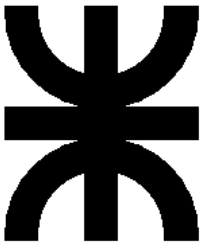


UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL SAN NICOLAS

INGENIERIA EN ELECTRONICA

PROBLEMA DE INGENIERÍA

TECNICAS DIGITALES III

Cálculo de distancia recorrida

Integrantes**:**

- Greganti Lucas
- Iñiguez Jose Luis
- Acerbo Ezequiel

Docentes**:**

- Profesor: Poblete Felipe
- Auxiliar: González Mariano

AÑO 2009

INDICE

OBJETIVOS DEL TRABAJO

El objetivo del trabajo es determinar la distancia recorrida en un salto realizado por un gimnasta, aplicando tratamiento digital de imágenes obtenidas con una cámara. La persona que realice el salto deberá colocarse un sticker (etiqueta adhesiva) el cual será tomado como referencia.

MATERIAS INTEGRADAS

Las siguientes son algunas de las materias que integran el desarrollo del proyecto. También, se indica el contenido que estará relacionado con las mismas.
• Técnicas Digitales III: Procesamiento Digital de Señales y Programación.
• Informática II: Punteros, estructuras, etc.

POSIBLES APLICACIONES

Las aplicaciones comerciales y/o didácticas que se le encuentra a este proyecto son:
• Medición de distancia recorrida de un objeto.
• Para estudios de biomecánica (ya sea centro de altos rendimientos, instituciones deportivas, etc)

PROFESORES ENTREVISTADOS

• Poblete, Felipe
• González, Mariano
• Botta, Ramiro
• Cullen, Luciano (Ingeniero Electrónico)
• Debates en clase de las posibles soluciones

BIBLIOGRAFÍA

• *Apuntes de cátedra.*
• Detección y seguimiento de objetos- Bernardo Calla, Gabriel Malespina, Enzo Varela y Cristian Palomeque
• *Sitios de Internet.*
• http://www.cimat.mx/~jbhayet/CLASES/VISIONROB/opencv2.pdf
• http://www.latindevelopers.com/forum
• http://eupt2.unizar.es/cmedrano/tutorial_opencv.pdf

Sitio para la descarga gratuita de OpenCV 5
http://descargas.itespresso.es/descargar-opencv/windows/utilidades/gestion-de-ficheros/756/

• Sitio para la descarga gratuita de DEV-Cpp 5.0
http://www.cuantosprogramas.com/descargar/1087/windws/Dev-C-5.0-Beta-9.2-4.9.9.2.html

DESARROLLO

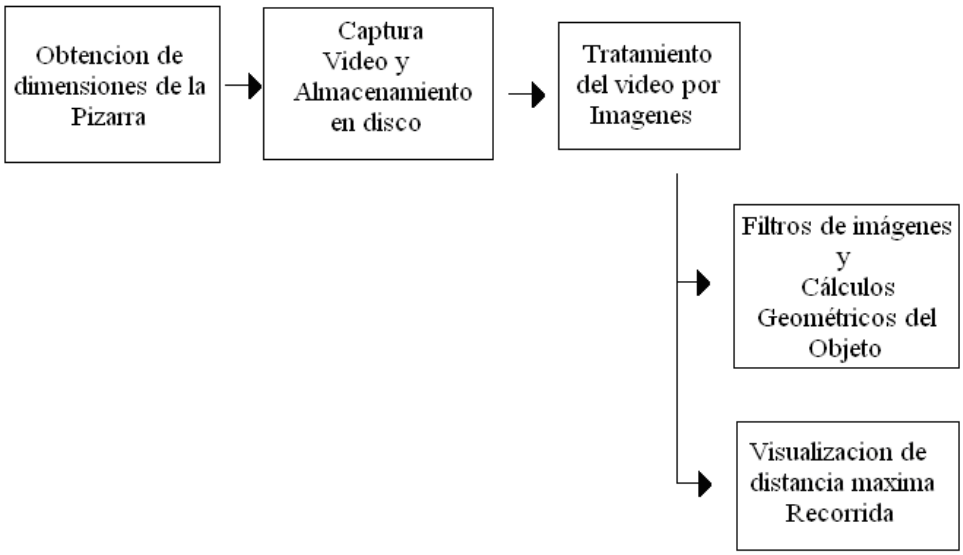
INTRODUCCIÓN

En la actualidad en el ámbito deportivo se requiere cada vez más de nuevas tecnologías que sirven como herramientas para mejorar las técnicas en el "alto rendimiento". La capacidad de salto en un sentido amplio es un factor de rendimiento importante en muchos deportes y disciplinas y en algunos es incluso determinante.
En nuestro caso trabajaremos sobre un archivo de video, en el cual mediante tratamiento digital de imágenes lograremos medir la distancia recorrida de diferentes partes del cuerpo en un salto.
Manipulando esta información los especialistas en la fase deportiva pueden obtener parámetros adicionales que dependen de dicha información y son de primordial importancia, tales como:
Fuerza explosiva, potencia anaeróbica aláctica, resistencia anaeróbica láctica.

