Advanced Vision

Assignment 2 Report

(s0817025) - L. Brown
(s0786036) - R.S. Thomson

9 March 2012

Implementation

The implementation of the system was separated into 5 different computational tasks:

- Detecting the hand object.
- Detecting the bounding box around the object.
- Generating the correct Motion History Image (MHI) for the sequence of images.
- Computing the Moment Invariant Descriptors (MID) by calculating complex moments for the MHI.
- Training a Bayesian classifier to classify unseen hand gesture data into one of the three classes: paper, scissors, or rock.

Below is an explanation of the algorithms used at each stage in order to achieve this. ‘Frame’ and ‘image’ are used synonymously throughout this paper.

Detecting the hand

Given at least one background image and a sequence of images representing the recording sequence of a specific hand motion, the task was to isolate the hand from the background. In order to obtain the image of the hand from the background pattern, a modification of the existing background removal was employed.

For each image in a video sequence a simple background removal algorithm was applied. For a given frame, f, and a background image, b, from a set of background images, B, the background is removed from f by thresholding the corresponding difference values between f and b. This produces a set of binary images, corresponding to the number of background frames, that are further combined using a logical AND to find the intersection of these images. This results in only considering pixels that are present in f and \forall b \element B, removing all visible background elements for a particular frame.

Once the background has been removed for a frame a form of cleanup is performed by eroding and dilating the regions within the image, removing all blobs within the image with an area less than ~500 pixels. This reduces the amount of noise that remains within the binary images obtained from the background removal algorithm.

Background removal is performed for all images for a given image sequence.

Detecting the sequence bounding box

To obtain the bounding box of the hand from a sequence of images a union is taken of all the resulting images from background removal through a logical OR operation applied to each image in the sequence. The bounding box the resulting binary image is then calculated by iterating through each pixel and finding the min-x, min-y, max-x, and max-y values for the image that contain a pixel with a value of 1 or more. These values then correspond to the overall bounding box for the image, such that the image can be cropped to these bounds without losing any object data.

Generating the Motion History Image

For each image within the sequence of size N, starting from the first image in the sequence and finishing on the last image, each frame is iteratively added to a compiled image – the MHI. The values of the pixels within each frame are first multiplied by i/N where i is the current frame number ranging from 1 to N, before being added to the resultant image.

Once the algorithm has iterated through each frame, a single grey-scale image is produced where the value of each pixel, ranging from 1/N to 1, is representative of the pixel's recency within the sequence. Pixels which appeared in later frames have a value closer to one and therefore appear brighter than those pixels which appeared in earlier frames, which have values that tend towards 1/N depending of their relative recency.

Generating the Moment Invariants for the Motion History Image

Using a modification of the description for complex moments it was possible to calculate the Moment Invariant Descriptors (MIDs) for the MHI by weighting the moments by the density values of the image. The center of mass is calculated in a normal fashion, though each pixel value is weighted by it's density (greyscale value). This produces a center of mass that favours the most recent frame.

The area of an MHI is calculated using a similar method, where the weight for a pixel is determined by the greyscale density of that pixel. Again, the most recent frame in the sequence has a greater impact on the overall area of MHI.

Since the area and center of mass are necessary when calculating the complex moments of an image it is possible to account for the density information, provided in the MHI, when calculating the MIDs. A set of six complex moments and the compactness of the MHI are calculated and returned as a feature vector for a given MHI.

This feature vector is then used as the training vector for a Naive Bayes classifier.

Classification with Naive Bayes

Using MATLAB's in-built Naive Bayes class a classifier of this type was trained using the feature vectors calculated from the MHI of each image sequence, along with the corresponding class for each feature vector. This Naive Bayes model was then used to predict the class of a feature vector for a previously unseen image sequence.

