ANEXO 5

```cpp
//*********************************************************************//
//     ANEXO 5. Programa de Tratamiento de Visión Artificial           //
//*********************************************************************//

// Librerias C++ generales
#include <cstdio>
#include <stdlib.h>
#include <iostream>
#include <stdio.h>
#include <ctype.h>
#include <Windows.h>
#include <fstream>
#include <vector>
#include <math.h>

// Librerias OpenCv generales
#include "opencv2/core/core.hpp"
#include "opencv2\core\utility.hpp"
#include "opencv2\core.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv\cv.hpp"
#include "opencv\cv.h"

// Librerias tratamiento de video
#include "opencv2\videoio.hpp"
#include "opencv2\video\background_segm.hpp"
#include "opencv2/video/tracking.hpp"
#include "opencv2/ml/ml.hpp"

// Librerias para la comunicación con Arduino, se adjunta al final del Anexo 5.
#include "SerialClass.h"

// Variales globales relacionadas con el tratamiento de la imagen
const static int SENSITIVITY_VALUE = 25;
const static int BLUR_SIZE = 10;
#define ESCAPE 27
#define pi 3.14159

// Variable global para recuadrar el movimiento
Rect Recuadro_movimiento = Rect(0, 0, 0, 0);

// Variables globales para definir la zona prohibida
int coordenadaX0 = 0;
int coordenadaY0 = 0;
int coordenadaX1 = 0;
int coordenadaY1 = 0;
int coordenadaX2 = 0;
int coordenadaY2 = 0;
int coordenadaX3 = 0;
int coordenadaY3 = 0;
int coordenadaX4 = 0;
int coordenadaY4 = 0;
int coordenadaX5 = 0;
int coordenadaY5 = 0;
```

```
int contador = 0;
int num_puntos;
int querer_zonaprohibida;

// Definir std y cv para las funciones OpenCv
using namespace std;
using namespace cv;

//*********************************************************************//
// Nombre de la función: conversion()
// Función: Función que transforma los números en formato tecto.
//          Se ejecuta cada vez que se necesite representar coordenadas.
// Entradas: Number, número que se quiere transcribir.
// Salidas: SS, número pasado a texto.
//*********************************************************************//
string conversion(int number)
{
    std::stringstream ss;
    ss << number;
    return ss.str();
}

//*********************************************************************//
// Nombre de la función: on_mouse()
// Función: Función que actúa según la acción realizada por el ratón.
//          En este caso, lee las coordenadas dónde se ha hecho "click"
//          con el raton
// Entradas: event, x, y, flags, *ustc.
// Salidas: coordenadas de los puntos.
//*********************************************************************//
void on_mouse(int event, int x, int y, int flags, void *ustc)
{
    // Si se va a definir la zona prohibida por 3 puntos
    if (num_puntos == 3)
    {
        Point pt3;
        // Cuando pulse el botón derecho del ratón
        if (event == CV_EVENT_LBUTTONDOWN)
        {
            pt3 = Point(x, y);

            // Almacenar primer punto
            if (contador == 0)
            {
                coordenadaX0 = x;
                coordenadaY0 = y;
            }
            // Almacenar segundo punto
            if (contador == 1)
            {
                coordenadaX1 = x;
                coordenadaY1 = y;
            }
            // Almacenar tercer punto
            if (contador == 2)
            {
```

```
                coordenadaX2 = x;
                coordenadaY2 = y;
            }
            contador++;
        }
    }

    // Si se va a definir la zona prohibida por 4 puntos
    if (num_puntos == 4)
    {
        Point pt4;
        // Cuando pulse el botón derecho del ratón
        if (event == CV_EVENT_LBUTTONDOWN)
        {
            pt4 = Point(x, y);

            // Almacenar primer punto
            if (contador == 0)
            {
                coordenadaX0 = x;
                coordenadaY0 = y;
            }
            // Almacenar segundo punto
            if (contador == 1)
            {
                coordenadaX1 = x;
                coordenadaY1 = y;
            }
            // Almacenar tercer punto
            if (contador == 2)
            {
                coordenadaX2 = x;
                coordenadaY2 = y;
            }
            // Almacenar cuarto punto
            if (contador == 3)
            {
                coordenadaX3 = x;
                coordenadaY3 = y;
            }

            contador++;
        }
    }

    // Si se va a definir la zona prohibida por 5 puntos
    if (num_puntos == 5)
    {
        Point pt5;

        if (event == CV_EVENT_LBUTTONDOWN)
        {
            pt5 = Point(x, y);

            if (contador == 0)
            {
```

```
                coordenadaX0 = x;
                coordenadaY0 = y;
            }

            if (contador == 1)
            {
                coordenadaX1 = x;
                coordenadaY1 = y;
            }

            if (contador == 2)
            {
                coordenadaX2 = x;
                coordenadaY2 = y;
            }

            if (contador == 3)
            {
                coordenadaX3 = x;
                coordenadaY3 = y;
            }

            if (contador == 4)
            {
                coordenadaX4 = x;
                coordenadaY4 = y;
            }

            contador++;
        }
    }

    // Si se va a definir la zona prohibida por 6 puntos
    if (num_puntos == 6)
    {
        Point pt6;
        if (event == CV_EVENT_LBUTTONDOWN)
        {
            pt6 = Point(x, y);

            if (contador == 0)
            {
                coordenadaX0 = x;
                coordenadaY0 = y;
            }

            if (contador == 1)
            {
                coordenadaX1 = x;
                coordenadaY1 = y;
            }

            if (contador == 2)
            {
                coordenadaX2 = x;
                coordenadaY2 = y;
```

```
            }

            if (contador == 3)
            {
                coordenadaX3 = x;
                coordenadaY3 = y;
            }

            if (contador == 4)
            {
                coordenadaX4 = x;
                coordenadaY4 = y;
            }

            if (contador == 5)
            {
                coordenadaX5 = x;
                coordenadaY5 = y;
            }
            contador++;
        }
    }
}

//*********************************************************************//
// Nombre de la función: mensaje()
// Función: Función de interacción con el usuario
//          Indica si se quiere indicar una zona prohibida y en caso
//          afirmativo, pide indicar el número de puntos con los cuales
//          se va a definir dicha área.
//          con el raton
// Entradas: ninguna.
// Salidas: mensajes por la pantalla de comandos.
//*********************************************************************//
int mensaje()
{
    puts("¿Quiere introducir una zona prohibida? Pulse 1 en caso afirmativo y 0 en caso negativo.");

    scanf_s("%i", &querer_zonaprohibida);

    if (querer_zonaprohibida == 1)
    {
        puts("Indique el numero de puntos con los que va a limitar la zona prohibida. Min:3 puntos, Max:6 puntos");

        scanf_s("%i", &num_puntos);

        printf_s("Haga click en la imagen en los %i puntos del poliedro que quiere limitar.", num_puntos);
    }

    return 0;
}

//*********************************************************************//
```

```cpp
// Nombre de la función: main()
// Función: Función principal del programa. Tareas que realiza:
//              - Tratamiento previo de imágenes.