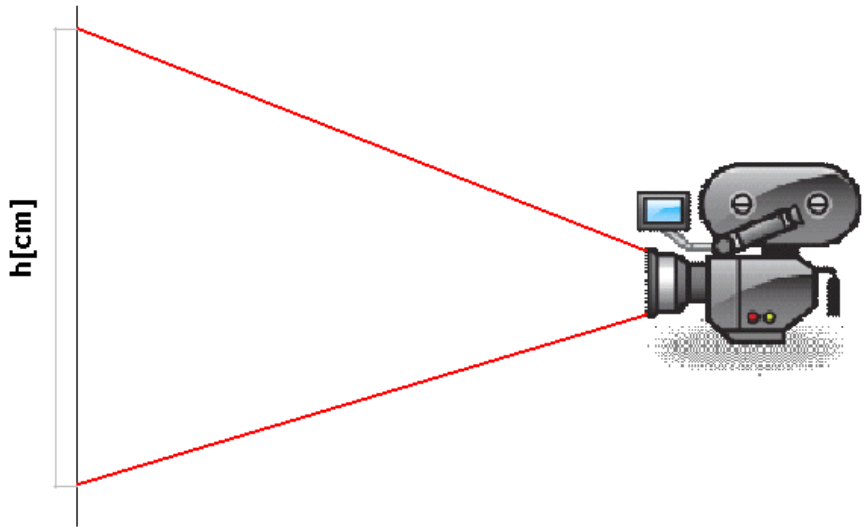
TEORÍA DE FUNCIONAMIENTO

El principio de funcionamiento explicado a continuación citará las funciones utilizadas:
El archivo de video con el cual trabajamos debe ser de extensión avi y las dimensiones de los frame de 480x640 píxeles debido a las especificaciones de las funciones a utilizar.
Los videos se grabaron con la cámara de un celular (25 FPS) y con una cámara digital.
(30 FPS). Estos como no eran de las características especificadas anteriormente, fueron convertidos mediante el software Riva.FLV.Encoder.v.2.0. Este archivo **"video.avi"** fue almacenado en una carpeta de OpenCV, quedando en la siguiente ubicación D:\OpenCV\samples\c.

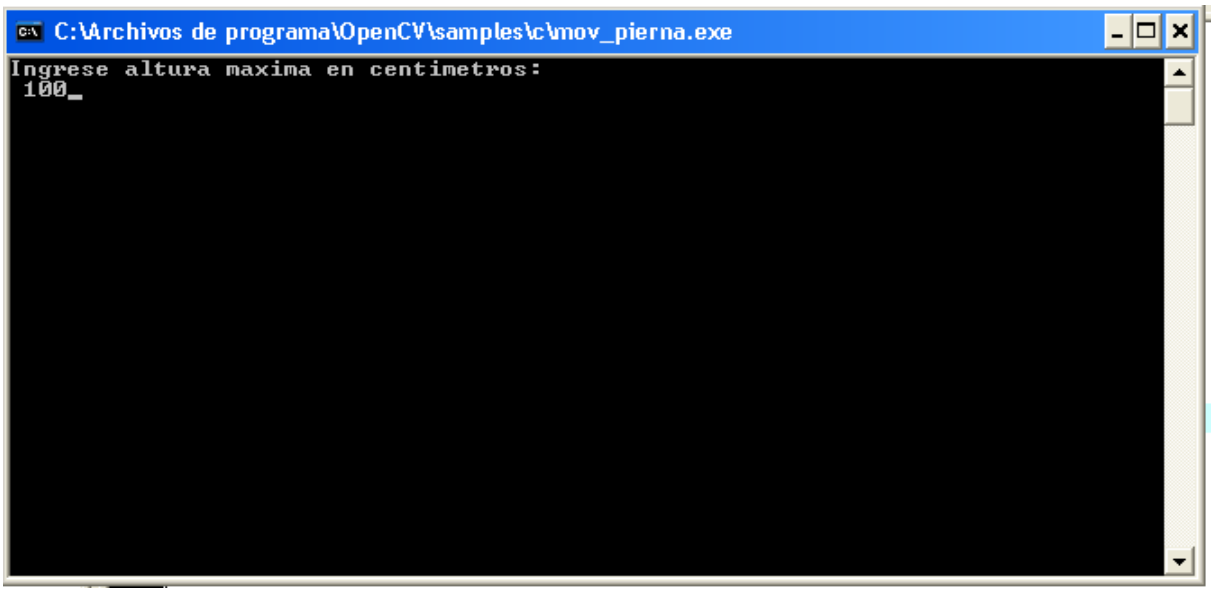
Para la captura del video, se utilizo la función **cvCaptureFromAVI("video.avi")**, la cual busca el video en la ruta descripta, en el caso de no encontrarlo se imprime en pantalla "No hay Imagen a tomar ".
En el siguiente diagrama se observa el camino realizado para llegar al resultado final de la distancia máxima distancia recorrida.