9.7030242e+00	3.4003276e-01	1.2339728e+01	-5.9887102e+00	-2.8233109e+00	-9.2825042e+01	3.8270495e+01
9.5618959e+00	7.0115420e-01	2.8346911e+02	7.2633875e+02	6.5062181e+01	6.3604219e+03	2.2016880e+04
1.1257609e+01	5.5422630e-01	1.2271954e+02	2.1221446e+02	1.8480090e+02	7.4338385e+03	9.2872073e+03
2.4039274e+01	6.6577506e-01	2.6179309e+02	1.1898461e+03	4.2501541e+02	6.5503327e+04	1.2081977e+04
1.3620593e+01	3.1298249e-01	3.3975191e+01	5.9840631e+01	7.1991399e+00	1.0170271e+03	2.8083325e+02
6.3929084e+00	6.1565553e-01	2.1364026e+02	8.0237230e+02	2.7960503e+02	3.7032315e+04	1.1882398e+04
5.1066811e+00	5.5086273e-01	1.4125996e+02	4.2260344e+02	1.9278997e+02	1.5339691e+04	6.4752680e+03
4.9407101e+00	5.5191811e-01	1.5477158e+02	5.1040528e+02	1.9028536e+02	1.9432648e+04	6.0038734e+03
1.1140508e+01	5.3856992e-01	7.3659914e+01	2.7245811e+01	6.4784479e+01	3.2139253e+01	2.4042140e+03
1.0802196e+01	7.5067118e-01	3.3600399e+02	1.2177207e+03	3.5747688e+02	5.6141164e+04	3.3778904e+04
1.0342646e+01	7.0257888e-01	1.3786353e+02	6.9399831e+02	3.7205407e+02	1.7915984e+04	2.9707001e+03
1.4300009e+01	4.3071997e-01	1.1839798e+02	3.1751606e+02	-8.8030128e+00	1.1598274e+04	9.5246207e+02
1.0065744e+01	3.2794530e-01	4.1290157e+01	7.7641262e+01	3.4410367e+00	1.2353341e+03	3.7157027e+02
7.9606490e+00	5.1305686e-01	8.5155950e+01	1.9458521e+02	1.2841812e+02	4.4558955e+03	1.6763508e+03
5.2266485e+00	4.5698977e-01	4.5476024e+01	7.6044821e+01	8.1420682e+01	1.3758691e+03	7.2630109e+02
8.8068092e+00	5.4852785e-01	1.2514534e+02	3.4932201e+02	1.6195443e+02	9.9730878e+03	2.9823734e+03
3.3856017e+00	4.0343711e-01	1.5783588e+01	-4.9009292e+00	2.6427015e+01	4.0451591e+01	1.6589839e+02
3.5441399e+00	4.6730624e-01	3.8264693e+01	5.6080681e+01	7.3005611e+01	8.2439961e+02	4.4996874e+02
3.7705177e+00	6.9270841e-01	2.7424156e+02	1.1097161e+03	3.6277601e+02	5.3598102e+04	1.9891292e+04
9.7274130e+00	2.5507625e-01	1.2564915e+01	1.7146993e+01	-2.7789798e-01	1.4436338e+02	1.5142000e+01
7.1124698e+00	2.8857750e-01	2.4745174e+01	3.7547439e+01	2.7349219e+00	4.1102088e+02	1.6153193e+02
5.9801380e+00	6.6770008e-01	3.1472506e+02	1.3512642e+03	2.4739146e+02	7.0970425e+04	1.8869177e+04
4.2895652e+00	4.9662968e-01	9.0638540e+01	2.5262215e+02	1.3314228e+02	6.5692708e+03	1.8913934e+03
5.7820216e+00	5.8260340e-01	1.7704974e+02	6.5883425e+02	2.4098095e+02	2.5763956e+04	5.4780634e+03

Table 1: *Feature vectors for all data sets.*

Results

The classifier was trained using MHIs calculated from 4 different subsets of the images sequences, of size n. The whole sequence ([1,N]), the first half of a sequence ([1,n/2]), the mid half of a sequence ([n/4, 3n/4]), and the mid third of a sequence ([n/3, 2n/3]). These results are calculated over all 512 different combinations of training and test sets.