//              - Resta de imágenes para detectar el movimiento.
//              - Tratamiento posterior de la resta resultante.
//              - Averiguar si el elemento en movimiento es el robot
//                buscado.
//              - En caso afirmativo, lo bordea e indica su localización.
//              - Analiza si está dentro de la zona prohibida.
//              - En caso afirmativo, envía una señal al Arduino por
//                puerto serie.
// Entradas: ninguna.
// Salidas: señal infrarroja, visualización del tratamiento.
//*********************************************************************//
int main()
{
    //Indicar puerto serie del Arduino
    Serial* Arduino = new Serial("COM3");

    // Matrices donde se guardarán las imágenes que forman del tratamiento
    Mat captura;
    Mat captura_gray;
    Mat frame_fondo;
    Mat frame_fondo_gray;
    Mat diferencia;
    Mat diferencia_editada;

    // Variables para identificar los movimientos
    double perimetro, area;
    int i = 0, clase, k = 0;
    int contadorX = 0;

    // Cargar el video de la webcam
    VideoCapture video_original(1);

    // Verificación de que el video se ha cargado correctamente
    if (!video_original.isOpened())
    {
        cout << "No se encontro camara.\n";
    }

    // Cargar la foto patrón
    frame_fondo = cv::imread("capturaf.png");
    cvtColor(frame_fondo, frame_fondo_gray, COLOR_BGR2GRAY);

    // Espera con el fín de estabilizar la imagen de la cámara
    Sleep(4000);

    // En esta foto indicamos con el ratón la zona prohibida
    namedWindow("ZONA PROHIBIDA");
    imshow("ZONA PROHIBIDA", frame_fondo);

    mensaje();

    // Si se quiere indicar zona prohibida, se dibuja sobre la imagen patrón
    if (querer_zonaprohibida == 1)
```

```
    {
        // Si se define el área con 3 puntos
        if (num_puntos == 3)
        {
            CvPoint Puntos_coordenadas[3];
            setMouseCallback("ZONA PROHIBIDA", on_mouse, 0);

            while (contador != 3)
            {
                waitKey(1);
            }

            destroyAllWindows();

            // Se dibuja la zona prohibida en la imagen patrón
            Puntos_coordenadas[0].x = coordenadaX0;
            Puntos_coordenadas[0].y = coordenadaY0;

            Puntos_coordenadas[1].x = coordenadaX1;
            Puntos_coordenadas[1].y = coordenadaY1;

            Puntos_coordenadas[2].x = coordenadaX2;
            Puntos_coordenadas[2].y = coordenadaY2;

            line(frame_fondo, Puntos_coordenadas[0], Puntos_coordenadas[1], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[1], Puntos_coordenadas[2], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[2], Puntos_coordenadas[0], CvScalar(255, 0, 0), 4, 8, 0);

            printf_s("(%i, %i); (%i, %i); (%i, %i);", coordenadaX0, coordenadaY0, coordenadaX1, coordenadaY1, coordenadaX2, coordenadaY2);
            imshow("ZONA PROHIBIDA", frame_fondo);
        }

        // Si se define el área con 4 puntos
        if (num_puntos == 4)
        {
            CvPoint Puntos_coordenadas[4];
            setMouseCallback("ZONA PROHIBIDA", on_mouse, 0);

            while (contador != 4)
            {
                waitKey(1);
            }

            destroyAllWindows();

            // Se dibuja la zona prohibida en la imagen patrón
            Puntos_coordenadas[0].x = coordenadaX0;
            Puntos_coordenadas[0].y = coordenadaY0;

            Puntos_coordenadas[1].x = coordenadaX1;
            Puntos_coordenadas[1].y = coordenadaY1;
```

```
            Puntos_coordenadas[2].x = coordenadaX2;
            Puntos_coordenadas[2].y = coordenadaY2;

            Puntos_coordenadas[3].x = coordenadaX3;
            Puntos_coordenadas[3].y = coordenadaY3;

            line(frame_fondo, Puntos_coordenadas[0], Puntos_coordenadas[1], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[1], Puntos_coordenadas[2], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[2], Puntos_coordenadas[3], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[3], Puntos_coordenadas[0], CvScalar(255, 0, 0), 4, 8, 0);

            printf_s("(%i, %i); (%i, %i); (%i, %i); (%i, %i)", coordenadaX0, coordenadaY0, coordenadaX1, coordenadaY1, coordenadaX2, coordenadaY2, coordenadaX3, coordenadaY3);

            imshow("ZONA PROHIBIDA", frame_fondo);
        }

        // Si se define el área con 5 puntos
        if (num_puntos == 5)
        {
            CvPoint Puntos_coordenadas[5];
            setMouseCallback("ZONA PROHIBIDA", on_mouse, 0);

            while (contador != 5)
            {
                waitKey(1);
            }

            destroyAllWindows();

            // Se dibuja la zona prohibida en la imagen patrón
            Puntos_coordenadas[0].x = coordenadaX0;
            Puntos_coordenadas[0].y = coordenadaY0;

            Puntos_coordenadas[1].x = coordenadaX1;
            Puntos_coordenadas[1].y = coordenadaY1;

            Puntos_coordenadas[2].x = coordenadaX2;
            Puntos_coordenadas[2].y = coordenadaY2;

            Puntos_coordenadas[3].x = coordenadaX3;
            Puntos_coordenadas[3].y = coordenadaY3;

            Puntos_coordenadas[4].x = coordenadaX4;
            Puntos_coordenadas[4].y = coordenadaY4;

            line(frame_fondo, Puntos_coordenadas[0], Puntos_coordenadas[1], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[1], Puntos_coordenadas[2], CvScalar(255, 0, 0), 4, 8, 0);
```

```
            line(frame_fondo, Puntos_coordenadas[2], Puntos_coordenadas[3], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[3], Puntos_coordenadas[4], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[4], Puntos_coordenadas[0], CvScalar(255, 0, 0), 4, 8, 0);

            imshow("ZONA PROHIBIDA", frame_fondo);
        }

        // Si se define el área con 6 puntos
        if (num_puntos == 6)
        {
            CvPoint Puntos_coordenadas[6];
            setMouseCallback("ZONA PROHIBIDA", on_mouse, 0);

            while (contador != 6)
            {
                waitKey(1);
            }

            destroyAllWindows();

            // Se dibuja la zona prohibida en la imagen patrón
            Puntos_coordenadas[0].x = coordenadaX0;
            Puntos_coordenadas[0].y = coordenadaY0;

            Puntos_coordenadas[1].x = coordenadaX1;
            Puntos_coordenadas[1].y = coordenadaY1;

            Puntos_coordenadas[2].x = coordenadaX2;
            Puntos_coordenadas[2].y = coordenadaY2;

            Puntos_coordenadas[3].x = coordenadaX3;
            Puntos_coordenadas[3].y = coordenadaY3;

            Puntos_coordenadas[4].x = coordenadaX4;
            Puntos_coordenadas[4].y = coordenadaY4;

            Puntos_coordenadas[5].x = coordenadaX5;
            Puntos_coordenadas[5].y = coordenadaY5;

            line(frame_fondo, Puntos_coordenadas[0], Puntos_coordenadas[1], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[1], Puntos_coordenadas[2], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[2], Puntos_coordenadas[3], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[3], Puntos_coordenadas[4], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[4], Puntos_coordenadas[5], CvScalar(255, 0, 0), 4, 8, 0);
            line(frame_fondo, Puntos_coordenadas[5], Puntos_coordenadas[0], CvScalar(255, 0, 0), 4, 8, 0);

            imshow("ZONA PROHIBIDA", frame_fondo);
```

```
        }
    }

    // Variable para finalizar la ejecuación de la aplicación al dar a escape
    char keypressed = 0;

    // Mientras no se pulse la tecla ESCAPE, la aplicación se ejecuta
    while (keypressed != ESCAPE)
    {
        // Variables para contornear la zona en movimiento
        Mat copia_diferencia_editada;
        vector< vector<Point> > contorno;
        vector<Vec4i> hierarchy;
        bool movimiento_detectado = false; // Por defecto, no se ha encontrado ningún elemento en movimiento

        // Variables para diferencias contornos
        Mat test_data(10, 2, CV_32F);
        double area_baseMIN, area_baseMAX; // Área base de la aspiradora objetivo

        // Captura de una imagen del video
        video_original.read(captura);

        // Pasar la imagen a escala de grises
        cvtColor(captura, captura_gray, COLOR_BGR2GRAY);

        // Restar ambas imágenes. Puesto que está todo inmovil salvo el robot, la imagen resultante mostrará el movimiento.