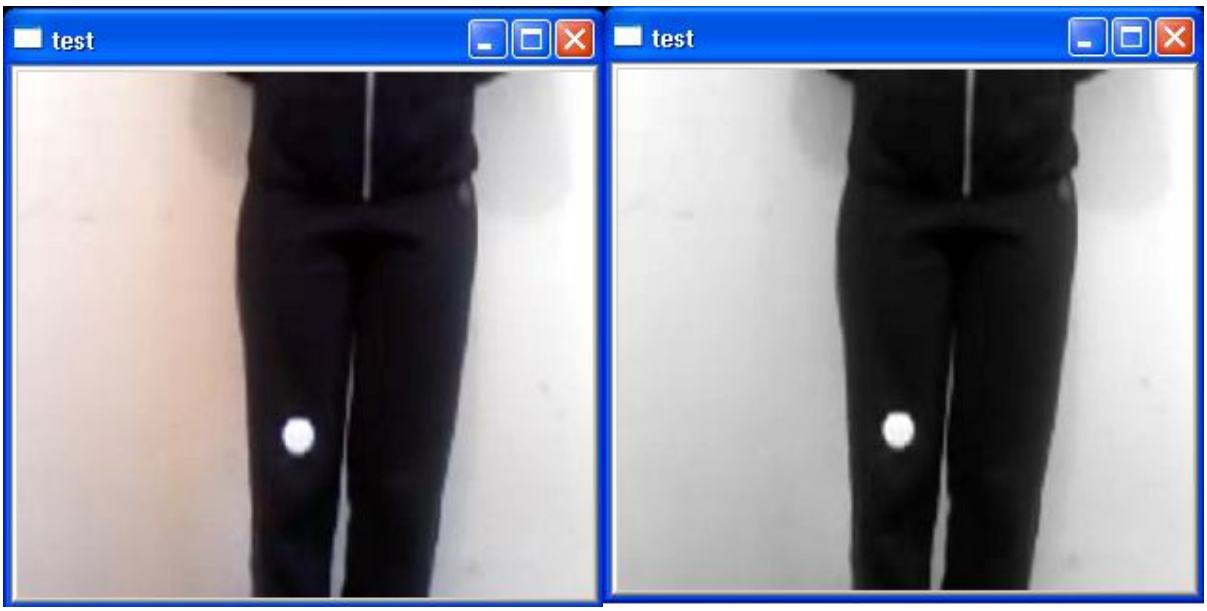
Para lograr la relación de distancia (real-pixel) el programa fue diseñado para que se ingrese como dato desde teclado la altura máxima (la cual es medida de antemano, que es la distancia de la pizarra de fondo (Pizarra= área donde se va a mover el objeto), ver grafico, que captura la cámara, la cual se observa en la siguiente imagen.



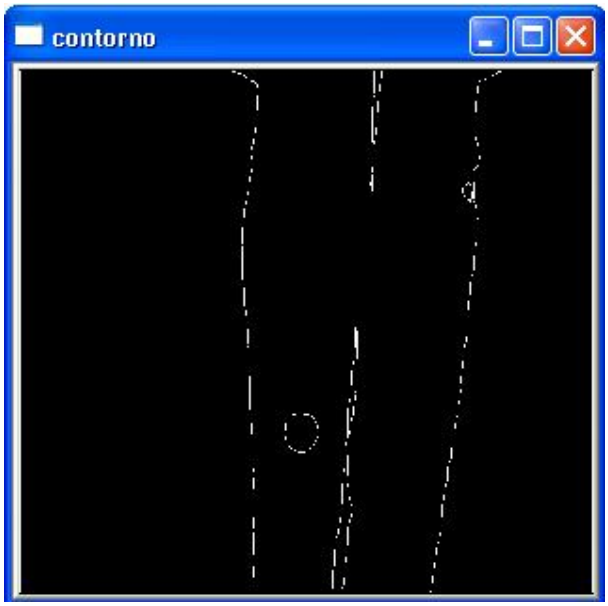
Si la altura capturada es por ejemplo un metro, debemos ingresar por teclado 100 que es el equivalente en centímetros. De esta forma surge la siguiente relación:
480 píxeles = 100 cm.
En la figura se observa el ejemplo anterior.



La captura de cada imagen se realizó con la función **cvGrabFrame ()**, esta me permite obtener la sucesión de los distintos frame, los cuales son guardados en buffer. La misma nos posibilitó tratar el video como una secuencia de imágenes.
A cada una de estas se las paso a escala de grises mediante la función **cvLoadImage ()**, porque así lo requieren las funciones citadas a continuación.