Sequence (size n)	**Paper Accuracy (%)**	**Scissors Accuracy (%)**	**Rock Accuracy (%)**	**Overall (%)**
[1,n]	10.5469	38.2812	39.0625	29.2969
[1, n/2]	67.1875	27.9297	0.0	31.7057
[n/4, 3n/4]	46.875	14.0625	71.875	44.2708
[n/3, 2n/3]	62.5	52.9297	14.0625	43.1641

Table 2: *Result comparison.*

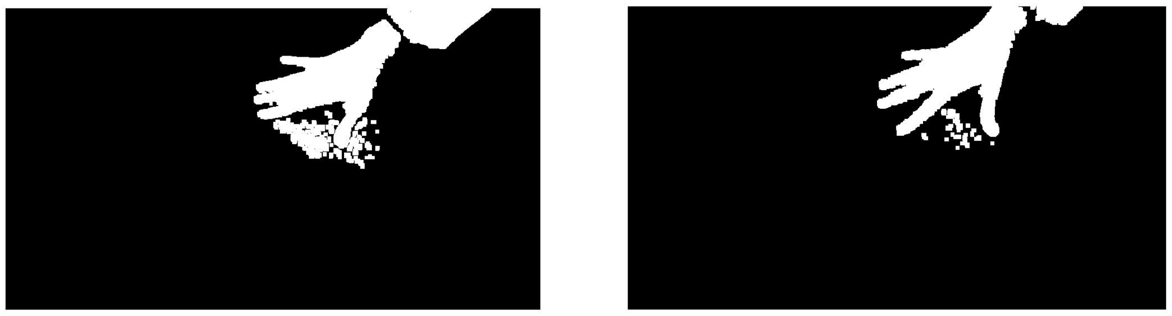

Figure 1: *Sample images from background removal.*

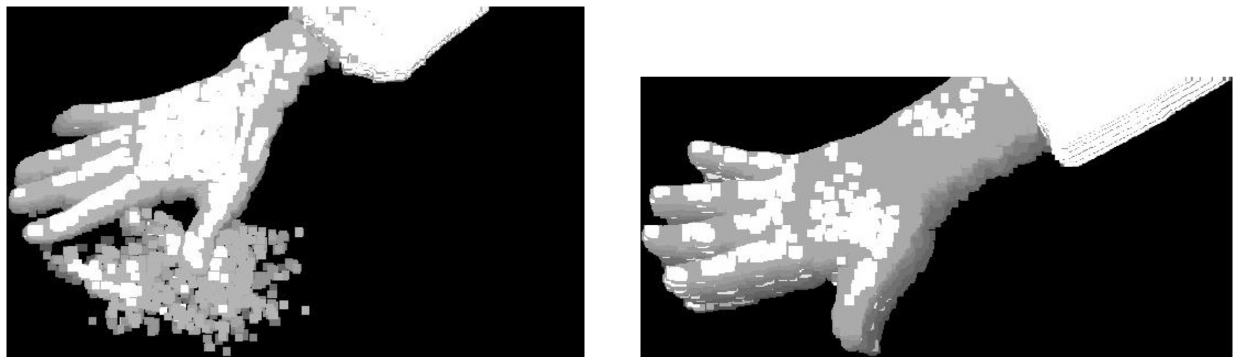

Figure 2: *Sample Motion History Images.*

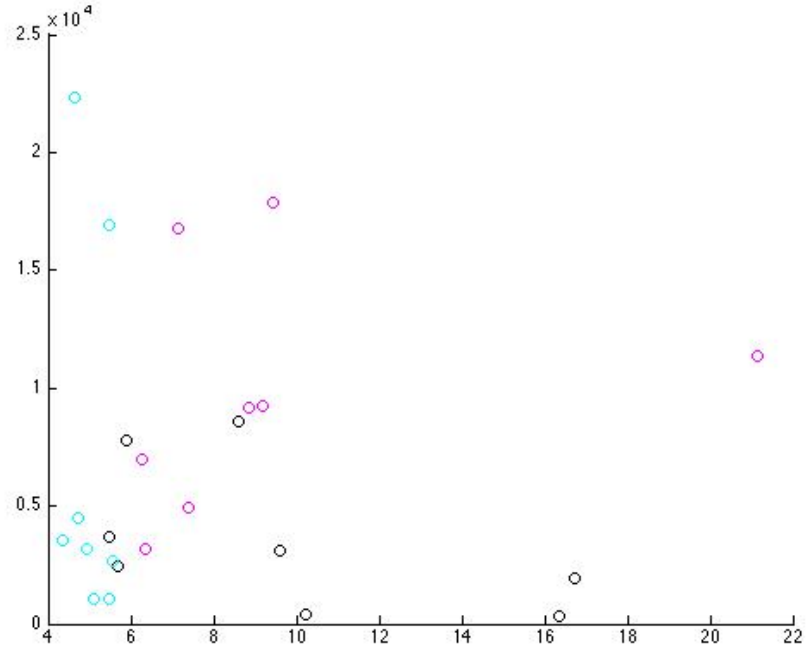

Figure 3: *Scatter plot for moments 1 and 3.*
Key: Black=Paper, Magenta=Scissors, Cyan=Rock

