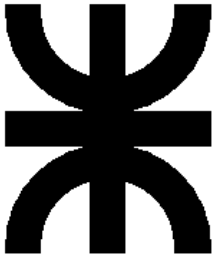

UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL SAN NICOLAS

INGENIERIA EN ELECTRONICA

PROBLEMA DE INGENIERÍA

TECNICAS DIGITALES III

MEDIDOR DE VELOCIDAD DE COLADA CONTINUA

Integrantes:

- Gorrochategui, Ezequiel
- Taruselli, Luciano

Docentes:

- Profesor: Poblete, Felipe
- Auxiliar: Gonzalez, Mariano

AÑO 2011

INDICE

MATERIAS INTEGRADAS

Las materias que integran el desarrollo del proyecto son:
• Técnicas Digitales III: Procesamiento Digital de Señales y Programación.
• Informática II: Punteros, estructuras, etc.

PROFESORES ENTREVISTADOS

• Poblete, Felipe
• González, Mariano

BIBLIOGRAFÍA

• Apuntes de cátedra.

• Detección y seguimiento de objetos- Bernardo Calla, Gabriel Malespina, Enzo Varela y
Cristian Palomeque

• Calculo de distancia recorrida Greganti Lucas, Iñiguez Jose Luis, Acerbo Ezequiel

• Sitios de Internet.

*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

INTRODUCCION

En las empresas de producción de acero se hace recurrente la necesidad de medir la velocidad de la colada continua para mejorar la calidad del producto acabado.

A continuación se hace una breve descripción acerca del proceso de colada continua.

COLADA CONTINUA

fig 1

En esta etapa del proceso, el acero liquido se vierte sobre un molde cuyo fondo se desplaza y su sección tiene la forma que queremos que tenga el producto final ya sean cuadrados, redondos, triangulares, planchas, etc. En la colada continua el producto sale sin parar hasta que se acaba el contenido de la cuchara, y luego se cambia la cuchara para que el proceso continúe indeterminadamente, por lo tanto se ahorra mucho dinero, se consume menos energía y el producto es de gran uniformidad.

