```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <math.h>

#define LINES 'L'
#define RECTS 'R'
#define PI 3.14159265358979323846

using namespace std;
using namespace cv;

    /*********************************************************************
    *                              EXERCISE 2                            *
    *                              ----------                            *
    *                            BINARY IMAGES                           *
    *********************************************************************/

IplImage * getHistogram(IplImage *, char);
IplImage * getBinImagePlus(IplImage *, unsigned, int *, int *, int *);
IplImage * interactiveThresholding(IplImage *, IplImage *, IplImage *, int *, int *, int
*);
IplImage * simpleThresholding(IplImage *, IplImage *, int *, int *, int *);
void reducedCentralMoments(IplImage *, double, double, double *, double *, double *);

int main(int argc, char ** argv) {

    /*********************************************************************
    *                                PART 1                              *
    *                                ------                              *
    *                            Thresholding                            *
    *********************************************************************/
        /// Uploading and converting to grey scale
    IplImage * img = cvLoadImage(argv[1], 1);
    IplImage * G = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
    cvCvtColor(img, G, CV_RGB2GRAY);

        /// Uploading directly in grey scale
    //IplImage * G = cvLoadImage(argv[1], CV_LOAD_IMAGE_GRAYSCALE);

    ///_____________________________Histogram_____________________________
    ///                DRAW A PRELIMINARY SIMPLE HISTOGRAM

    IplImage * his = getHistogram(G, LINES);

    cvNamedWindow("Grey scale",0);
    cvShowImage("Grey scale", G);

    cvNamedWindow("histogram", 0);
    cvShowImage("histogram", his);

        /// First threshold selection
    printf("\nTake a look at the grey scale image and histogram then press any key.\n");
    cvWaitKey(0);

    ///_____________________Binary Image Formation_____________________________


    /// These are needed in PART 2 to calculate the coordinates of the centre of mass.
    /// Declared now because they can be calculated in parallel with the thresholding.
    int tot = 0, xcm = 0, ycm = 0;

    IplImage * bin;

    ///_________________INTERACTIVE THRESHOLDING - A little unstable...
        /// Grey scale histogram converted to 3-channel image first.
        /// It will be useful later to draw the threshold on the histogram.
    IplImage * Chis = cvCreateImage(cvGetSize(his), IPL_DEPTH_8U, 3);
    cvCvtColor(his, Chis, COLOR_GRAY2RGB);
    cvFlip(Chis, Chis, 0);

    bin = interactiveThresholding(G, Chis, his, &tot, &xcm, &ycm);

    cvSaveImage("histogram.png", Chis);

    ///_____________________SIMPLE THRESHOLDING
    /// Observe grey scale and histogram, select a threshold once, and hope!
    //bin = simpleThresholding(G, his, &tot, &xcm, &ycm);


    /*********************************************************************
    *                                PART 2                              *
    *                                ------                              *
```

```
    *                           Centre of Mass                             *
    ******************************************************************/
        /// Coordinates
    ycm /= tot;
    xcm /= tot;

    printf("\nCenter of mass: (%d , %d)\n", xcm, ycm);

        /// Draw a cross in the picture
    IplImage * I = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 3);
    cvCvtColor(bin, I, COLOR_GRAY2RGB);  // To draw in the binary image
    //cvCvtColor(G, I, COLOR_GRAY2RGB);  // To draw in the grey scale image

        /// Draw a cross in the center of mass
    unsigned short A = 10;
    cvLine(I, cvPoint((xcm - A),(ycm - A)), cvPoint((xcm + A),(ycm + A)),
cvScalar(0,0,255), 2, 4);
    cvLine(I, cvPoint((xcm - A), (ycm + A)), cvPoint((xcm + A),(ycm - A)),
cvScalar(0,0,255), 2, 4);

        /// Just a different style of cross
    //cvLine(I, cvPoint(xcm,(ycm - A)), cvPoint(xcm,(ycm + A)), cvScalar(255,0,255), 2, 4);
    //cvLine(I, cvPoint((xcm - A), ycm), cvPoint((xcm + A), ycm), cvScalar(255,0,255), 2,
4);


    /*********************************************************************
    *                               PART 3                               *
    *                              --------                              *
    *                           Image Moments                           *
    ******************************************************************/
    double M_20 = 0, M_02 = 0, M_11 = 0;
    reducedCentralMoments(bin, xcm, ycm, &M_20, &M_02, &M_11);

        /// Inclination of Principal Axis
    double th = 0.5 * atan2((2 * M_11), (M_20 - M_02));
    double th_deg = (th / PI) * 180;
    printf("\nInclination of main axis (deg): %.2f\n", th_deg);
    double m = tan(th);
    printf("\nAngular Coefficient: %.2f\n", m);

        /// Draw a line at an angle theta, through the centre of mass
    cvLine(I, cvPoint((xcm - 30),(ycm - 30*m)), cvPoint((xcm + 30),(ycm + 30*m)),
cvScalar(255,0,0), 2, 4);

    cvNamedWindow("Center of Mass", 0);
    cvShowImage("Center of Mass", I);

    cvSaveImage("Binary.png", bin);
    cvSaveImage("Center_of_Mass.png", I);

    /*********************************************************************
    *                               PART 4                               *
    *                              --------                              *