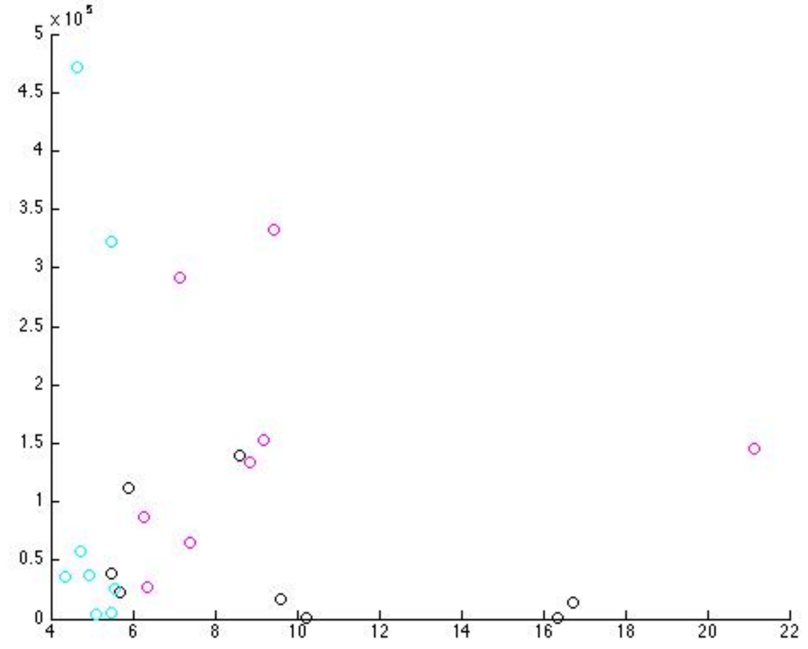

Figure 4: *Scatter plot for moments 1 and 4.*
Key: Black=Paper, Magenta=Scissors, Cyan=Rock

Overall Performance and Analysis

Table 2 shows the performance of the classifier with the given training features. Immediately, it is possible to see that using all frames from a given image sequence results in poor classification accuracy. Given that the MHI gives the most weight to the last frame in a sequence it is important to ensure the final frames are highly representative of the hand gesture class. Brief observation of the images sequences shows that the first and last few frames are images of the hand moving into and out of the view of the camera, rather than gesturing one of rock, paper or scissors.
The next set of tests were run using the first half of each image sequence, where it is expected that the fully gestured hand position is given the most weight when generating the MHI and subsequently when calculating the MID feature vectors. Results differ greatly between each of the three classes, possibly as a result of the first few frames incorrectly contributing to the classification.
Furthermore, it is possible to reduce the image sequence to the same number of images as the previous test while focusing on the median images in the sequence, where these are likely representative of the hand gesture while ignoring the hand moving in and out of camera view. The accuracy gain is noticeable, especially with respect to the Rock class, even though the overall accuracy is not hugely higher than before. However, the reason for this is due to poor accuracy when classifying a hand gesturing scissors.
It may be possible to further reduce the effect of these outlier frames by further reducing the subset around the central frame. For the mid third of frames there is a clear improvement for scissors, while paper accuracy remains relatively unchanged. The accuracy for rock, however, falls by some significant degree as a result. This may be due to important features being lost when cutting the out the extra frames.
The success of the system for each class varies greatly between the subset chosen when generating the MHI, however it should be possible to isolate the strengths and weaknesses of each subset choice with further analysis.

Figure 1 shows a couple of examples of the background removal process. It can be observed that, although the background is successfully removed in most cases, there still remains some residual noise escaping the process. It is possible to correct this with further erosion and dilation, though useful information may be lost in this process.

