] / 2))
                cv.Circle(dst, center, cv.Round(magnitude*1.2),
                    color, 3, cv.CV_AA, 0)
84              cv.Line(dst,
                        center,
86                      (cv.Round(center[0] + magnitude * cos(
                            angle * cv.CV_PI / 180)),
                         cv.Round(center[1] - magnitude * sin(
                            angle * cv.CV_PI / 180))),
88                      color,
                        3,
90                      cv.CV_AA,
                        0)
92          self.last=last
            self.mhi=mhi
94          self.storage=storage
            self.mask=mask
96          self.orient=orient
            self.segmask=segmask
98          self.counter=counter
            self.buf=buf
```