        absdiff(captura_gray, frame_fondo_gray, diferencia);

        // La diferencia obtenida se difumina, de forma que el contorno se vuelve más grueso
        blur(diferencia, diferencia_editada, cv::Size(BLUR_SIZE, BLUR_SIZE));

        // Umbralizar
        // Se hace notar más la diferencia obtenida, la cual la se pinta de color blanco (255)
        // Con una sensibilidad inferior a 20 aparece ruido.
        // Con una sensibilidad superior a 20 se reduce el contorno de la diferencia
        threshold(diferencia_editada, diferencia_editada, SENSITIVITY_VALUE, 255, THRESH_BINARY);

        //¿Dónde se ha producido el movimiento?
        // Se copia la parte del video donde sólo se ve el movimiento en blanco
        diferencia_editada.copyTo(copia_diferencia_editada);

        // Se buscan contornos. Serán muy nitidos tras el tratamiento anterior.
        findContours(copia_diferencia_editada, contorno, hierarchy, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);

        // Se indica el número de contornos que hay en la imagen. No sólo el robot tiene por qué estar en movimiento
```

```
        cout << endl << "El número total de objetos en movimiento es: " << contorno.size() << endl;

        //¿Se ha encontrado algun contorno?
        if (contorno.size()>0)
            movimiento_detectado = true;
        else
            movimiento_detectado = false;

        // Si se ha detectado algún contorno, se procede a su estudio
        if (movimiento_detectado)
        {
            // Se analiza el tamaño de los contornos encontrados
            for (size_t i = 0; i<contorno.size(); i++)
            {
                // Lectura del área y del perímetro
                area = contourArea(contorno[i], false);
                perimetro = arcLength(contorno[i], 1);

                // Se recuadra el contorno del movimiento para obtener su centro de coordenadas
                Recuadro_movimiento = boundingRect(contorno[i]);

                //Coordenadas del centro del rectángulo
                int coordenadaX;
                int coordenadaY;
                int a;
                int b;
                int a1;
                int b1;
                int a2;
                int b2;
                int a3;
                int b3;
                int a4;
                int b4;
                int a5;
                int b5;

                coordenadaX = Recuadro_movimiento.x + Recuadro_movimiento.width / 2;
                coordenadaY = Recuadro_movimiento.y + Recuadro_movimiento.height / 2;

                if (coordenadaY <= 50 && coordenadaY > 0)
                {
                    area_baseMIN = 500;
                    area_baseMAX = 9000;
                }
                else if (coordenadaY <= 100 && coordenadaY > 50)
                {
                    area_baseMIN = 1000;
                    area_baseMAX = 9000;
                }
                else if (coordenadaY <= 150 && coordenadaY > 100)
                {
```

```
                    area_baseMIN = 1000;
                    area_baseMAX = 9000;
                }
                else if (coordenadaY <= 200 && coordenadaY > 150)
                {
                    if (coordenadaX > 200)
                    {
                        area_baseMIN = 1000;
                        area_baseMAX = 9000;
                    }
                    else
                    {
                        area_baseMIN = 1000;
                        area_baseMAX = 9000;
                    }
                }
                else if (coordenadaY <= 250 && coordenadaY > 200)
                {
                    if (coordenadaX > 200)
                    {
                        area_baseMIN = 1000;
                        area_baseMAX = 9000;
                    }
                    else
                    {
                        area_baseMIN = 1000;
                        area_baseMAX = 9000;
                    }

                }
                else if (coordenadaY <= 300 && coordenadaY > 250)
                {
                    if (coordenadaX > 200)
                    {
                        area_baseMIN = 1000;
                        area_baseMAX = 9000;
                    }
                    else
                    {
                        area_baseMIN = 1000;
                        area_baseMAX = 9000;
                    }

                }
                else if (coordenadaY <= 350 && coordenadaY > 300)
                {
                    area_baseMIN = 1000;
                    area_baseMAX = 9000;
                }
                else if (coordenadaY <= 400 && coordenadaY > 350)
                {
                    area_baseMIN = 1000;
                    area_baseMAX = 9000;
                }
                else
                {
```

```
                    area_baseMIN = 1000;
                    area_baseMAX = 9000;
                }

                test_data.at<float>(i, 0) = perimetro;
                test_data.at<float>(i, 1) = area;

                // Si el área leída no se encuentra entre nuestros límites, el objeto en movimiento no es nuestro robot.
                if (area > area_baseMAX || area < area_baseMIN)
                {
                    // No hacer nada, no es nuestro robot
                }
                // Si el área leída se encuentra entre nuestros límites, el objeto en movimiento es nuestro robot.
