```
/* Based on the solution of Andreas Hoffmann */
/* FILE: aufgabe1a.cpp */

#include <stdio.h>
#include <conio.h>
#include <fcntl.h>
#include <io.h>
#include <iostream>
#include <fstream>
#include <string.h>

#include "main.h"

#define RESIZED_WIDTH 3
#define RESIZED_HEIGHT 3
#define HUMAN 1
#define ANIMAL -1
#define NUM_OF_FEATURES RESIZED_WIDTH*RESIZED_HEIGHT

char* path= "C:\\Users\\Krisztian.Buza\\ImageProcessing\\VOC2008-
            data\\data\\VOCtrainval_14-Jul-2008\\VOCdevkit\\VOC2008\\JPEGImages\\";

void aufgabe1a() {

  imageset_Type train_images[] = {
    {"2007_000175.jpg", ANIMAL}, {"2007_000528.jpg", ANIMAL},
    {"2007_001397.jpg", ANIMAL}, {"2007_001299.jpg", ANIMAL},
    {"2007_001960.jpg", ANIMAL}, {"2007_000346.jpg", HUMAN},
    {"2007_000323.jpg", HUMAN}, {"2007_001423.jpg", HUMAN},
    {"2007_002079.jpg", HUMAN}, {"2007_002545.jpg", HUMAN} };
  imageset_Type test_images[] = {
    {"2007_003104.jpg", HUMAN}, {"2007_002845.jpg", ANIMAL},
    {"2007_004969.jpg", ANIMAL}, {"2007_003580.jpg", HUMAN} };

  int num_train_images = sizeof(train_images) / sizeof(imageset_Type);
  int num_test_images = sizeof(test_images) / sizeof(imageset_Type);

  // create matrices for holding data
  CvMat* train_data = cvCreateMat(num_train_images, NUM_OF_FEATURES, CV_32FC1);
  CvMat* response_data = cvCreateMat(num_train_images, 1, CV_32FC1);
  CvMat* test_data = cvCreateMat(num_test_images, NUM_OF_FEATURES, CV_32FC1);


  // populate train data and train classes(responses)
  populateData(train_images, num_train_images, 0, train_data, response_data);

  // populate test data
  populateData(train_images, num_test_images, test_data, 0, 0);

  // apply naive bayes classifier
  CvNormalBayesClassifier bayes;

  // If you want to apply an other classifier, you just have to
  // define the type of "bayes" differently
  // ("bayes" is the name of the variable)
  // CvSVM bayes;

  bayes.train(train_data, response_data);
  CvMat* temp = cvCreateMat(1, NUM_OF_FEATURES, CV_32FC1);

  // classify test images
  for (int p = 0; p < num_test_images; p++) {
   // row by row
   cvGetRow(test_data, temp, p);

   // print out prediction
   char* object_prediction = ((uchar)bayes.predict(temp) == 1)? "HUMAN" : "ANIMAL";
   char* object_intruth = (test_images[p].response == 1)? "HUMAN" : "ANIMAL";
   printf("At the picture %s is an %6s. Computer predicts it is an %6s\n",
       test_images[p].image_name, object_intruth, object_prediction);
```

```
  }

  // cleanup steps
  cvReleaseMat(&train_data);
  cvReleaseMat(&response_data);
  cvReleaseMat(&test_data);

}


// Produces features for train/test data
void populateData(imageset_Type* images, int size, CvMat* test,
CvMat* train, CvMat* responses) {
  int z, i, j, h;
  uchar* hsv_values = NULL;

  for (h = 0; h < size; h++) {
    char* temp = images->image_name;
    char* imageName =
      (char *) calloc(strlen(path) + strlen(temp) + 1, sizeof(char));
    strcat(imageName, path);
    strcat(imageName, temp);

    IplImage* original_img = cvLoadImage(imageName, 1);

    if (!original_img) { return; }

    IplImage* resized_img = cvCreateImage(cvSize(RESIZED_WIDTH,RESIZED_HEIGHT),
        original_img->depth,original_img->nChannels);
    IplImage* hsv_img = cvCreateImage(cvSize(RESIZED_WIDTH,RESIZED_HEIGHT),
        original_img->depth,original_img->nChannels);
    cvResize(original_img, resized_img);
    cvReleaseImage(&original_img);
    cvCvtColor(resized_img, hsv_img, CV_BGR2HSV);
    cvReleaseImage(&resized_img);
    z=0;
    for (i = 0; i < hsv_img->height; i++) {
        for (j = 0; j < hsv_img->width; j++) {
            hsv_values = &CV_IMAGE_ELEM(hsv_img, uchar, i, j*3);
            if (train) cvmSet(train, h, z++, (double)hsv_values[2]);
            if (test) cvmSet(test, h, z++, (double)hsv_values[2]);
            std::cout<<(double)hsv_values[2]<<" ";
            hsv_values = NULL;
        }
    }
    std::cout<<"\n";
    cvReleaseImage(&hsv_img);
    if (responses) cvmSet(responses, h, 0, (double)images->response);
    images++;
  }
}



/* FILE: main.cpp */


#include "main.h"
int main(int arg, char ** argv) {
      aufgabe1a();
      return 0;
}
```

```
/* FILE: main.h */

#ifndef MAIN_H_
#define MAIN_H_
#include "cv.h"
#include "highgui.h"
#include "ml.h"
#include "stdio.h"
typedef struct imageset {
      char* image_name;
      uchar response;h
} imageset_Type;
void aufgabe1a();
void populateData(imageset_Type* images, int size, CvMat* test,
        CvMat* train, CvMat* responses);
#endif /*MAIN_H_*/
```