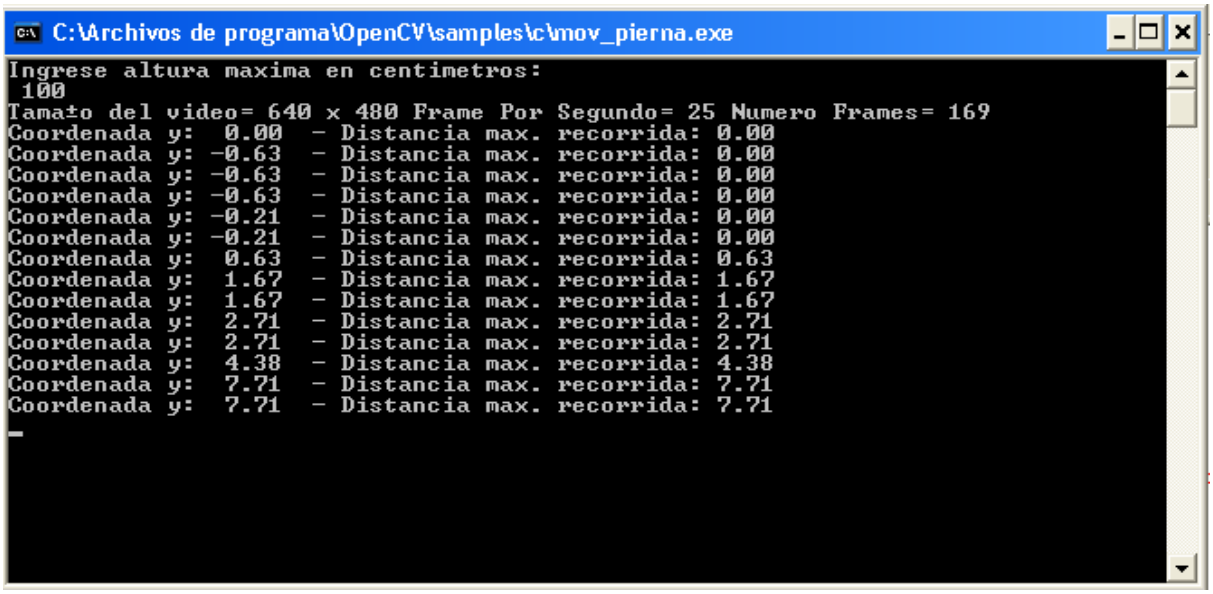
Una vez pasada a escala de grises, se calculo todos los posibles contornos que tuviera la imagen, obteniendo los mismo con la función **cvFindContours ().**
No solo se obtuvo el contorno de sticker, sino también contornos indeseados.



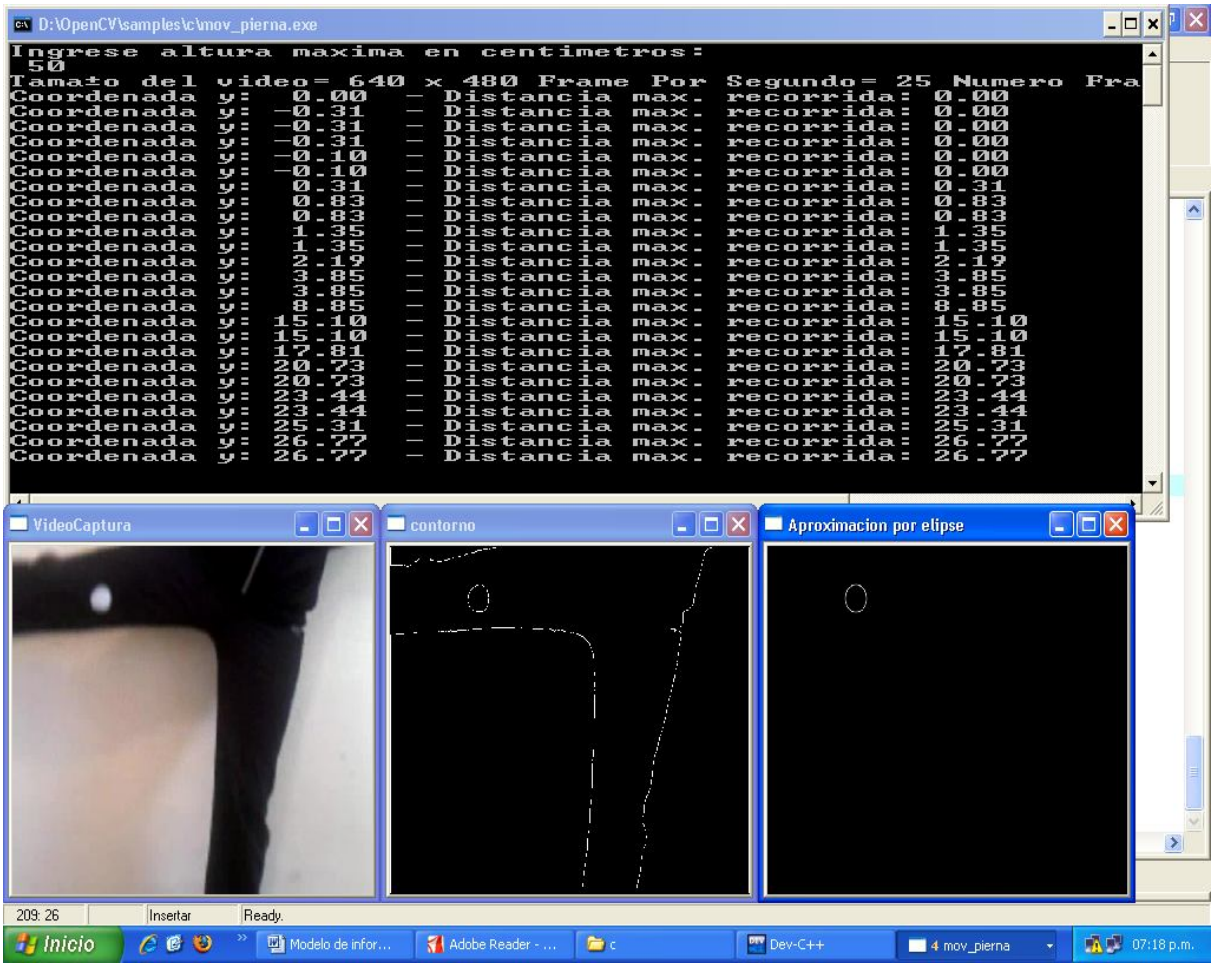
En la primer etapa de filtrado se comenzó por aproximar mediante elipses todo contorno cerrado con la función **cvFitEllipse()**.