                else
                {
                    cout << "ROBOT DETECTADO" << endl;

                    // Se dibuja su contorno
                    drawContours(captura, contorno, i, Scalar(255), 2);

                    // Se dibuja el centro del rectángulo en el video original
                    circle(captura, Point(coordenadaX, coordenadaY), 2, Scalar(0, 255, 0), 2);
                    line(captura, Point(coordenadaX, coordenadaY), Point(coordenadaX, coordenadaY - 5), Scalar(0, 255, 0), 2);
                    line(captura, Point(coordenadaX, coordenadaY), Point(coordenadaX, coordenadaY + 5), Scalar(0, 255, 0), 2);
                    line(captura, Point(coordenadaX, coordenadaY), Point(coordenadaX - 5, coordenadaY), Scalar(0, 255, 0), 2);
                    line(captura, Point(coordenadaX, coordenadaY), Point(coordenadaX + 5, coordenadaY), Scalar(0, 255, 0), 2);

                    // Se escriben las coordenadas en el video
                    putText(captura, "(" + conversion(coordenadaX) + "," + conversion(coordenadaY) + ")", Point(coordenadaX, coordenadaY), 1, 1, Scalar(255, 0, 0), 2);

                    // Calcular el radio
                    int radio;
                    radio = (perimetro / (2 * pi)) + 15;

                    // Se dibujan un conjunto de puntos alrededor del contorno
                    a = coordenadaX + (radio*sin(45));
                    b = coordenadaY + (radio*cos(45));
                    a1 = coordenadaX + (radio*sin(90));
                    b1 = coordenadaY + (radio*cos(90));
                    a2 = coordenadaX + (radio*sin(135));
                    b2 = coordenadaY + (radio*cos(135));
                    a3 = coordenadaX + (radio*sin(180));
                    b3 = coordenadaY + (radio*cos(180));
                    a4 = coordenadaX + (radio*sin(225));
                    b4 = coordenadaY + (radio*cos(225));
```

```
                    a5 = coordenadaX + (radio*sin(270));
                    b5 = coordenadaY + (radio*cos(270));

                    circle(captura, Point(a, b), 2, Scalar(0, 255, 0), 2);
                    circle(captura, Point(a1, b1), 2, Scalar(0, 255, 0), 2);
                    circle(captura, Point(a2, b2), 2, Scalar(0, 255, 0), 2);
                    circle(captura, Point(a3, b3), 2, Scalar(0, 255, 0), 2);
                    circle(captura, Point(a4, b4), 2, Scalar(0, 255, 0), 2);
                    circle(captura, Point(a5, b5), 2, Scalar(0, 255, 0), 2);


                    // Detectar si el robot se encuentra dentro o fuera de la zona prohibida
                    Point P0;
                    Point P1;
                    Point P2;
                    Point P3;
                    Point P4;
                    Point P5;

                    P0 = Point(coordenadaX0, coordenadaY0);
                    P1 = Point(coordenadaX1, coordenadaY1);
                    P2 = Point(coordenadaX2, coordenadaY2);
                    if (num_puntos == 4)
                    {
                        P3 = Point(coordenadaX3, coordenadaY3);
                    }
                    if (num_puntos == 5)
                    {
                        P4 = Point(coordenadaX4, coordenadaY4);
                    }
                    if (num_puntos == 6)
                    {
                        P5 = Point(coordenadaX5, coordenadaY5);
                    }

                    // Se calculan el número de cortes con el polígono al trazar una línea desde el punto hacia el exterior
                    double xinters;
                    double xinters1;
                    double xinters2;
                    double xinters3;
                    double xinters4;
                    double xinters5;

                    int cortes = 0;
                    int cortes1 = 0;
                    int cortes2 = 0;
                    int cortes3 = 0;
                    int cortes4 = 0;
                    int cortes5 = 0;

                    if (b > MIN(P0.y, P1.y))
                    {
                        if (b <= MAX(P0.y, P1.y))
                        {
```

```
                            if (a <= MAX(P0.x, P1.x))
                            {
                                if (P0.y != P1.y)
                                {
                                    xinters = (b - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                    if (P0.x == P1.x || a <= xinters)
                                        cortes++;
                                }
                            }
                        }
                    }

                    if (b1 > MIN(P0.y, P1.y))
                    {
                        if (b1 <= MAX(P0.y, P1.y))
                        {
                            if (a1 <= MAX(P0.x, P1.x))
                            {
                                if (P0.y != P1.y)
                                {
                                    xinters1 = (b1 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                    if (P0.x == P1.x || a1 <= xinters1)
                                        cortes1++;
                                }
                            }
                        }
                    }

                    if (b2 > MIN(P0.y, P1.y))
                    {
                        if (b2 <= MAX(P0.y, P1.y))
                        {
                            if (a2 <= MAX(P0.x, P1.x))
                            {
                                if (P0.y != P1.y)
                                {
                                    xinters2 = (b2 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                    if (P0.x == P1.x || a2 <= xinters2)
                                        cortes2++;
                                }
                            }
                        }
                    }


                    if (b3 > MIN(P0.y, P1.y))
                    {
                        if (b3 <= MAX(P0.y, P1.y))
                        {
                            if (a3 <= MAX(P0.x, P1.x))
```

```
                            {
                                if (P0.y != P1.y)
                                {
                                    xinters3 = (b3 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                    if (P0.x == P1.x || a3 <= xinters3)
                                        cortes3++;
                                }
                            }
                        }
                    }

                    if (b4 > MIN(P0.y, P1.y))
                    {
                        if (b4 <= MAX(P0.y, P1.y))
                        {
                            if (a4 <= MAX(P0.x, P1.x))
                            {
                                if (P0.y != P1.y)
                                {
                                    xinters4 = (b4 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                    if (P0.x == P1.x || a4 <= xinters4)
                                        cortes4++;
                                }
                            }
                        }
                    }

                    if (b5 > MIN(P0.y, P1.y))
                    {
                        if (b5 <= MAX(P0.y, P1.y))
                        {
                            if (a5 <= MAX(P0.x, P1.x))
                            {
                                if (P0.y != P1.y)
                                {
                                    xinters5 = (b5 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                    if (P0.x == P1.x || a5 <= xinters5)
                                        cortes5++;
                                }
                            }
                        }
                    }

                    if (b> MIN(P1.y, P2.y))
                    {
                        if (b <= MAX(P1.y, P2.y))
                        {
                            if (a <= MAX(P1.x, P2.x))
                            {
                                if (P1.y != P2.y)
```

```
                                {
                                    xinters = (b - P1.y)*(P2.x - P1.x) / (P2.y - P1.y) + P1.x;

                                    if (P1.x == P2.x || a <= xinters)
                                        cortes++;
                                }
                            }
                        }
                    }

                    if (b1> MIN(P1.y, P2.y))
                    {
                        if (b1 <= MAX(P1.y, P2.y))
                        {
                            if (a1 <= MAX(P1.x, P2.x))
                            {
                                if (P1.y != P2.y)
                                {
                                    xinters1 = (b1 - P1.y)*(P2.x - P1.x) / (P2.y - P1.y) + P1.x;

                                    if (P1.x == P2.x || a1 <= xinters1)
                                        cortes1++;
                                }
                            }
                        }
                    }

                    if (b2> MIN(P1.y, P2.y))
                    {
                        if (b2 <= MAX(P1.y, P2.y))
                        {
                            if (a2 <= MAX(P1.x, P2.x))
                            {
                                if (P1.y != P2.y)
                                {
                                    xinters2 = (b2 - P1.y)*(P2.x - P1.x) / (P2.y - P1.y) + P1.x;

                                    if (P1.x == P2.x || a2 <= xinters2)
                                        cortes2++;
                                }
                            }
                        }
                    }

                    if (b3> MIN(P1.y, P2.y))
                    {