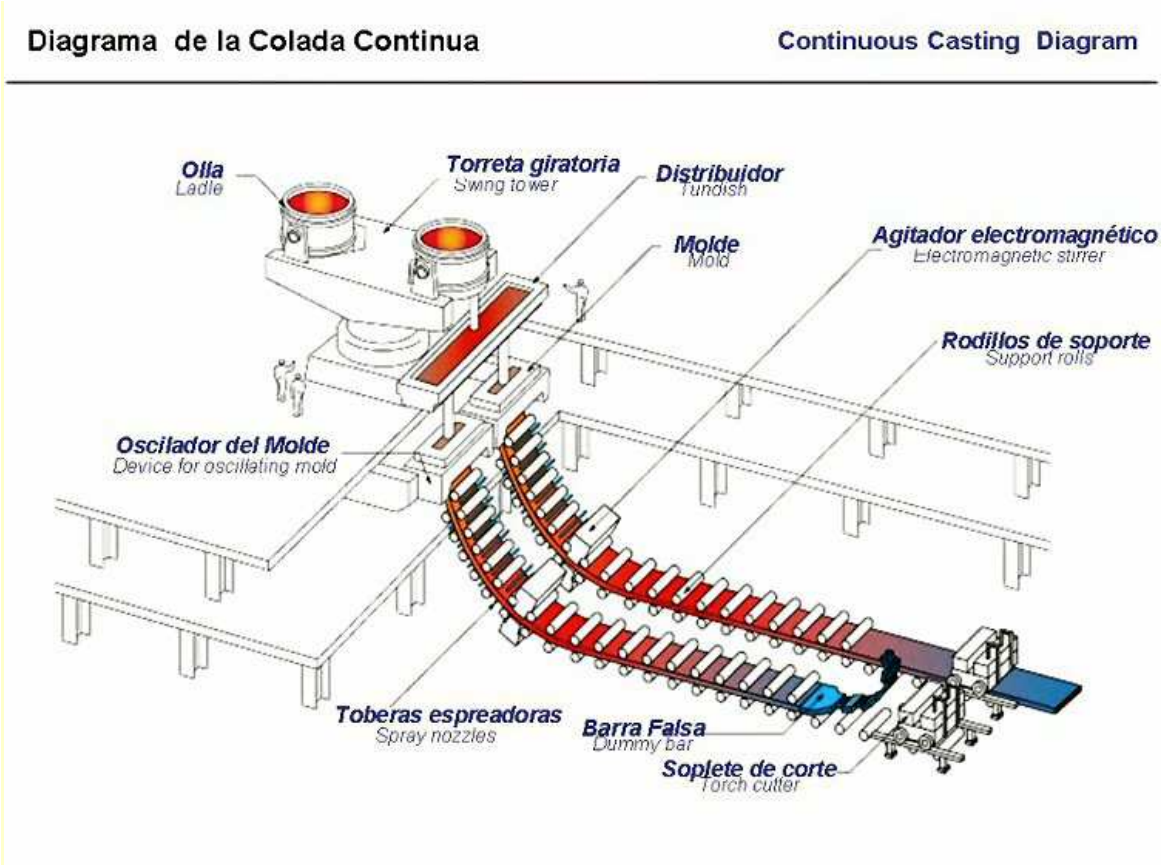

fig 2

El acero líquido de la cuchara se vacía en un recipiente de donde, a velocidad controlada, pasa a un molde de sección cuadrada. La velocidad es una de las variables mas importantes en el proceso, ya que no todos los tipos de aceros se pueden colar a la misma velocidad. Las paredes del molde se lubrican para que el acero no se adhiera y se mantienen "frías" refrigerándolas con serpentines de agua. El molde además, se hace vibrar para ayudar a que el acero se deslice. El molde no tiene tapa inferior porque el acero que ha solidificado en el extremo inferior, sirve como tapa. Luego el acero, ya sólido pero al rojo vivo, pasa por una serie de rodillos que lo traccionan hasta llegar al pantógrafo donde, con sopletes, la sección cuadrada se corta en tramos de la longitud deseada.

Vamos a hacer hincapié en la medición de la velocidad de la colada, que es muy importante para sistemas de control de la estructura del molde de la Colada, y que va a ser tema de desarrollo para nuestro proyecto.

Se debe tener una medición bastante certera de la velocidad instantánea de la colada, ya que de lo contrario se pueden generar fallas de producción.

Como contramedida existe un sistema de Alarmas que se encarga de predecir lo que le llaman sticker en los moldes. Estos se pueden producir porque el polvo colador no es lo suficiente o no llega a tener las propiedades requeridas para la lubricación del material o también porque la velocidad para ese grado de acero no es la correcta, he aquí la importancia de conocer la velocidad.

El sistema actual de medición cuenta con una serie de encoder en la mitad de la colada en un rodillo. Éste requiere mantenimiento y una alta precisión con respecto al diámetro real del rodillo, debido que con esta variable se calcula la velocidad. La fricción con el acero va desgastando el rodillo y por esto es necesario un mantenimiento constante, ya que un cambio en el radio del rodillo implicaría un error de medición en la velocidad de la colada.

Además de estos aspectos vistos anteriormente, la maquina de CCD tiene varios modelos que están continuamente controlando que no se produzca ninguna rotura y deben tener un valor bastante preciso de la velocidad para realizar cálculos en función a ella.

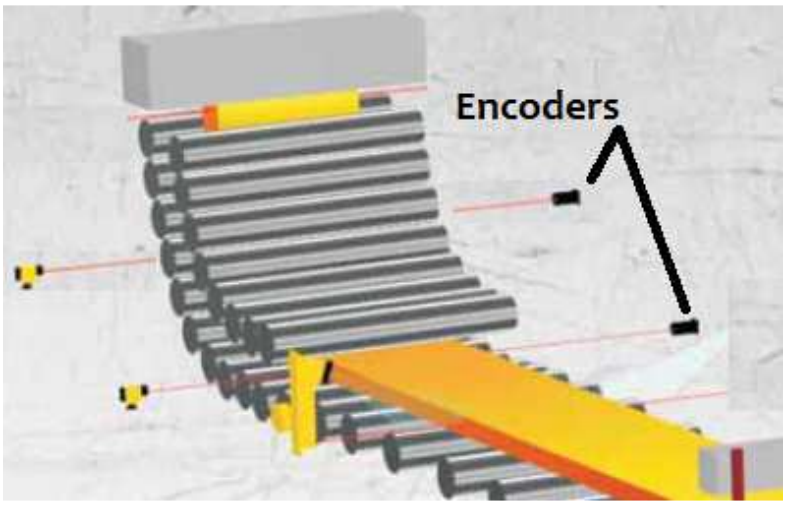

fig 3

Nuestra propuesta consiste en presentarle a la industria siderúrgica un método alternativo de gran versatilidad y con mejores prestaciones.