    *                        Moment Invanriances                         *
    ******************************************************************/

    //_____________________________________________________________IN CLASS


    ///*********************************************************************
    ///                               Clean-Up                             *
    ///*********************************************************************

    cvWaitKey(0);

    cvReleaseImage(&img);
    cvReleaseImage(&his);
    cvReleaseImage(&G);
    cvReleaseImage(&bin);
    cvReleaseImage(&I);
    cvReleaseImage(&Chis);  // Comment this out if you're doing simple thresholding

    cvDestroyAllWindows();

    return 0;
}



    /*********************************************************************
    * From the pointer to a greyscale image, draws a histogram image and  *
    * returns a pointer to it. Two possible styles can be selected:       *
    * LINES (a line for each bin)                                         *
```

```
    * RECTS (a rectangle for each bin)                                 *
    ************************************************************************/
IplImage * getHistogram(IplImage * gray, char style) {
    int dep = 256;
    int hist[dep];  // Histogram for gray scale image
    int st = 4;     // To separate lines in the histograms, for better look ;)

        /// Initialize histograms
    for (int i = 0; i < dep; i++)
        hist[i] = 0;

        /// Calculate histogram
    for(int i = 0; i < gray->height; i++) {
        char * ptr = gray->imageData + i*gray->widthStep;
        for(int j = 0; j < gray->width; j++) {
                if(*ptr < 0) {
                    char c = *ptr;
                    uchar x = (uchar) c;
                    hist[x] += 1;
                }
                else
                    hist[(unsigned)(*ptr)] += 1;
                ptr++;
        }
    }

        /// Create image for the histograms
    IplImage * his = cvCreateImage(cvSize(st*dep,600), IPL_DEPTH_8U, 1);
    cvSet(his, 0);                      // Initialize image (all black)
    his->origin = IPL_ORIGIN_BL;        // Set the origin in the bottom left corner

        /// Draw a line/Rectangle for each bin
    if (style == LINES)
        for (int i = 0; i < dep; i++)
            if (hist[i] != 0)
                cvLine(his, cvPoint(i*st, 0), cvPoint(i*st, hist[i]/10), 150, 1, 4);
    else if (style == RECTS)
        for (int i = 0; i < dep; i++)
            if (hist[i] != 0)
                cvRectangle(his, cvPoint(i*st,hist[i]/10), cvPoint((i+1)*st,0), 150, -1,
4);
    else {
        printf("\nError. No style selected.\n");
        cvReleaseImage(&his);
        return NULL;
    }

    return his;
}


    /************************************************************************
    * Input: a pointer to a GREY SCALE image, a threshold, 3 pointers to     *
    * integer for the necessary computation to determine the center of mass. *
    * Forms a binary image, counts the number of pixels set to 0, sums       *
    * the values of the 2 coordinates for the pixels set to 0 (used later to *
    * calculate the center of mass coordinates).                             *
    *                                                                        *
    * PS. It is possible to calculate the coordinates even in this function  *
    * but it is done later to respect the structure of the exercise          *
    ************************************************************************/
IplImage * getBinImagePlus(IplImage * img, unsigned t, int * tot, int * xcm, int * ycm) {
    IplImage * I = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
    for(int i = 0; i < img->height; i++) {
        char * ptr = img->imageData + i*img->widthStep;
        char * p = I->imageData + i*I->widthStep;
        for(int j = 0; j < img->width; j++) {
                if ((uchar)(*ptr) >= (uchar)t)
                    *p = 255;
                else {
                    // In this case we are on the object
                    *p = 0;
                    (*tot)++;
                    (*ycm) += i;
                    (*xcm) += j;
                }
            ptr++;
            p++;
        }
    }
    return I;
}
```

```
247     /*************************************************************************
248     * Interactively allows to select a threshold and see the position in a   *
249     * histogram. When the user is satisfied it shows the binary image built  *
250     * with the selected threshold and if the result is not satisfying allows *
251     * to select a new threshold again. It returns a pointer to the final     *
252     * binary image that the used has confirmed as ultimately satisfactory.   *
253     *                                                                        *
254     * The function needs:                                                    *
255     *    - a pointer to a GREY SCALE image to apply the threshold to;        *
256     *    - a pointer to a preliminary histogram image (necessary to restore  *
257     *      the histogram when changing the threshold);                       *
258     *    - a pointer to a 3-channel image for the histogram to be updated    *
259     *      when a new threshold is selected (it must be passed as input      *
260     *      because the function must return another pointer);                *
261     *    - 3 pointers to int to compute the center of mass coordinates       *
262     *************************************************************************/