                        if (b3 <= MAX(P1.y, P2.y))
                        {
                            if (a3 <= MAX(P1.x, P2.x))
                            {
                                if (P1.y != P2.y)
                                {
                                    xinters3 = (b3 - P1.y)*(P2.x - P1.x) /
```

```
                        (P2.y - P1.y) + P1.x;

                                        if (P1.x == P2.x || a3 <= xinters3)
                                            cortes3++;
                                    }
                                }
                            }
                        }

                        if (b4> MIN(P1.y, P2.y))
                        {
                            if (b4 <= MAX(P1.y, P2.y))
                            {
                                if (a4 <= MAX(P1.x, P2.x))
                                {
                                    if (P1.y != P2.y)
                                    {
                                        xinters4 = (b4 - P1.y)*(P2.x - P1.x) /
                        (P2.y - P1.y) + P1.x;

                                        if (P1.x == P2.x || a4 <= xinters4)
                                            cortes4++;
                                    }
                                }
                            }
                        }

                        if (b5> MIN(P1.y, P2.y))
                        {
                            if (b5 <= MAX(P1.y, P2.y))
                            {
                                if (a5 <= MAX(P1.x, P2.x))
                                {
                                    if (P1.y != P2.y)
                                    {
                                        xinters5 = (b5 - P1.y)*(P2.x - P1.x) /
                        (P2.y - P1.y) + P1.x;

                                        if (P1.x == P2.x || a5 <= xinters5)
                                            cortes5++;
                                    }
                                }
                            }
                        }

                        if (num_puntos == 3)
                        {
                            if (b > MIN(P2.y, P0.y))
                            {
                                if (b <= MAX(P2.y, P0.y))
                                {
                                    if (a <= MAX(P2.x, P0.x))
                                    {
                                        if (P2.y != P0.y)
                                        {
                                            xinters = (b - P2.y)*(P0.x - P2.x) /
```

```
                        (P0.y - P2.y) + P2.x;

                                        if (P2.x == P0.x || a <= xinters)
                                            cortes++;
                                    }
                                }
                            }
                        }

                        if (b1 > MIN(P2.y, P0.y))
                        {
                            if (b1 <= MAX(P2.y, P0.y))
                            {
                                if (a1 <= MAX(P2.x, P0.x))
                                {
                                    if (P2.y != P0.y)
                                    {
                                        xinters1 = (b1 - P2.y)*(P0.x -
                        P2.x) / (P0.y - P2.y) + P2.x;

                                        if (P2.x == P0.x || a1 <= xinters1)
                                            cortes1++;
                                    }
                                }
                            }
                        }

                        if (b2 > MIN(P2.y, P0.y))
                        {
                            if (b2 <= MAX(P2.y, P0.y))
                            {
                                if (a2 <= MAX(P2.x, P0.x))
                                {
                                    if (P2.y != P0.y)
                                    {
                                        xinters2 = (b2 - P2.y)*(P0.x -
                        P2.x) / (P0.y - P2.y) + P2.x;

                                        if (P2.x == P0.x || a2 <= xinters2)
                                            cortes2++;
                                    }
                                }
                            }
                        }

                        if (b3 > MIN(P2.y, P0.y))
                        {
                            if (b3 <= MAX(P2.y, P0.y))
                            {
                                if (a3 <= MAX(P2.x, P0.x))
                                {
                                    if (P2.y != P0.y)
                                    {
                                        xinters3 = (b3 - P2.y)*(P0.x -
                        P2.x) / (P0.y - P2.y) + P2.x;

```

```
                                        if (P2.x == P0.x || a3 <= xinters3)
                                            cortes3++;
                                    }
                                }
                            }
                        }

                        if (b4 > MIN(P2.y, P0.y))
                        {
                            if (b4 <= MAX(P2.y, P0.y))
                            {
                                if (a4 <= MAX(P2.x, P0.x))
                                {
                                    if (P2.y != P0.y)
                                    {
                                        xinters4 = (b4 - P2.y)*(P0.x - P2.x) / (P0.y - P2.y) + P2.x;

                                        if (P2.x == P0.x || a4 <= xinters4)
                                            cortes4++;
                                    }
                                }
                            }
                        }

                        if (b5 > MIN(P2.y, P0.y))
                        {
                            if (b5 <= MAX(P2.y, P0.y))
                            {
                                if (a5 <= MAX(P2.x, P0.x))
                                {
                                    if (P2.y != P0.y)
                                    {
                                        xinters5 = (b5 - P2.y)*(P0.x - P2.x) / (P0.y - P2.y) + P2.x;

                                        if (P2.x == P0.x || a5 <= xinters5)
                                            cortes5++;
                                    }
                                }
                            }
                        }

                    }

                    if (num_puntos == 4 || num_puntos == 5 || num_puntos == 6)
                    {
                        if (b  > MIN(P2.y, P3.y))
                        {
                            if (b <= MAX(P2.y, P3.y))
                            {
                                if (a <= MAX(P2.x, P3.x))
                                {
                                    if (P2.y != P3.y)
                                    {
```

```
                                        xinters = (b - P2.y)*(P3.x - P2.x) / (P3.y - P2.y) + P2.x;

                                        if (P2.x == P3.x || a <= xinters)
                                            cortes++;
                                    }
                                }
                            }
                        }

                        if (b1 > MIN(P2.y, P3.y))
                        {
                            if (b1 <= MAX(P2.y, P3.y))
                            {
                                if (a1 <= MAX(P2.x, P3.x))
                                {
                                    if (P2.y != P3.y)
                                    {
                                        xinters1 = (b1 - P2.y)*(P3.x - P2.x) / (P3.y - P2.y) + P2.x;

                                        if (P2.x == P3.x || a1 <= xinters1)
                                            cortes1++;
                                    }
                                }
                            }
                        }

                        if (b2  > MIN(P2.y, P3.y))
                        {
                            if (b2 <= MAX(P2.y, P3.y))
                            {
                                if (a2 <= MAX(P2.x, P3.x))
                                {
                                    if (P2.y != P3.y)
                                    {
                                        xinters2 = (b2 - P2.y)*(P3.x - P2.x) / (P3.y - P2.y) + P2.x;

                                        if (P2.x == P3.x || a2 <= xinters2)
                                            cortes2++;
                                    }
                                }
                            }
                        }


                        if (b3  > MIN(P2.y, P3.y))
                        {
                            if (b3 <= MAX(P2.y, P3.y))
                            {
                                if (a3 <= MAX(P2.x, P3.x))
                                {
                                    if (P2.y != P3.y)
                                    {
                                        xinters3 = (b3 - P2.y)*(P3.x -
```

```
                        P2.x) / (P3.y - P2.y) + P2.x;

                                            if (P2.x == P3.x || a3 <= xinters3)
                                                cortes3++;
                                        }
                                    }
                                }
                            }


                            if (b4  > MIN(P2.y, P3.y))
                            {
                                if (b4 <= MAX(P2.y, P3.y))
                                {
                                    if (a4 <= MAX(P2.x, P3.x))