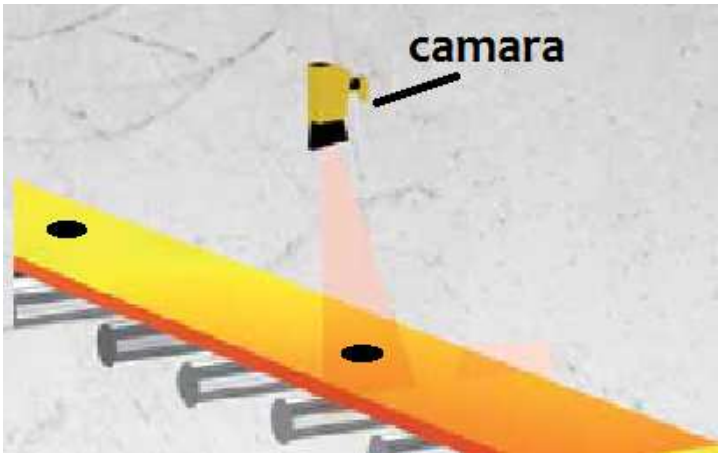

fig 4

Lo que ofrecemos es un proyecto que consta de software en gran medida con lo cual no requiere casi de mantenimiento, ya que solo necesita de la utilización de una cámara y una máquina con potencia de procesamiento en la cual se ejecute el sistema medidor.

MODELIZACION

Para la modelización del proceso se utiliza una plancha de telgopor (que representa el acero antes de la máquina de oxicorte) sobre el cual se efectúa una marca de pintura resistente a altas temperaturas y mediante el uso de una cámara web interpretaremos el movimiento de la misma haciendo efectivo el seguimiento, mientras que esta se va desplazando unidimensionalmente. Con sólo tener conocimiento de la distancia que recorre y tomando el tiempo en que lo hace podremos determinar su velocidad instantánea.