Siguiendo con la discriminación se optó por suprimir las elipses que fueran demasiadas grandes o pequeñas comparadas con la correspondiente al sticker, ya que de antemano conocemos su dimensión.

Luego seguimos por calcular el centro del objeto con la función **cvEllipse ()** la cual retorna el centro de las elipses, descartando el movimiento en el eje x.
Aplicando una regla de tres simple se relaciono las coordenadas en píxeles con la distancia en centímetros recorridos.
El primer centro obtenido fue almacenado como la posición inicial, la cual corresponde al valor cero y se tomara como referencia para la medición de distancia máxima.
Después se fueron comparando los sucesivos valores y almacenando el máximo en una variable.
Además se puede ver en la imagen de pantalla el tamaño del frame en pixeles y la numero de frame por segundo.
El signo negativo describe que el objeto se ha situado por debajo del nivel inicial, debido al movimiento de impulso previo al salto.

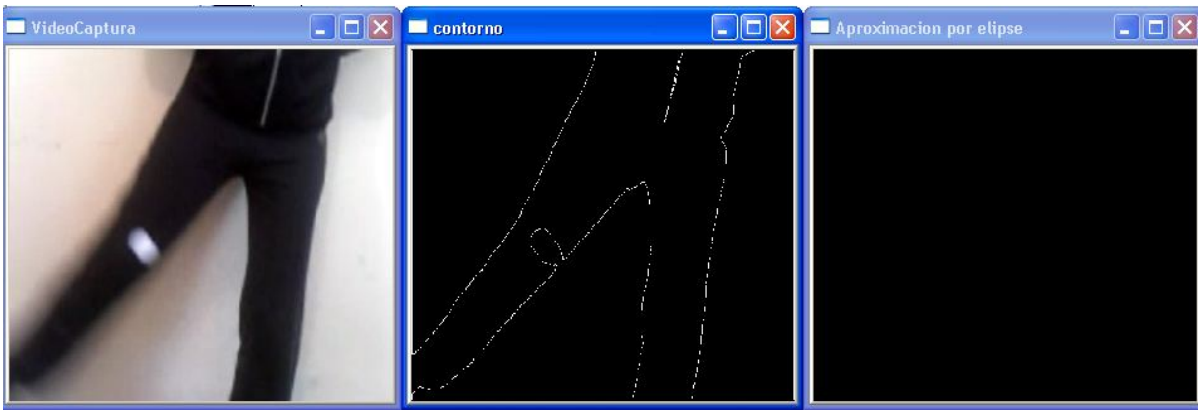


Luego se utilizo las funciones para situar las distintas ventanas, las cuales muestran: el video sin reformas, video con todos los contorno encontrados, video solo contorno de la elipse, y una pantalla mostrando los valores calculados, estos fueron seleccionados pensado en optimizar la visualización e interpretación de dicho proceso.