                                    {
                                        if (P2.y != P3.y)
                                        {
                                            xinters4 = (b4 - P2.y)*(P3.x -
                        P2.x) / (P3.y - P2.y) + P2.x;

                                            if (P2.x == P3.x || a4 <= xinters4)
                                                cortes4++;
                                        }
                                    }
                                }
                            }


                            if (b5  > MIN(P2.y, P3.y))
                            {
                                if (b5 <= MAX(P2.y, P3.y))
                                {
                                    if (a5 <= MAX(P2.x, P3.x))
                                    {
                                        if (P2.y != P3.y)
                                        {
                                            xinters5 = (b5 - P2.y)*(P3.x -
                        P2.x) / (P3.y - P2.y) + P2.x;

                                            if (P2.x == P3.x || a5 <= xinters5)
                                                cortes5++;
                                        }
                                    }
                                }
                            }

                        }

                        if (num_puntos == 4)
                        {
                            if (b  > MIN(P3.y, P0.y))
                            {
                                if (b <= MAX(P3.y, P0.y))
                                {
                                    if (a <= MAX(P3.x, P0.x))
```

```
                                {
                                    if (P3.y != P0.y)
                                    {
                                        xinters = (b - P3.y)*(P0.x - P3.x) / (P0.y - P3.y) + P3.x;

                                        if (P3.x == P0.x || a <= xinters)
                                            cortes++;

                                    }
                                }
                            }
                        }

                        if (b1 > MIN(P3.y, P0.y))
                        {
                            if (b1 <= MAX(P3.y, P0.y))
                            {
                                if (a1 <= MAX(P3.x, P0.x))
                                {
                                    if (P3.y != P0.y)
                                    {
                                        xinters1 = (b1 - P3.y)*(P0.x - P3.x) / (P0.y - P3.y) + P3.x;

                                        if (P3.x == P0.x || a1 <= xinters1)
                                            cortes1++;

                                    }
                                }
                            }
                        }


                        if (b2  > MIN(P3.y, P0.y))
                        {
                            if (b2 <= MAX(P3.y, P0.y))
                            {
                                if (a2 <= MAX(P3.x, P0.x))
                                {
                                    if (P3.y != P0.y)
                                    {
                                        xinters2 = (b2 - P3.y)*(P0.x - P3.x) / (P0.y - P3.y) + P3.x;

                                        if (P3.x == P0.x || a2 <= xinters2)
                                            cortes2++;

                                    }
                                }
                            }
                        }


                        if (b3  > MIN(P3.y, P0.y))
                        {
```

```
                            if (b3 <= MAX(P3.y, P0.y))
                            {
                                if (a3 <= MAX(P3.x, P0.x))
                                {
                                    if (P3.y != P0.y)
                                    {
                                        xinters3 = (b3 - P3.y)*(P0.x - P3.x) / (P0.y - P3.y) + P3.x;

                                        if (P3.x == P0.x || a3 <= xinters3)
                                            cortes3++;

                                    }
                                }
                            }
                        }


                        if (b4  > MIN(P3.y, P0.y))
                        {
                            if (b4 <= MAX(P3.y, P0.y))
                            {
                                if (a4 <= MAX(P3.x, P0.x))
                                {
                                    if (P3.y != P0.y)
                                    {
                                        xinters4 = (b4 - P3.y)*(P0.x - P3.x) / (P0.y - P3.y) + P3.x;

                                        if (P3.x == P0.x || a4 <= xinters4)
                                            cortes4++;

                                    }
                                }
                            }
                        }


                        if (b5  > MIN(P3.y, P0.y))
                        {
                            if (b5 <= MAX(P3.y, P0.y))
                            {
                                if (a5 <= MAX(P3.x, P0.x))
                                {
                                    if (P3.y != P0.y)
                                    {
                                        xinters5 = (b5 - P3.y)*(P0.x - P3.x) / (P0.y - P3.y) + P3.x;

                                        if (P3.x == P0.x || a5 <= xinters5)
                                            cortes5++;

                                    }
                                }
                            }
                        }
```

```
                    }

                    if (num_puntos == 5 || num_puntos == 6)
                    {

                        if (b > MIN(P3.y, P4.y))
                        {
                            if (b <= MAX(P3.y, P4.y))
                            {
                                if (a <= MAX(P3.x, P4.x))
                                {
                                    if (P3.y != P4.y)
                                    {
                                        xinters = (b - P3.y)*(P4.x - P3.x) / (P4.y - P3.y) + P3.x;

                                        if (P3.x == P4.x || a <= xinters)
                                            cortes++;
                                    }
                                }
                            }
                        }

                        if (b1 > MIN(P3.y, P4.y))
                        {
                            if (b1 <= MAX(P3.y, P4.y))
                            {
                                if (a1 <= MAX(P3.x, P4.x))
                                {
                                    if (P3.y != P4.y)
                                    {
                                        xinters1 = (b1 - P3.y)*(P4.x - P3.x) / (P4.y - P3.y) + P3.x;

                                        if (P3.x == P4.x || a1 <= xinters1)
                                            cortes1++;
                                    }
                                }
                            }
                        }

                        if (b2 > MIN(P3.y, P4.y))
                        {
                            if (b2 <= MAX(P3.y, P4.y))
                            {
                                if (a2 <= MAX(P3.x, P4.x))
                                {
                                    if (P3.y != P4.y)
                                    {
                                        xinters2 = (b2 - P3.y)*(P4.x - P3.x) / (P4.y - P3.y) + P3.x;

                                        if (P3.x == P4.x || a2 <= xinters2)
                                            cortes2++;
                                    }
```

```
                                }
                            }
                        }


                        if (b3 > MIN(P3.y, P4.y))
                        {
                            if (b3 <= MAX(P3.y, P4.y))
                            {
                                if (a3 <= MAX(P3.x, P4.x))
                                {
                                    if (P3.y != P4.y)
                                    {
                                        xinters3 = (b3 - P3.y)*(P4.x - P3.x) / (P4.y - P3.y) + P3.x;

                                        if (P3.x == P4.x || a3 <= xinters3)
                                            cortes3++;
                                    }
                                }
                            }
                        }


                        if (b4 > MIN(P3.y, P4.y))
                        {
                            if (b4 <= MAX(P3.y, P4.y))
                            {
                                if (a4 <= MAX(P3.x, P4.x))
                                {
                                    if (P3.y != P4.y)
                                    {
                                        xinters4 = (b4 - P3.y)*(P4.x - P3.x) / (P4.y - P3.y) + P3.x;

                                        if (P3.x == P4.x || a4 <= xinters4)
                                            cortes4++;
                                    }
                                }
                            }
                        }


                        if (b5 > MIN(P3.y, P4.y))
                        {
                            if (b5 <= MAX(P3.y, P4.y))
                            {
                                if (a5 <= MAX(P3.x, P4.x))
                                {
                                    if (P3.y != P4.y)
                                    {
                                        xinters5 = (b5 - P3.y)*(P4.x - P3.x) / (P4.y - P3.y) + P3.x;