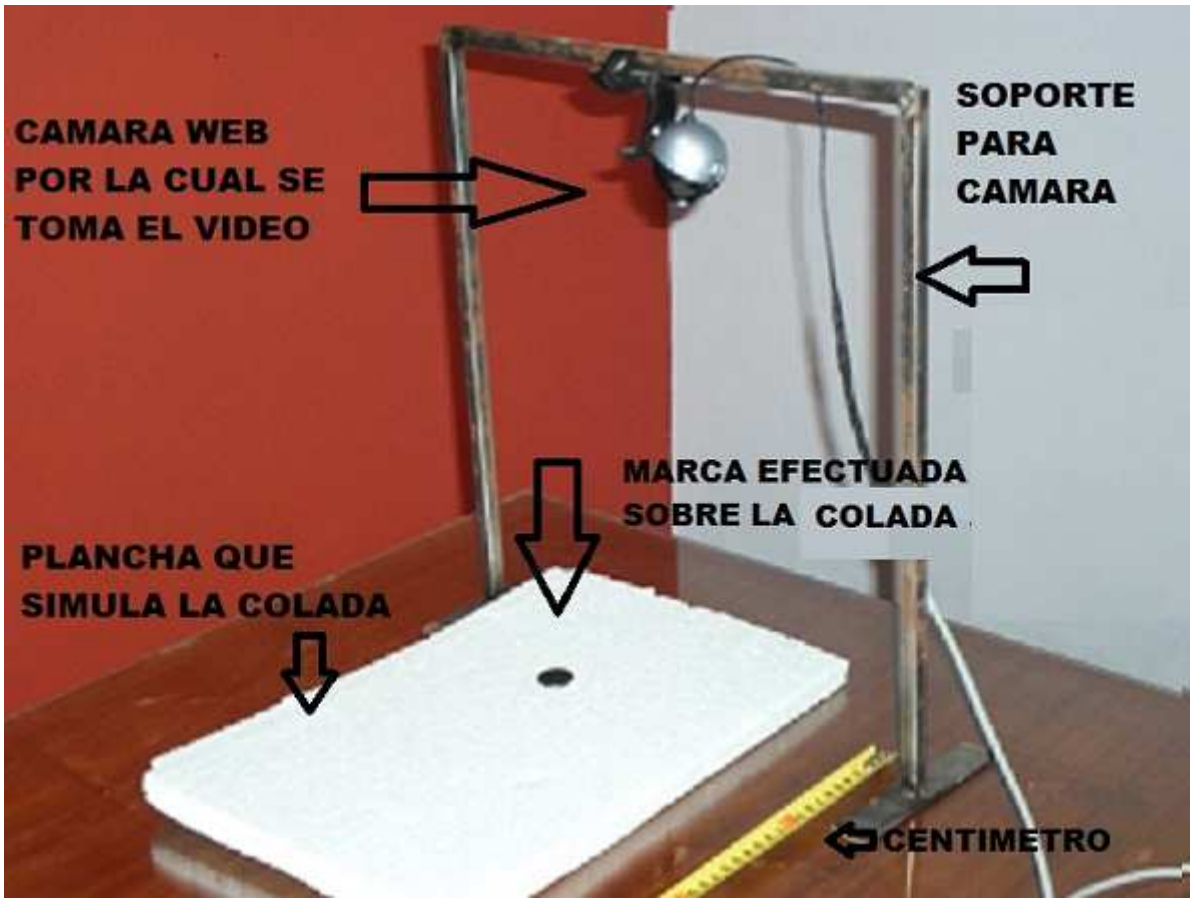

fig 5.

DESARROLLO

Mediante una cámara web conectada a través del puerto USB, son adquiridas las imágenes que luego serán interpretadas por un software desarrollado bajo el lenguaje de programación C y mediante el uso de las librerías OpenCV, IPL, realizamos un procesamiento digital de las imágenes obteniendo, en nuestro caso, la velocidad final.

¿POR QUÉ ELEGIMOS DevCpp?

Este programa posee un entorno de desarrollo amigable con el usuario y permite una fácil compilación del archivo generado, además permite generar un archivo ejecutable y de esta manera no se requiere licencia.
Mediante una máquina Windows se puede ejecutar sin problema.
En la documentación leida acerca de las librerías de OpenCV, nos encontramos que otros programas no son de fácil configuración para la utilización de las librerías de visión artificial, con lo cual optamos por hacer nuestro desarrollo con este entorno.

¿POR QUÉ ELEGIMOS LA LIBRERÍA OPENCV?

Estas librerías ofrecen funcionalidades para procesar imágenes y la programación se realiza en C, con lo cual estamos familiarizados.

¿QUÉ ES OPENCV?

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.
Open CV es multiplataforma, existiendo versiones para GNU/Linux, MAC OS X y Windows.

APLICACIONES CONOCIDAS

- OpenCV ha sido usada en el sistema de visión del vehículo no tripulado
- OpenCVse usa en sistemas de vigilancia de vídeo
- OpenCV es la clave en el programa Swistrack, una herramienta de seguimiento distribuida

- ✓ Sistemas operativos soportados: Windows7 / Vista/ XP
- ✓ Máxima resolución de imagen fija: 8 Mega pixels
- ✓ USB 1.1

fig 6.

Requerimientos del sistema:

- PC compatible con Intel Pentium 4 1,2 GHz ó superior
- Sistema operativo Microsoft Windows Vista/XP/2000
- 128 MB de RAM
- CD-ROM
- Puerto USB 1.1 ó 1.0 disponible
- 600 MB de espacio libre en el disco duro (se recomienda 260 MB)
- DirectX 9.0 ó superior

DESCRIPCIÓN DEL SISTEMA DE MEDICIÓN

fig 7.

Una vez que dado inicio al sistema, se espera que el PLC de la señal de detección en el nivel de colada. En este instante se envía una señal para notificarle al PLC que se realice una marca. Contamos con un feedback con la respuesta que se realizó la marca, iniciándose así un proceso de identificación de marca.
En esta primer imagen se ve que la colada esta quieta por lo tanto tenemos una velocidad de 0.00 cm/seg. El tiempo que se requiere para la cuenta de la velocidad se puede ajustar desde el software, y con esto ya entra en juego la resolución que vamos a tener de la velocidad instantánea. Dado que la velocidad de la colada no llega a ser mayor a 2 metros por minuto (aprox 3.4 cm/seg el máximo a calcular) para los aceros que tienen una velocidad de colado rápida, el procesamiento de las señales no necesitan de un scan demasiado rápido de las variables, lo cual nos juega a favor.