Figure 2 shows a couple of examples of the Motion History Images. It is possible to see why weighting the median frame is important for classification. The downfall here is that the arm heavily influences the calculation of the feature vectors and subsequently the classification.

One further way to observe classification differences would be to employ a different classification technique than that of Naive Bayes. For example, using a Hidden Markov Model (HMM) to probabilistically weight states for each class may results in a most robust classifier. This would require feature vectors to be obtained from each frame in an image sequence rather than combining the frames into the MHI, to account for similarities in frame-to-frame motion between training and test data.

Appendix

1. main.m

```
% This is a support function that calls everything we need.
function main(fname)
  bg = imread('background1.jpg');
  bg2 = imread('background2.jpg');
  bgs = { bg bg2 };

  % define all the individual sets (no longer training).
  train_sets_paper = { '1-1/' '1-2/' '1-3/' '2-2/' '2-5/' '3-7/' '3-8/' '3-9/' }; % test using '3-9/' };
  train_sets_scissors = { '1-7/' '1-8/' '1-9/' '2-1/' '2-4/' '3-4/' '3-5/' '3-6/' }; % test using '3-6/' };
  train_sets_rock = { '1-4/' '1-5/' '1-6/' '2-3/' '2-6/' '3-1/' '3-2/' '3-3/' }; % test using '3-3/' };

  % define the lists to store the moment invariant descriptors
  p_moms = [];
  s_moms = [];
  r_moms = [];

  % define what we need for the Naïve Bayes classifier
  naive_training = [];

  sizey = size(train_sets_paper);
  tnum = sizey(2); % 7

  % these are the proceedures for generating the moment invariant
  % descriptors for all the training images.
  % what we have done is run this once for all image sets and stored the
  % outcome in a file, saving us calculating this every time we run.
  % for paper
  for i = 1 : tnum
    im_dir = strcat('av212data/train/', train_sets_paper{i});
    images = removeBackgroundFromImageSet(bgs, im_dir);

    [l u r d] = getSequenceBoundingBox(images);
    mhi = createMHI(images, [l u r d]);
    mhi_moms = getMomentInvDesc(mhi);

    p_moms = [p_moms ; mhi_moms];
    naive_training = [naive_training ; mhi_moms];
    pp = i % this gives us an idea of how long things are taking
  end

  pp = 'Done with paper'

  % for scissors
  for i = 1 : tnum
    im_dir = strcat('av212data/train/', train_sets_scissors{i});
    images = removeBackgroundFromImageSet(bgs, im_dir);

    [l u r d] = getSequenceBoundingBox(images);
    mhi = createMHI(images, [l u r d]);
    mhi_moms = getMomentInvDesc(mhi);

    s_moms = [s_moms ; mhi_moms];
    naive_training = [naive_training ; mhi_moms];
    pp = i
  end

  pp = 'Done with scissors'

  % for rock
  for i = 1 : tnum
    im_dir = strcat('av212data/train/', train_sets_rock{i});
    images = removeBackgroundFromImageSet(bgs, im_dir);

    [l u r d] = getSequenceBoundingBox(images);
    mhi = createMHI(images, [l u r d]);
    mhi_moms = getMomentInvDesc(mhi);

    r_moms = [r_moms ; mhi_moms];
    naive_training = [naive_training ; mhi_moms];
    pp = i
  end

  pp = 'Done with rock'

  % fname = naive_training.firsthalf.txt
  % only needed to do this once, or the image processing is changed.
  save 'naive_training.midthird.txt' naive_training '-ASCII';

  pp = fname

end
```

2. removeBackgroundFromImageSet.m

```
function images = removeBackgroundFromImageSet( bgImages, imageDir )

    imagefiles = dir(strcat(imageDir,'*.jpg'));  % collect all images in specified directory
    nfiles = length(imagefiles);    % number of files found
    images = {};     %init the image array

    for i = 1 : nfiles

       %get the image file
       currentfilename = imagefiles(i).name;
       filepath = strcat(imageDir,currentfilename);
       currentimage = imread(filepath);

       %remove bg
       currentimage = bgremove(currentimage,bgImages);

       %clean image
       currentimage = cleanup(currentimage,3,0,0); %maybe 5,3,0
       %currentimage = bwareaopen(currentimage,4000);

       %store the image
       images{i} = currentimage;
    end

end
```