                                        if (P3.x == P4.x || a5 <= xinters5)
                                            cortes5++;
```

```
                                    }
                                }
                            }
                        }
                    }

                    if (num_puntos == 5)
                    {
                        if (b > MIN(P4.y, P0.y))
                        {
                            if (b <= MAX(P4.y, P0.y))
                            {
                                if (a <= MAX(P4.x, P0.x))
                                {
                                    if (P4.y != P0.y)
                                    {
                                        xinters = (b - P4.y)*(P0.x - P4.x) / (P0.y - P4.y) + P4.x;

                                        if (P4.x == P0.x || a <= xinters)
                                            cortes++;
                                    }
                                }
                            }
                        }

                        if (b1 > MIN(P4.y, P0.y))
                        {
                            if (b1 <= MAX(P4.y, P0.y))
                            {
                                if (a1 <= MAX(P4.x, P0.x))
                                {
                                    if (P4.y != P0.y)
                                    {
                                        xinters1 = (b1 - P4.y)*(P0.x - P4.x) / (P0.y - P4.y) + P4.x;

                                        if (P4.x == P0.x || a1 <= xinters1)
                                            cortes1++;
                                    }
                                }
                            }
                        }

                        if (b2 > MIN(P4.y, P0.y))
                        {
                            if (b2 <= MAX(P4.y, P0.y))
                            {
                                if (a2 <= MAX(P4.x, P0.x))
                                {
                                    if (P4.y != P0.y)
                                    {
                                        xinters2 = (b2 - P4.y)*(P0.x - P4.x) / (P0.y - P4.y) + P4.x;

                                        if (P4.x == P0.x || a2 <= xinters2)
```

```
                                        cortes2++;
                                    }
                                }
                            }
                        }


                        if (b3 > MIN(P4.y, P0.y))
                        {
                            if (b3 <= MAX(P4.y, P0.y))
                            {
                                if (a3 <= MAX(P4.x, P0.x))
                                {
                                    if (P4.y != P0.y)
                                    {
                                        xinters3 = (b3 - P4.y)*(P0.x - P4.x) / (P0.y - P4.y) + P4.x;

                                        if (P4.x == P0.x || a3 <= xinters3)
                                            cortes3++;
                                    }
                                }
                            }
                        }


                        if (b4 > MIN(P4.y, P0.y))
                        {
                            if (b4 <= MAX(P4.y, P0.y))
                            {
                                if (a4 <= MAX(P4.x, P0.x))
                                {
                                    if (P4.y != P0.y)
                                    {
                                        xinters4 = (b4 - P4.y)*(P0.x - P4.x) / (P0.y - P4.y) + P4.x;

                                        if (P4.x == P0.x || a4 <= xinters4)
                                            cortes4++;
                                    }
                                }
                            }
                        }


                        if (b5 > MIN(P4.y, P0.y))
                        {
                            if (b5 <= MAX(P4.y, P0.y))
                            {
                                if (a5 <= MAX(P4.x, P0.x))
                                {
                                    if (P4.y != P0.y)
                                    {
                                        xinters5 = (b5 - P4.y)*(P0.x - P4.x) / (P0.y - P4.y) + P4.x;

```

```
                                        if (P4.x == P0.x || a5 <= xinters5)
                                            cortes5++;
                                    }
                                }
                            }
                        }
                    }

                    if (num_puntos == 6)
                    {
                        if (b > MIN(P4.y, P5.y))
                        {
                            if (b <= MAX(P4.y, P5.y))
                            {
                                if (a <= MAX(P4.x, P5.x))
                                {
                                    if (P4.y != P5.y)
                                    {
                                        xinters = (b - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a <= xinters)
                                            cortes++;
                                    }
                                }
                            }
                        }

                        if (b1 > MIN(P4.y, P5.y))
                        {
                            if (b1 <= MAX(P4.y, P5.y))
                            {
                                if (a1 <= MAX(P4.x, P5.x))
                                {
                                    if (P4.y != P5.y)
                                    {
                                        xinters1 = (b1 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a1 <= xinters1)
                                            cortes1++;
                                    }
                                }
                            }
                        }

                        if (b2 > MIN(P4.y, P5.y))
                        {
                            if (b2 <= MAX(P4.y, P5.y))
                            {
                                if (a2 <= MAX(P4.x, P5.x))
                                {
                                    if (P4.y != P5.y)
                                    {
                                        xinters2 = (b2 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;
```

```
                                        if (P0.x == P1.x || a2 <= xinters2)
                                            cortes2++;
                                    }
                                }
                            }
                        }

                        if (b3 > MIN(P4.y, P5.y))
                        {
                            if (b3 <= MAX(P4.y, P5.y))
                            {
                                if (a3 <= MAX(P4.x, P5.x))
                                {
                                    if (P4.y != P5.y)
                                    {
                                        xinters3 = (b3 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a3 <= xinters3)
                                            cortes3++;
                                    }
                                }
                            }
                        }

                        if (b4 > MIN(P4.y, P5.y))
                        {
                            if (b4 <= MAX(P4.y, P5.y))
                            {
                                if (a4 <= MAX(P4.x, P5.x))
                                {
                                    if (P4.y != P5.y)
                                    {
                                        xinters4 = (b4 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a4 <= xinters4)
                                            cortes4++;
                                    }
                                }
                            }
                        }

                        if (b5 > MIN(P4.y, P5.y))
                        {
                            if (b5 <= MAX(P4.y, P5.y))
                            {
                                if (a5 <= MAX(P4.x, P5.x))
                                {
                                    if (P4.y != P5.y)
                                    {
                                        xinters5 = (b5 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a5 <= xinters5)
```

```
                                        cortes5++;
                                    }
                                }
                            }
                        }

                        if (b > MIN(P5.y, P0.y))
                        {
                            if (b <= MAX(P5.y, P0.y))
                            {
                                if (a <= MAX(P5.x, P0.x))
                                {
                                    if (P5.y != P0.y)
                                    {
                                        xinters = (b - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a <= xinters)
                                            cortes++;
                                    }
                                }
                            }
                        }

                        if (b1 > MIN(P5.y, P0.y))
                        {
                            if (b1 <= MAX(P5.y, P0.y))
                            {
                                if (a1 <= MAX(P5.x, P0.x))
                                {
                                    if (P5.y != P0.y)
                                    {
                                        xinters1 = (b1 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a1 <= xinters1)