- ✓ Las imágenes que se ven son:
  - Video Captura: la imagen tratada en escala de grises.
  - Elipse final: Es la imagen de la detección de las elipses que ya es filtrada a una determinada medida que nos interesa a nosotros. Este parámetro también se puede modificar si fuera necesario.

Ambas explicadas anteriormente

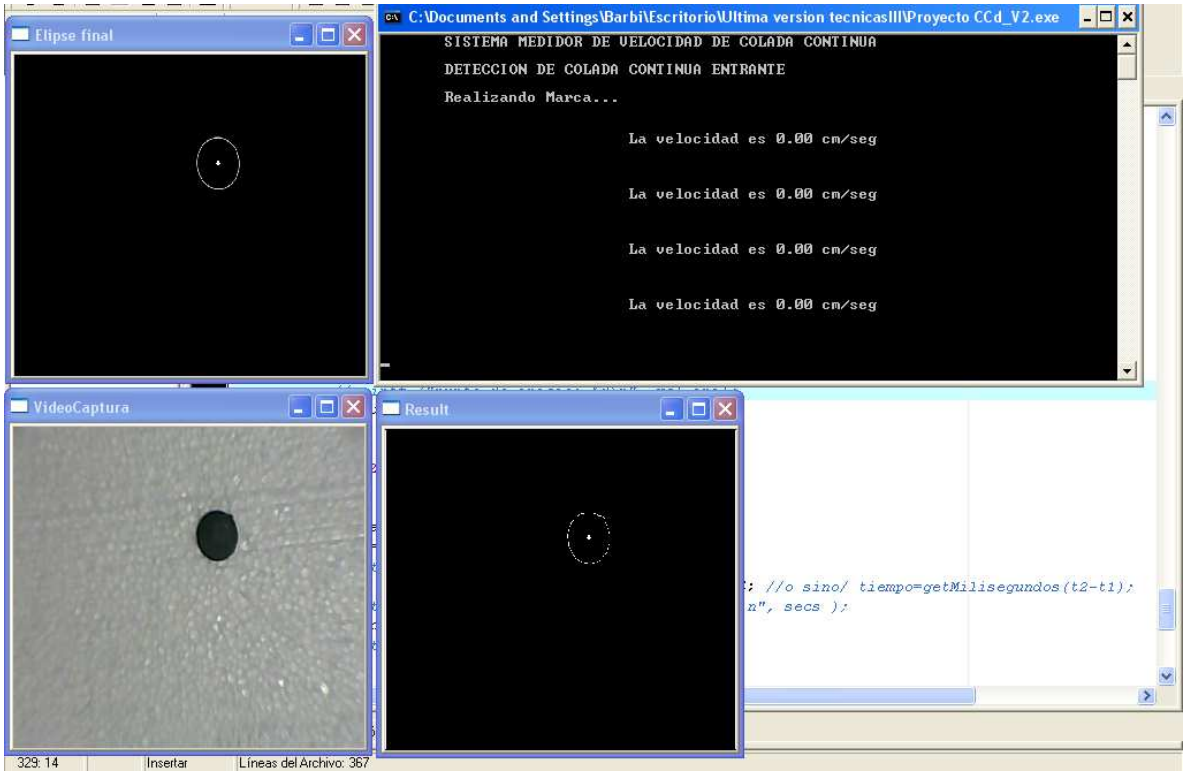

fig 8

En las figuras 9 y 10 empezamos a tener una velocidad en la colada, con lo cual vemos que el valor de la velocidad comienza a variar de acuerdo al desplazamiento de la marca.