3. getSequenceBoundingBox.m

```
function [ left, up, right, down ] = getSequenceBoundingBox( images )

    imageSize = size(images{1});
    finalImage = zeros(imageSize(1),imageSize(2));

    n = size(images);

    for i = 1 : n(2)
        image = images{i};
        imageSize = size(image);

        for x = 1 : imageSize(1)
        for y = 1 : imageSize(2)
            if(image(x,y) ~= 0)
                finalImage(x,y) = 1;
            end
        end
        end
    end

    [ left, up, right, down ] = getBoundingBox(finalImage);

end
```

4. createMHI.m

```
function [ finalImage ] = createMHI( images , cropBounds )

    imageSize = size(images{1});
    finalImage = zeros(imageSize(1),imageSize(2));

    n = size(images);
    colour = 1/n(2);

    c = n(2);
    start = round(c/4);
    half = round(c/2);
    finish = round(start + (c/2));
    third = round(c/3);
    twothirds = third*2;

    %
    %n(2) is the number of images
    % for i = n(2)/4 : n(2)/4 + n(2)/2
    % half of the images, excluding first quarter and last quarter
    % for i = 1 : n(2)/2
    % excluding the last half, so the image with the highest weight is the
    % middle frame (likely the full hand sign).
    % for i = third : twothirds
    for i = third : twothirds
        image = images{i};
        imageSize = size(image);

        for y = 1 : imageSize(1)
        for x = 1 : imageSize(2)
            if(image(y,x) ~= 0)
                finalImage(y,x) = (image(y,x) * (colour*i));
            end
        end
        end
    end

    finalImage = imcrop( finalImage, cropBounds );

end
```

5. getMomentInvDesc.m

```
% We have altered getproperties.m to work for our MHI images.
function vec = getMomentInvDesc( Image )

  [H,W] = size(Image);
  area = getMHIArea(Image);
  bwim = createBWImage(Image);
  perim = bwarea(bwperim(bwim,4));

  compactness = perim*perim/(4*pi*area);

  % rescale properties so all have size proportional
  % to image size
  % vec = [4*sqrt(area), perim, H*compactness];

  % Without rotation invariants
  % get scale-normalized complex central moments
  c11 = complexmoment(Image,1,1) / (area^2);
  c20 = complexmoment(Image,2,0) / (area^2);
  c30 = complexmoment(Image,3,0) / (area^2.5);
  c21 = complexmoment(Image,2,1) / (area^2.5);
  c12 = complexmoment(Image,1,2) / (area^2.5);
  %c=[c11,c20,c30,c21,c12]

  % with rotation invariants
  % get invariants, scaled to [-1,1] range
  ci1 = real(c11);
  ci2 = real(1000*c21*c12);
  tmp = c20*c12*c12;
  ci3 = 10000*real(tmp);
  ci4 = 10000*imag(tmp);
  tmp = c30*c12*c12*c12;
  ci5 = 1000000*real(tmp);
  ci6 = 1000000*imag(tmp);

  vec = [compactness,ci1,ci2,ci3,ci4,ci5,ci6];

end
```

6. getBoundingBox.m

```
function [ left, up, right, down ] = getBoundingBox( image )

    imageSize = size(image);

    left = imageSize(1)+1;
    up = imageSize(2)+1;
    right = 0;
    down = 0;

    for y = 1 : imageSize(1)
    for x = 1 : imageSize(2)
        if(image(y,x) == 1)
           if(x < left)
              left = x;
           end
           if(y < up)
              up = y;
           end
           if(x > right)
              right = x;
           end
           if(y > down)
              down = y;
           end
        end
    end
    end

end
```