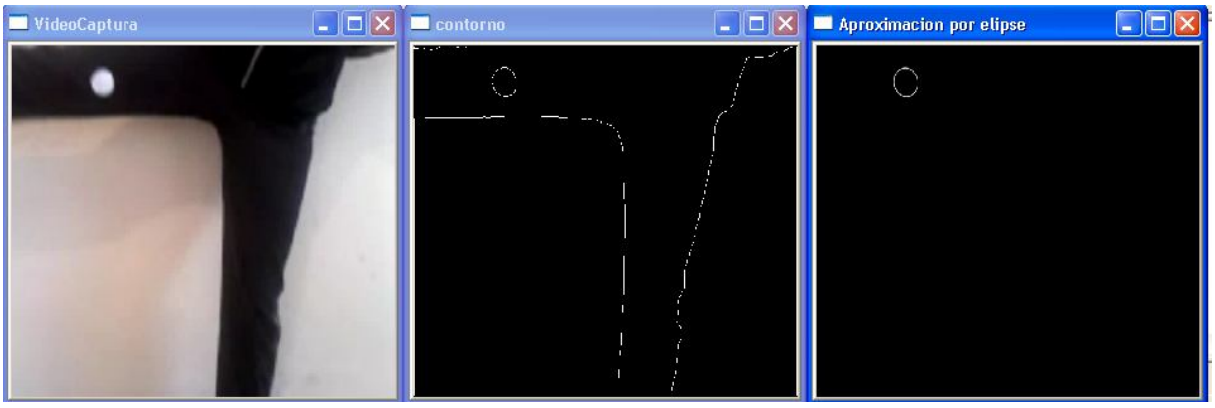


RESULTADOS DE LAS PRUEBAS

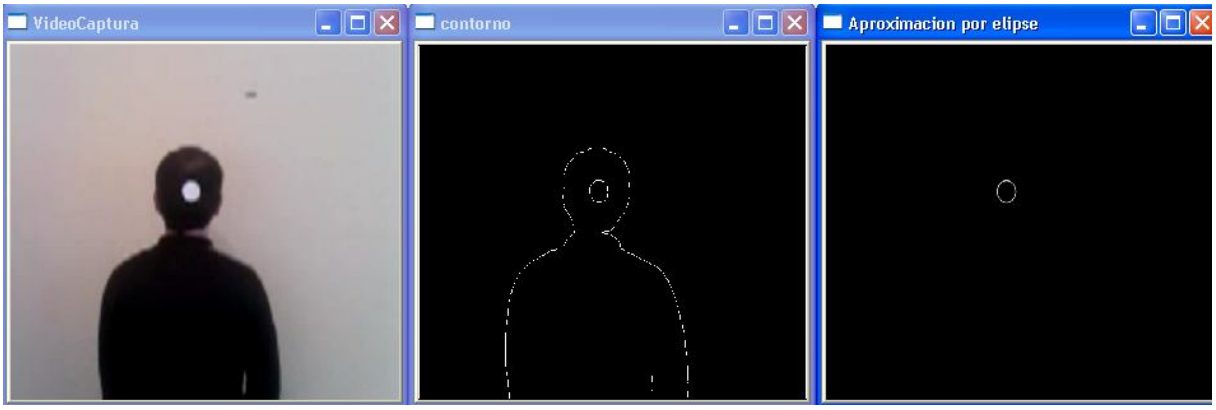
Uno de los inconvenientes que nos encontramos fue la limitación de los frame por Segundo de las cámaras que utilizamos. Por este motivo no pudimos tomar el video directamente desde una WebCam, y decidimos filmar y luego almacenar en disco.
Las cámaras al momento de filmar movimientos rápidos, nos restringió obtener el contornos en todos los frames, ya que el sticker debido a la velocidad, describía una estela, la cual al no respetar las dimensiones elípticas fueron rechazadas.



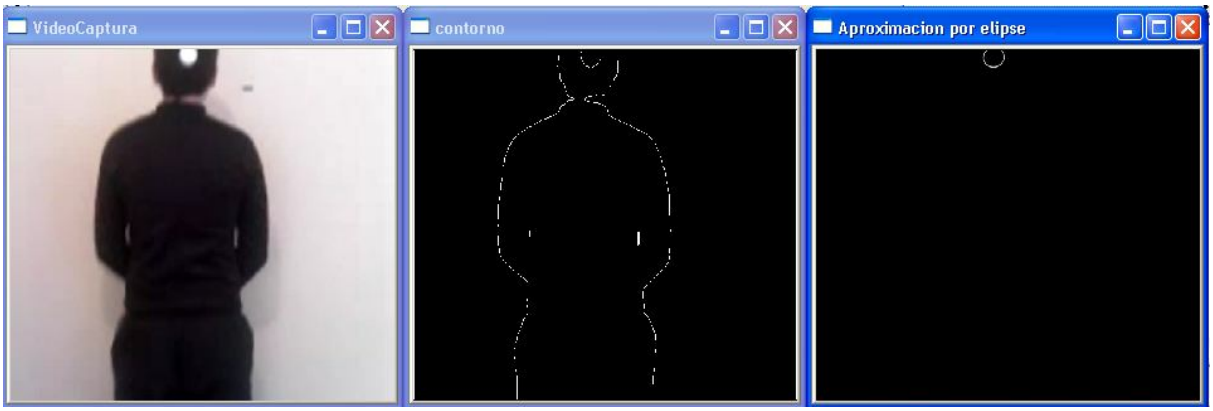
En realidad no fue un problema, ya que la medición más importante a realizar seria la de la posición máxima de la trayectoria recorrida.
En el ejemplo del salto, seria cuando el objeto comienza a descender, luego de haber alcanzado la altura máxima y velocidad nula, por eso dicha imagen nunca se perdió.