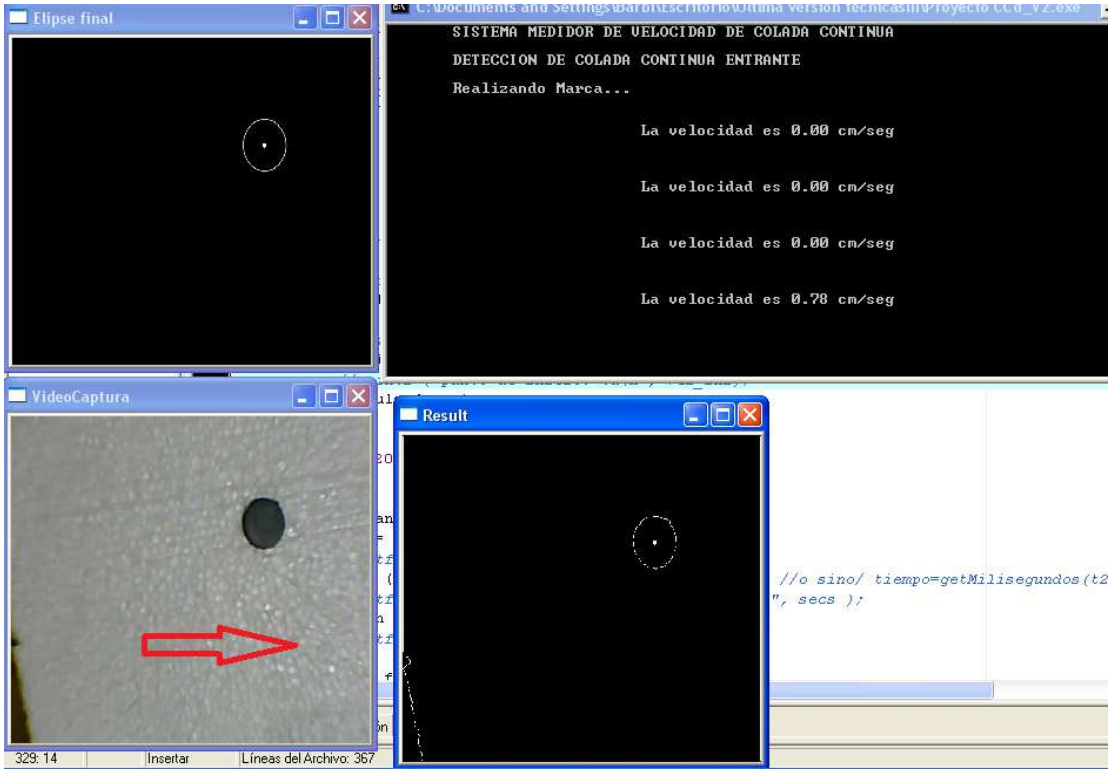

fig 9

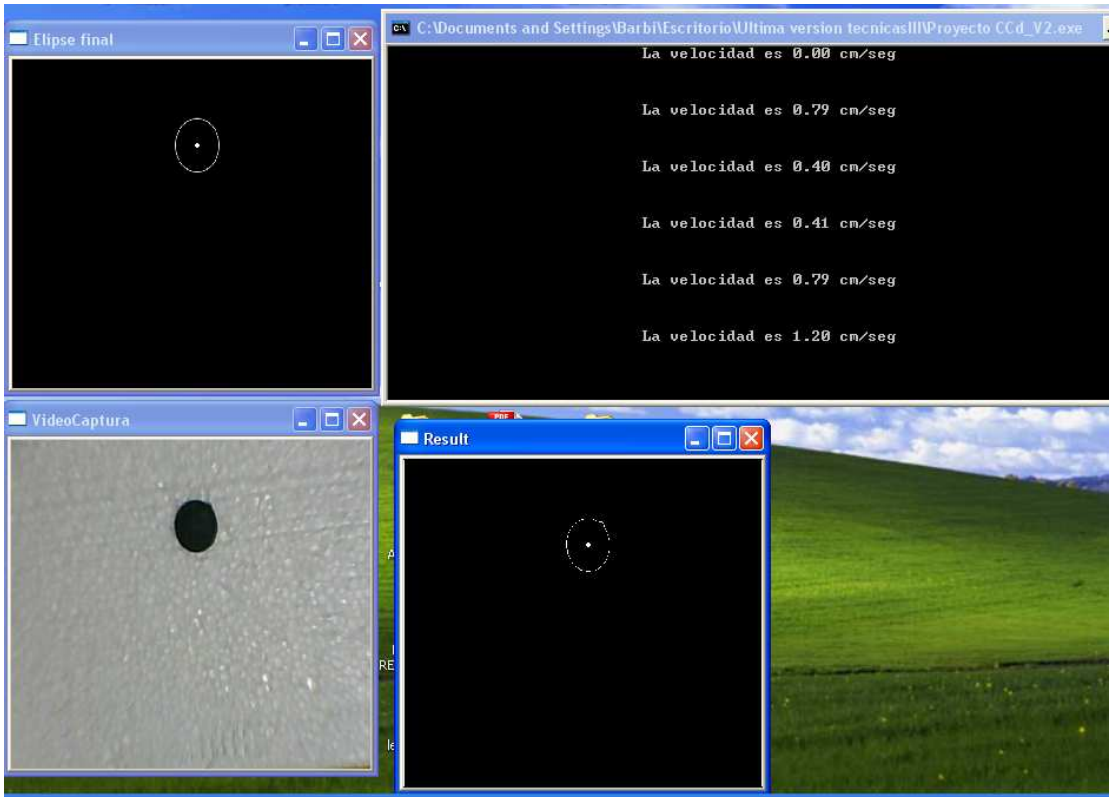

fig 10

Si la marca sobrepasa un limite fijado por el software, se da aviso al PLC que se debe realizar una nueva marca, y una vez hecho esto la lógica de control pasa identificar

solamente la nueva marca, de esta manera se continua con el cálculo de la velocidad sin interrupciones de tiempo.

Como observamos en las imágenes, se indica en la pantalla el aviso para que se realice una nueva marca. (fig 11)