7. bgremove.m

```
function binimage = bgremove (partim, bgims)

    T = double(min(double(min(min(partim))+(max(max(partim))-min(min(partim)))/2))/255);

    %detect change in red, green and blue values. Makes WxHx3 image
    outims = {};
    n = numel(bgims);
    [H,W] = size(partim);
    for i = 1 : n
```

```
        bgim = bgims(i);
        bgim = cell2mat(bgim);

        outim = zeros(H,W);
        for r = 1 : H
          for c = 1 : W
            if abs(double(bgim(r,c)) - double(partim(r,c))) > T*50
                outim(r,c) = 1;
            end
          end
        end

        outims{i} = outim;
    end

    % Here we are taking the logical AND of all images to find their
    % intersection.
    composite = ones(H,W);
    if (n > 1)
      for i = 1 : n
        im = outims{i};
        for r = 1 : H
          for c = 1 : W
            composite(r,c) = composite(r,c) && im(r,c);
          end
        end
      end
    else
      composite = outims{1};
    end

    %combine red, green and blue elements detected to make one black and white image
    [H,W] = size(composite);
    outim2 = zeros(H,W/3);
    for r = 1 : H
      for c = 1 : W/3;
        outim2(r,c) = double(composite(r,c)+composite(r,c+W/3)+composite(r,c+W*2/3));
      end
    end

    binimage = outim2;
```

8. complexmoment.m

```
% gets a given complex central moment value
function muv = complexmoment(Image,u,v)

  [rbar cbar] = centerofmass(Image);
  [H,W] = size(Image);

  c_uv_sum = 0;

  for r = 1 : H
    for c = 1 : W
      c1 = complex(r - rbar, c - cbar);
      c2 = complex(r - rbar, cbar - c);
      to_sum = c1^u * c2^v * Image(r,c);
      c_uv_sum = c_uv_sum + to_sum;
    end
  end

  muv = c_uv_sum;
end
```

9. centerofmass.m

```
function [rbar cbar] = centerofmass(im)
    [H,W] = size(im);

    % r: the sum of the height
    % c: the sum of the width
    r = 0;
    c = 0;

    area = 0;

    % Calculate
    for x = 1 : W
        for y = 1 : H
           if (im(y,x) > 0)
             r = r + y;
             c = c + x;
             area = area + im(y,x);
           end
        end
    end

    % Get the averages (dividing by the area) and return them.
    rbar = round(r/area);
    cbar = round(c/area);
    com = [rbar cbar];
end
```

10. getMHIArea.m

```
function area = getMHIArea( im )
  [H,W] = size(im);
  area = 0;

  for x = 1 : W
      for y = 1 : H
         if (im(y,x) > 0)
           area = area + im(y,x);
         end
      end
  end
end
```

11. maintest.m

```
function [ ptr str rtr ]  = maintest( pt, st, rt, fname)
%This function performs all the same steps as main.m
% Though here there are additions to cover iterating through all training
% and perform all tests.
% The inputs:
% pt, st, rt
% are the test numbers to be used for pt_num, st_num and rt_num.
% The outputs are:
% ptr: paper_test_result
% str: scissors_test_result
% rtr: rock_test_result

  % define all the individual sets (no longer training).
  train_sets_paper = { '1-1/' '1-2/' '1-3/' '2-2/' '2-5/' '3-7/' '3-8/' '3-9/' };
  train_sets_scissors = { '1-7/' '1-8/' '1-9/' '2-1/' '2-4/' '3-4/' '3-5/' '3-6/' };
  train_sets_rock = { '1-4/' '1-5/' '1-6/' '2-3/' '2-6/' '3-1/' '3-2/' '3-3/' };

  % define what we need for the Naïve Bayes classifier
  naive_training = [];
  naive_classes = {};
  % fname = 'naive_training.txt';
  % load the pre-calculated moment invariant descriptors
  all_moms = importdata(fname);
  p_test = [];
  s_test = [];
  r_test = [];
  pt_num = pt;
  st_num = st;
  rt_num = rt;

  % repopulate the moms and easily choose test mom
  for i = 1 : 8
    mom = [all_moms(i,1) all_moms(i,2) all_moms(i,3) all_moms(i,4) all_moms(i,5) all_moms(i,6) all_moms(i,7)];
    if (i==pt_num)
      p_test = mom;
    else
      naive_training = [naive_training ; mom];
    end
  end
  for i = 9 : 16
    mom = [all_moms(i,1) all_moms(i,2) all_moms(i,3) all_moms(i,4) all_moms(i,5) all_moms(i,6) all_moms(i,7)];
    if (i==(st_num+8))
      s_test = mom;
    else
      naive_training = [naive_training ; mom];
    end
  end
  for i = 17 : 24
    mom = [all_moms(i,1) all_moms(i,2) all_moms(i,3) all_moms(i,4) all_moms(i,5) all_moms(i,6) all_moms(i,7)];
    if (i==(rt_num+16))
      r_test = mom;
    else
      naive_training = [naive_training ; mom];
    end
  end

  % there are 7 of each class used to train the classifier
  for i=1 : 21
    if (i<=7)
      naive_classes{i} = 'paper';
    end
    if (i>7 && i<=14)
      naive_classes{i} = 'scissors';
    end
    if (i>14)
      naive_classes{i} = 'rock';
    end
  end

  nc = naive_classes';

  % create a naive bayes classifier and fit our training data.
  nb = NaiveBayes.fit(naive_training, nc);

  % make predictions on our test data.
  pt = nb.predict(p_test);
  st = nb.predict(s_test);
  rt = nb.predict(r_test);

  % we return the Naïve Bayes classifier for test purposes.
  ptr = pt;
  str = st;
  rtr = rt;

end
```