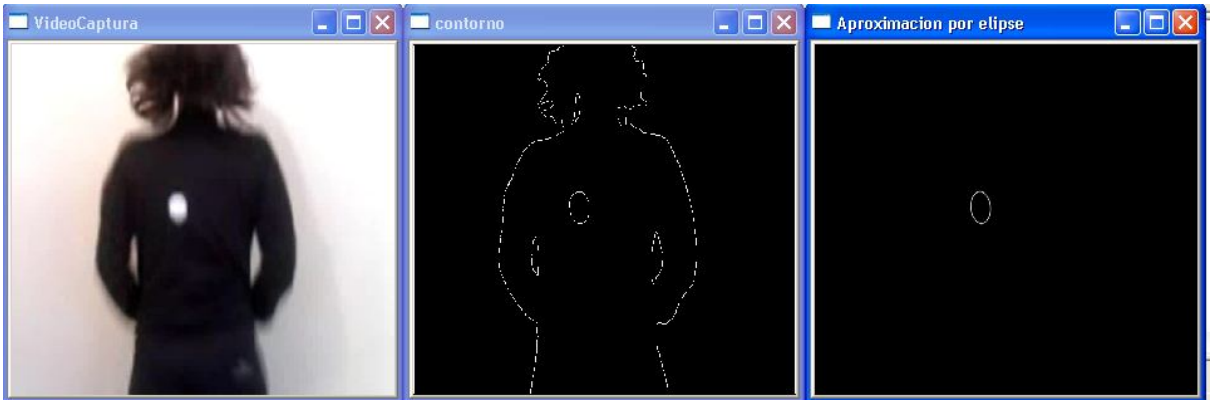


A continuación citamos como ejemplo, el cálculo de nuevas distancias recorridas:



En estas imágenes se puede observar como se comenzó filtrando la imagen obteniendo el contorno y luego discrimar para obtener las elipses de la dimensiones del sticker.





Con respecto a las funciones utilizadas, no hubo mayores inconvenientes, más que entender bien la sintaxis y su correcto funcionamiento.

CONCLUSIONES

El desarrollo de este proyecto nos ha resultado altamente gratificante, ya que nos proporciono una visión distinta acerca de los compiladores, el de trabajar con video e imágenes.
En base al objetivo general de este proyecto, se pudo notar que éste fue cumplido a cabalidad en el grupo, toda vez que hubo un real aprendizaje por parte del grupo este fue ampliamente emotivo, el cual nos impulsaba a seguir con el mismo.
Se pudo experimentar lo que es el proceso de diseño, y cómo la solución a un determinado problema no es única ni puede ser concebida en su totalidad, sino como una mezcla de soluciones a problemas específicos que son consecuencia, o parte, del problema mayor.
Como consecuencia de lo anterior, y no sin dificultades, el grupo logró diseñar una etapa de un proceso, como fue visto a lo largo de todo este informe, en este diseño se aplicó el proceso de captura de video, en el cual se abría el abanico de posibles soluciones a medida que se iba avanzando por éste. Esperando que sirva como base para otras personas y finalmente, para facilitar el camino a los futuros ingenieros.

ANEXOS:

Anexo 1: CD

Se presenta junto al informe escrito un CD con los programas utilizados para llegar al objetivo planteado.