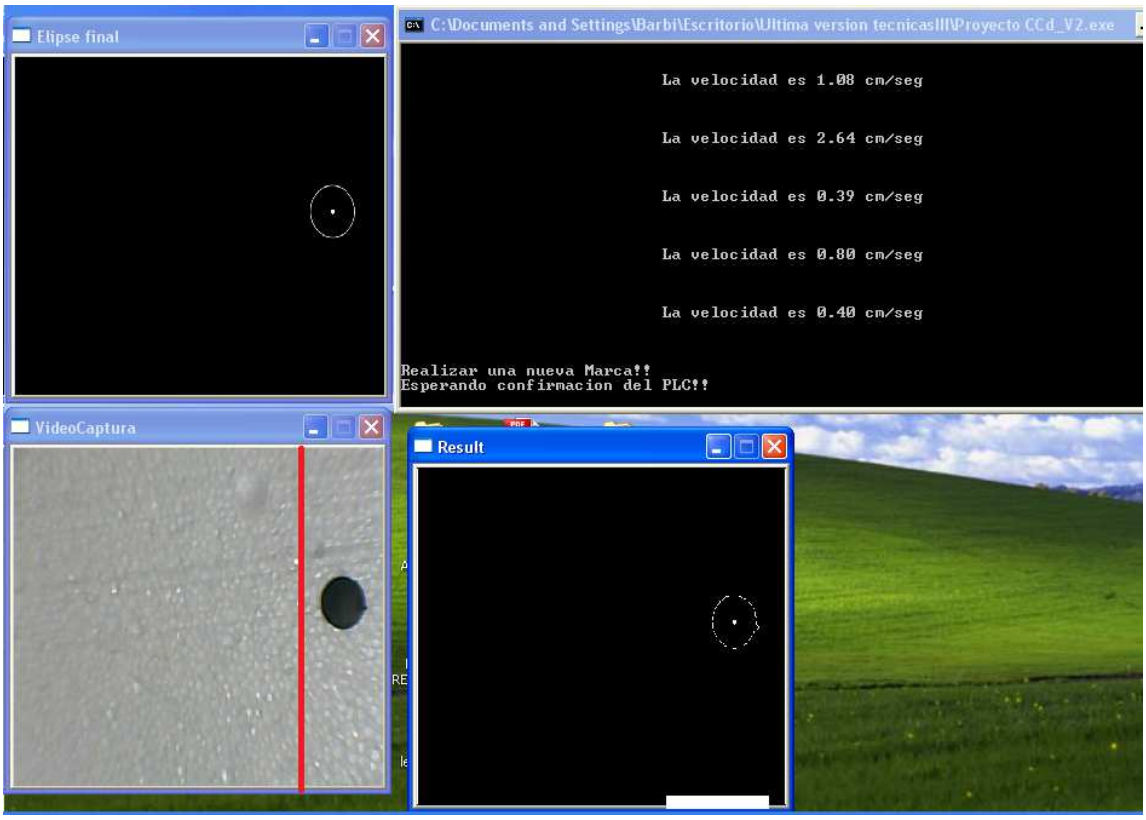

fig 11

En la fig 12 se observa lo descrito anteriormente, donde a medida que sigue avanzando la colada aparece la nueva marca y el sistema calcula la velocidad, despreciando las coordenadas de la marca que se encuentra a la derecha, la cual corresponde a la marca más antigua. Los ceros que aparecen son debido a que en la simulación no se siguió moviendo la marca cuando se realizó la nueva.