12. runtests.m

```
function runtests(dname,fname)
  % Here we run the classifier for all data and record the data to a file.

  %dname ='results_firsthalf/';
  %fname = 'filename of training data';

  %For all paper train/test combinations
  for i = 1 : 8

    % For all scissors train/test combinations
    for j = 1 : 8

      % For all rock train/test combinations
      for k = 1 : 8

        [ ptr str rtr ] = maintest(i,j,k,fname);

        filename = strcat(dname,int2str(i), '_', int2str(j), '_', int2str(k), '.txt');
        fileID = fopen(filename,'w');
        fprintf(fileID,'%s\n', ptr{1});
        fprintf(fileID,'%s\n', str{1});
        fprintf(fileID,'%s\n', rtr{1});
        fclose(fileID);

      end
    end
  end
end
```

13. evaluate.m

```
function evaluate(dname)
  % Here we run the classifier for all data and record the data to a file.

  paper_count = 0;
  scissors_count = 0;
  rock_count = 0;

  %dname = 'results_firsthalf/';

  %For all paper train/test combinations
  for i = 1 : 8

    % For all scissors train/test combinations
    for j = 1 : 8

      % For all rock train/test combinations
      for k = 1 : 8

        filename = strcat(dname,int2str(i), '_', int2str(j), '_', int2str(k), '.txt');
        fileID = fopen(filename,'r');
        results = textscan(fileID, '%s', 3);

        ptr = results{1}{1};
        str = results{1}{2};
        rtr = results{1}{3};

        TF = strcmpi(ptr,'paper');
        paper_count = paper_count + TF;

        TF = strcmpi(str,'scissors');
        scissors_count = scissors_count + TF;

        TF = strcmpi(rtr,'rock');
        rock_count = rock_count + TF;
        fclose(fileID);

      end
    end
  end

        pap_percent = paper_count / 512 * 100
        sci_percent = scissors_count / 512 * 100
        roc_percent = rock_count / 512 * 100

        overall_percent = (paper_count + scissors_count + rock_count) / 1536 * 100
        end
```

13. scattman.m

```
% Build and draw our scatter plots
function scattman( fname )
% Draw a scatter plot for the training data
  all_moms = importdata(fname);

  x_plot = zeros(1,24);
  y_plot = zeros(1,24);
  colour = zeros(24,3);
  area = [];

  num = 24;

  for i = 1 : num
    x_plot(i) = all_moms(i,1);
    y_plot(i) = all_moms(i,4);
  end

  % black
  for i = 1 : 8
    colour(i,1) = 0;%[1 1 0];
    colour(i,2) = 0;
    colour(i,3) = 0;
  end

  % magenta
  for i = 9 : 16
    colour(i,1) = 1;%[1 0 1];
    colour(i,2) = 0;
    colour(i,3) = 1;
  end

  % cyan
  for i = 17 : 24
    colour(i,1) = 0;%[0 1 1];
    colour(i,2) = 1;
    colour(i,3) = 1;
  end

%   I'm the scatman
%   Ski bi di bi di do bap do bap
%   Do ba do bap
  scatter(x_plot,y_plot,area,colour);
%   Repeat after me
%   It's a scoobie oobie doobie scoobie doobie melody
%   Sing along with me
%   It's a scoobie oobie doobie scoobie doobie melody

end
```