                                            cortes1++;
                                    }
                                }
                            }
                        }


                        if (b2 > MIN(P5.y, P0.y))
                        {
                            if (b2 <= MAX(P5.y, P0.y))
                            {
                                if (a2 <= MAX(P5.x, P0.x))
                                {
                                    if (P5.y != P0.y)
                                    {
                                        xinters2 = (b2 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a2 <= xinters2)
                                            cortes2++;
```

```
                                    }
                                }
                            }
                        }


                        if (b3 > MIN(P5.y, P0.y))
                        {
                            if (b3 <= MAX(P5.y, P0.y))
                            {
                                if (a3 <= MAX(P5.x, P0.x))
                                {
                                    if (P5.y != P0.y)
                                    {
                                        xinters3 = (b3 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a3 <= xinters3)
                                            cortes3++;
                                    }
                                }
                            }
                        }

                        if (b4 > MIN(P5.y, P0.y))
                        {
                            if (b4 <= MAX(P5.y, P0.y))
                            {
                                if (a4 <= MAX(P5.x, P0.x))
                                {
                                    if (P5.y != P0.y)
                                    {
                                        xinters4 = (b4 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a4 <= xinters4)
                                            cortes4++;
                                    }
                                }
                            }
                        }

                        if (b5 > MIN(P5.y, P0.y))
                        {
                            if (b5 <= MAX(P5.y, P0.y))
                            {
                                if (a5 <= MAX(P5.x, P0.x))
                                {
                                    if (P5.y != P0.y)
                                    {
                                        xinters5 = (b5 - P0.y)*(P1.x - P0.x) / (P1.y - P0.y) + P0.x;

                                        if (P0.x == P1.x || a5 <= xinters5)
                                            cortes5++;
                                    }
```

```
                                }
                            }
                        }
                    }

                    // Si el número de cortes de cualquiera de los puntos es par, el robot está dentro de la zona prohibida
                    if (cortes % 2 != 0 || cortes1 % 2 != 0 || cortes2 % 2 != 0 || cortes3 % 2 != 0 || cortes4 % 2 != 0 || cortes5 % 2 != 0)
                    {
                        cout << "DENTRO" << endl;
                        putText(captura, "DENTRO", Point(200, 40), 1, 3, Scalar(0, 0, 255), 3);

                        //Estamos dentro de la zona prohibida, mandamos señal al arduino para que mande señal de IR intermitente
                        if (contadorX <= 10 || contadorX == 20)
                        {
                            if (contadorX == 20)
                                contadorX = 0;
                            else if (contadorX <= 10)
                                contadorX = contadorX + 1;

                            // Envío al Arduino
                            if (Arduino->IsConnected())
                            {
                                char palabra[] = "1";
                                Arduino->WriteData(palabra, sizeof(palabra) - 1);
                                Sleep(1);
                            }

                        }
                        else
                        {

                            contadorX = contadorX + 1;

                            // Envío al Arduino
                            if (Arduino->IsConnected()) {
                                char palabra[] = "0";
                                Arduino->WriteData(palabra, sizeof(palabra) - 1);
                                Sleep(1);
                            }
                        }

                    }
                    else
                    {   // Si el número de cortes de cualquiera de los puntos es impar, el robot está fuera de la zona prohibida

                        cout << "FUERA" << endl;
                        putText(captura, "FUERA", Point(200, 40), 1, 3, Scalar(0, 0, 255), 3);
```

```cpp

                        //Estamos fuera de la zona prohibida, mandamos señal al arduino para que deje de mandar señal de IR
                        if (Arduino->IsConnected())
                        {
                            char palabra[] = "0";

                            Arduino->WriteData(palabra, sizeof(palabra) - 1);
                            Sleep(1);
                        }
                    }
                }
            }
        }

        // Representación
        imshow("Localización del movimiento", captura);

        // Si se pulsa escape, se sale del programa
        if (waitKey(30) >= 0)
            break;
        keypressed = waitKey(1);
    }

    //Liberamos memoria
    destroyAllWindows();

    //Fin del programa
    return 0;
}
```

```cpp
//*********************************************************************//
//     ANEXO 5. Programa de Tratamiento de Visión Artificial           //
//   Libreria SerialClass, necesario de añadir al programa principal   //
//*********************************************************************//

#include "SerialClass.h"

Serial::Serial(char *portName)
{
    //We're not yet connected
    this->connected = false;

    //Try to connect to the given port throuh CreateFile
    this->hSerial = CreateFileA(portName,
        GENERIC_READ | GENERIC_WRITE,
        0,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,
        NULL);

    //Check if the connection was successfull
    if (this->hSerial == INVALID_HANDLE_VALUE)
    {
        //If not success full display an Error
        if (GetLastError() == ERROR_FILE_NOT_FOUND) {

            //Print Error if neccessary
            printf("ERROR: Handle was not attached. Reason: %s not available. \n", portName);

        }
        else
        {
            printf("ERROR!!!");
        }
    }
    else
    {
        //If connected we try to set the comm parameters
        DCB dcbSerialParams = { 0 };

        //Try to get the current
        if (!GetCommState(this->hSerial, &dcbSerialParams))
        {
            //If impossible, show an error
            printf("failed to get current serial parameters!");
        }
        else
        {
            //Define serial connection parameters for the arduino board
            dcbSerialParams.BaudRate = CBR_9600;
            dcbSerialParams.ByteSize = 8;
            dcbSerialParams.StopBits = ONESTOPBIT;
            dcbSerialParams.Parity = NOPARITY;

```

```
            //Set the parameters and check for their proper application
            if (!SetCommState(hSerial, &dcbSerialParams))
            {
                printf("ALERT: Could not set Serial Port parameters");
            }
            else
            {
                //If everything went fine we're connected
                this->connected = true;
                //We wait 2s as the arduino board will be reseting
                Sleep(ARDUINO_WAIT_TIME);
            }
        }
    }

}

Serial::~Serial()
{
    //Check if we are connected before trying to disconnect
    if (this->connected)
    {
        //We're no longer connected
        this->connected = false;
        //Close the serial handler
        CloseHandle(this->hSerial);
    }
}

int Serial::ReadData(char *buffer, unsigned int nbChar)
{
    //Number of bytes we'll have read
    DWORD bytesRead;
    //Number of bytes we'll really ask to read
    unsigned int toRead;

    //Use the ClearCommError function to get status info on the Serial port
    ClearCommError(this->hSerial, &this->errors, &this->status);

    //Check if there is something to read
    if (this->status.cbInQue>0)
    {
        //If there is we check if there is enough data to read the required number
        //of characters, if not we'll read only the available characters to prevent
        //locking of the application.
        if (this->status.cbInQue>nbChar)
        {
            toRead = nbChar;
        }
        else
        {
            toRead = this->status.cbInQue;
        }

```

```
        //Try to read the require number of chars, and return the number of read bytes on success
        if (ReadFile(this->hSerial, buffer, toRead, &bytesRead, NULL) && bytesRead != 0)
        {
            return bytesRead;
        }

    }

    //If nothing has been read, or that an error was detected return -1
    return -1;

}


bool Serial::WriteData(char *buffer, unsigned int nbChar)
{
    DWORD bytesSend;

    //Try to write the buffer on the Serial port
    if (!WriteFile(this->hSerial, (void *)buffer, nbChar, &bytesSend, 0))
    {
        //In case it don't work get comm error and return false
        ClearCommError(this->hSerial, &this->errors, &this->status);

        return false;
    }
    else
        return true;
}

bool Serial::IsConnected()
{
    //Simply return the connection status
    return this->connected;
}
```