263
264 #define NEW 'N'
265 #define OK 'O'
266
267 IplImage * interactiveThresholding(IplImage * G, IplImage * Chis, IplImage * his, int
    *tot, int *xcm, int *ycm) {
268     unsigned int t = 0;     // Threshold
269     char st = 4;            // To place correctly the threshold line in the image
270     IplImage * bin;         // To contain the binary image
271
272     printf("\nInsert Threshold: ");
273     scanf("%u", &t);
274
275     while(1) {
276         while(1) {
277             if (!(t >= 0 && t <= 255)) {
278                 printf("\nInvalid value!\n");
279                 printf("\nInsert Threshold: ");
280                 scanf("%u", &t);
281                 continue;
282             }
283             cvCvtColor(his, Chis, COLOR_GRAY2RGB);
284             cvFlip(Chis, Chis, 0);
285             cvLine(Chis, cvPoint(t*st, 0), cvPoint(t*st, 600), cvScalar(0,0,255), 2, 4);
286             cvShowImage("histogram", Chis);
287             printf("\nPress 'O' to continue or 'N' to select a new threshold (in the
    histogram image window)\n");
288             int c = cvWaitKey(0);
289             if (c == OK)
290                 break;
291             else if (c == NEW) {
292                 printf("\nInsert Threshold: ");
293                 scanf("%u", &t);
294                 continue;
295             }
296             else {
297                 printf("\nERROR! Press 'O' or 'N' next time.\n");
298                 printf("\nInsert Threshold: ");
299                 scanf("%u", &t);
300                 continue;
301             }
302         }
303         bin = getBinImagePlus(G, t, tot, xcm, ycm);  // Note: in this context, tot, xcm,
    vcm are pointers
304         cvNamedWindow("Binary Image", 0);
305         cvShowImage("Binary Image", bin);
306         printf("\nPress 'O' if you are satisfied, or 'N' to select a new threshold (in the
    binary image window)\n");
307         int k = cvWaitKey(0);
308         if (k == OK)
309             break;
310         else if (k == NEW) {
311             printf("\nInsert Threshold: ");
312             scanf("%u", &t);
313             continue;
314         }
315         else {
316             printf("\nERROR! Press 'O' or 'N' next time.\n");
317             printf("\nInsert Threshold: ");
318             scanf("%u", &t);
319             continue;
320         }
321     }
322     return bin;
323 }
324
325     /*************************************************************************
326     * This function allows the user to select a threshold and draws a new    *
```

```
    * histogram with a visual indication of the threshold. At the same time  *
    * computes the necessary numbers for the centre of mass.                  *
    *                                                                         *
    * The function needs:                                                     *
    *    - a pointer to a GREY SCALE image to apply the threshold to;         *
    *    - a pointer to a preliminary histogram image (starting point to      *
    *      draw the new histogram with the threshold);                        *
    *    - 3 pointers to int to compute the center of mass coordinates        *
    ***************************************************************************/
IplImage * simpleThresholding(IplImage * G, IplImage * his, int *tot, int *xcm, int
*ycm) {
    unsigned int t = 0;    // Threshold
    char st = 4;           // To place correctly the threshold line in the image
    IplImage * bin;        // To contain the binary image

    printf("\nInsert Threshold: ");
    scanf("%u", &t);

    IplImage * Chis = cvCreateImage(cvGetSize(his), IPL_DEPTH_8U, 3);
        /// Represent a grey scale histogram in a 3-channel image.
        /// It will be useful later to draw the threshold on the histogram.
    cvCvtColor(his, Chis, COLOR_GRAY2RGB);
    cvFlip(Chis, Chis, 0);

    cvLine(Chis, cvPoint(t*st, 0), cvPoint(t*st, 600), cvScalar(0,0,255), 2, 4);
    cvShowImage("histogram", Chis);
    cvSaveImage("histogram.png", Chis);
    cvReleaseImage(&Chis);

    bin = getBinImagePlus(G, t, tot, xcm, ycm);
    cvNamedWindow("Binary Image", 0);
    cvShowImage("Binary Image", bin);

    return bin;
}


    /***************************************************************************
    * Given a pointer to a binary image and the coordinates of the centre of *
    * of mass, and 3 pointers to location where the results will be stored    *
    * it calculates the reduced central moments.                              *
    * It prints out the central moments as well as the reduced central ones. *
    ***************************************************************************/
void reducedCentralMoments(IplImage * bin, double xcm, double ycm, double * M_20, double
* M_02, double * M_11) {
    long tot = 0;
    double b, r;

    for(int i = 0; i < bin->height; i++) {
        char * p = bin->imageData + i*bin->widthStep;
        for(int j = 0; j < bin->width; j++) {
            if ((uchar)(*p) == 0) {
                tot++;
                b = j - xcm;
                r = pow(b, 2);
                *M_20 += r;
                b = i - ycm;
                r = pow(b, 2);
                *M_02 += r;
                *M_11 += (i - ycm) * (j - xcm);
            }
            p++;
        }
    }

    printf("\nCentral Moments:\n\tm_11 = %.2f\n\tm_02 = %.2f\n\tm_20 = %.2f\n", *M_11,
*M_02, *M_20);

        /// Reduced Central Moments
    *M_02 /= tot;
    *M_20 /= tot;
    *M_11 /= tot;

    printf("\nReduced Central Moments:\n\tm_11 = %.2f\n\tm_02 = %.2f\n\tm_20 = %.2f\n",
*M_11, *M_02, *M_20);
}
```