Anexo 2: LISTADOS DE PROGRAMAS

```
#include "stdio.h"
#include "conio.h"
#ifndef _EiC
#include "cv.h"
#include "highgui.h"
#endif

int slider_pos = 71;
IplImage *data_img=NULL;
IplImage *image02=NULL, *image04=NULL, *image05=NULL, *img_result=NULL;
    int a = 0;
      float pos_rel = 0;
    int x;
      float inicial = 0.0;

void process_image(int h)
{
    CvMemStorage* stor;
    CvSeq* cont;
    CvBox2D32f* box;
    CvPoint *PointArray=NULL;
    CvPoint2D32f *PointArray2D32f=NULL;
      CvPoint center;
      CvSize size;
      static float max_h=0,t=480;
      static float max_real= 0;
      float   alt=0;
    float  alt_real=0;
    stor = cvCreateMemStorage(0);
      cont = cvCreateSeq(CV_SEQ_ELTYPE_POINT, sizeof(CvSeq), sizeof(CvPoint) ,
stor);

    // Umbral de la imagen para la función cvfindcontours
    cvThreshold( data_img, image02, slider_pos, 255, CV_THRESH_BINARY );

    // Encuentra el contorno.
    cvFindContours( image02, stor, &cont, sizeof(CvContour),CV_RETR_LIST,
CV_CHAIN_APPROX_NONE, cvPoint(0,0));

    // Imagenes claras. Usa ipl
    cvZero(image02);
    cvZero(image04);
    cvZero(image05);
    // Dibuja el contorno mediante elipses
    for(;cont;cont = cont->h_next)
    {   //int size.widthsize.height
        int i; // Indicador de ciclo.
        int count = cont->total; // punto de contorno

        if( count < 6 )
```

```
            continue;

        // Menoria para el punto de contorno
        PointArray = (CvPoint*)malloc( count*sizeof(CvPoint) );
            if(PointArray == NULL)
            {
                  printf("process_image: no se puede asignar PointArray\n");
                  exit(0);
            }
        PointArray2D32f = (CvPoint2D32f*)malloc( count*sizeof(CvPoint2D32f) );
            if(PointArray2D32f == NULL)
            {
                  printf("process_image: no se puede asignar
PointArray2D32f\n");
                  exit(0);
            }

        // Memoria para los datos de la elipse.
        box = (CvBox2D32f*)malloc(sizeof(CvBox2D32f));

        cvCvtSeqToArray(cont, PointArray, CV_WHOLE_SEQ);

        // Convierte CvPoint a CvBox2D32f.
        for(i=0; i<count; i++)
        {
            PointArray2D32f[i].x = (float)PointArray[i].x;
            PointArray2D32f[i].y = (float)PointArray[i].y;
        }

        // Aplica elipse al contorno.
        cvFitEllipse(PointArray2D32f, count, box);

        // Dibuja el contorno.

cvDrawContours(image04,cont,CV_RGB(255,255,255),CV_RGB(255,255,255),0,1,8,cvPoin
t(0,0));

        // Información de coordenadas
        center.x = cvRound(box->center.x);
        center.y = cvRound(box->center.y);
            alt = t - center.y;


        size.width = cvRound(box->size.width*0.5);
        size.height = cvRound(box->size.height*0.5);
        box->angle = -box->angle;

        // Dibuja la elipse menor a 100
        if((size.width <100 && size.height<100)&& (size.width >1 &&
size.height>1) )
        {cvEllipse(image05, center, size,
                  box->angle, 0, 360,
                  CV_RGB(0,0,255), 1, CV_AA, 0);

             if(a==0)
              {
              a++;
              inicial=alt;
              pos_rel=alt-inicial;

              }
```

```
             else
              pos_rel=alt-inicial;

            //calculo el maximo
            if(pos_rel > max_h)
                  max_h = pos_rel;
                  t=480;
                  alt_real= (pos_rel*x)/t;
                  max_real= (max_h*x)/t;
         // printf("ancho y: %d - alto: %d\n", size.width,size.height );
            printf("Coordenada y: %5.2f  - Distancia max. recorrida: %2.2f\n",
alt_real, max_real);
            //system ("cls");

              a=1;
        }

        // Libera memoria.
        free(PointArray);
        free(PointArray2D32f);
        free(box);
    }

    // muestra elipse.
    cvShowImage( "Aproximacion por elipse", image05 );
    cvMoveWindow("Aproximacion por elipse", 640, 400);
    cvShowImage( "contorno", image04 );
    cvMoveWindow("contorno", 320, 400);
}

void suma_imagen(IplImage *imagen_a, IplImage *imagen_b, IplImage *imagen_c)
{
      int i,j;

      for(i=0;i<imagen_a->height;i++)
      {
            for(j=0;j<imagen_a->width;j++)
            {
            //    imagen_c->imageData[i][j] = imagen_a->imageData[i][j] +
imagen_b->imageData[i][j];
        }
      }

}

int CalculaCentro()
{
    image02 = cvCloneImage( data_img );
    image04 = cvCloneImage( data_img );
      image05 = cvCloneImage( data_img );
    process_image(0);

    cvReleaseImage(&image02);


    return 0;
}


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

```
      CvCapture *capture;
      IplImage *imagen_orig = NULL;
      int i=0,c;
    int frameH,frameW,fps,numFrames;

      capture = cvCaptureFromAVI("mov_pierna.avi");

      printf("Ingrese altura maxima en centimetros:   \n ");
      scanf("%d",&x);



      if(!cvGrabFrame(capture))
    {
      printf("No hay Imagen a tomar\n");
      printf("%p\n",capture);
      getchar();
      exit(-1);
      }

    //Propiedades de la imagen
    frameH = (int) cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT);
    frameW = (int) cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH);
    fps = (int) cvGetCaptureProperty(capture, CV_CAP_PROP_FPS);
    numFrames = (int) cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_COUNT);
    printf("Tamaño del video= %i x %i Frame Por Segundo= %i Numero Frames=
%i\n",frameW,frameH, fps, numFrames);

      cvNamedWindow( "VideoCaptura",0);
    cvNamedWindow("contorno", 0);
      cvNamedWindow("Aproximacion por elipse", 0);
      for(i=0;i<numFrames-4;i++)
      {
            //capturamos la imagen
            cvGrabFrame(capture);
            imagen_orig=cvRetrieveFrame(capture);
            cvShowImage("VideoCaptura",imagen_orig);
        cvMoveWindow("VideoCaptura", 0, 400);
            //guardo a archivo y lo leo en esc de grises
            cvSaveImage("imagen.jpg", imagen_orig);
            const char* imge = argc == 2 ? argv[1] : (char*)"imagen.jpg";
            data_img = cvLoadImage(imge,0);

            CalculaCentro();

            c = cvWaitKey(1);
      }

    cvQueryFrame(capture);
    //Libera la fuente de la captura
    cvReleaseCapture(&capture);

      getch();
    getchar();

    cvDestroyWindow("VideoCaptura");
    cvDestroyWindow("contorno");
      cvDestroyWindow("Aproximacion por elipse");
return 0;
}
```