Ahora realizaremos una nueva marca, pero siguiendo siempre con una velocidad distinta de cero, es decir, antes y después de realizar la marca se continuó con el cálculo. (fig 13)

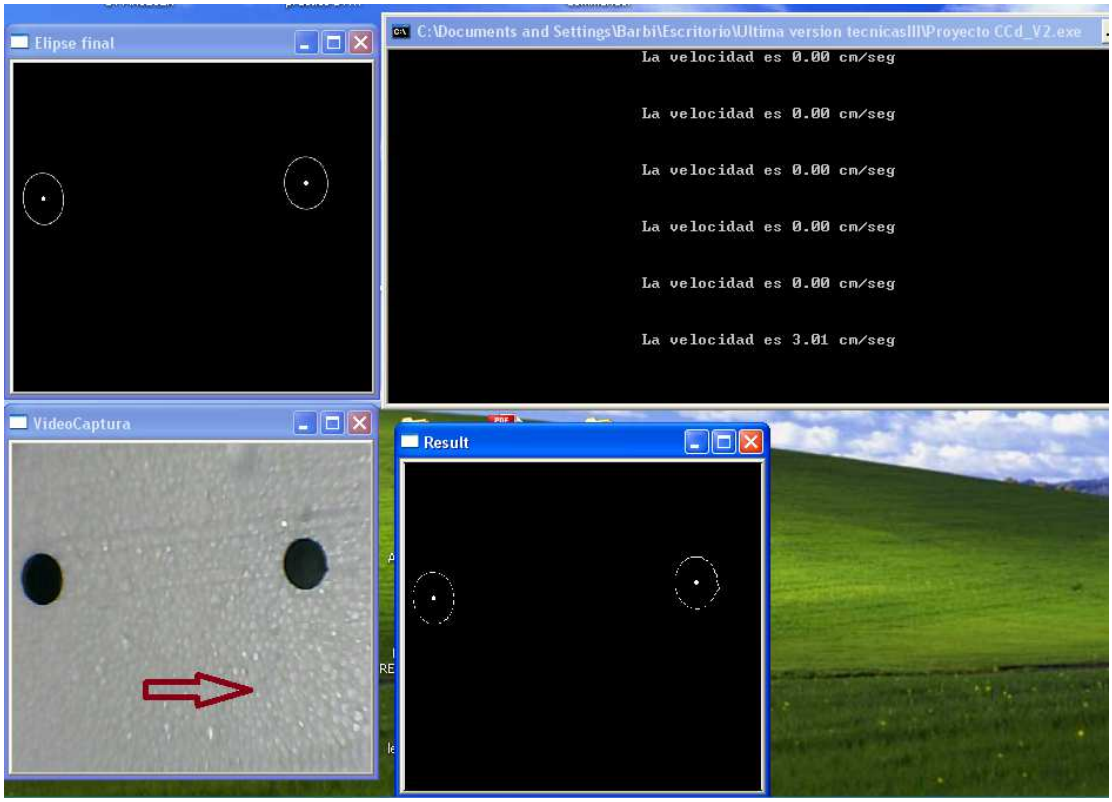
Elipse final
C:\Documents and Settings\Barbi\Escritorio\Ultima version tecnicasIII\Proyecto CCd_V2.exe
La velocidad es 0.00 cm/seg
La velocidad es 0.00 cm/seg
La velocidad es 0.00 cm/seg
La velocidad es 0.00 cm/seg
La velocidad es 0.00 cm/seg
La velocidad es 3.01 cm/seg
VideoCaptura
Result

fig 12

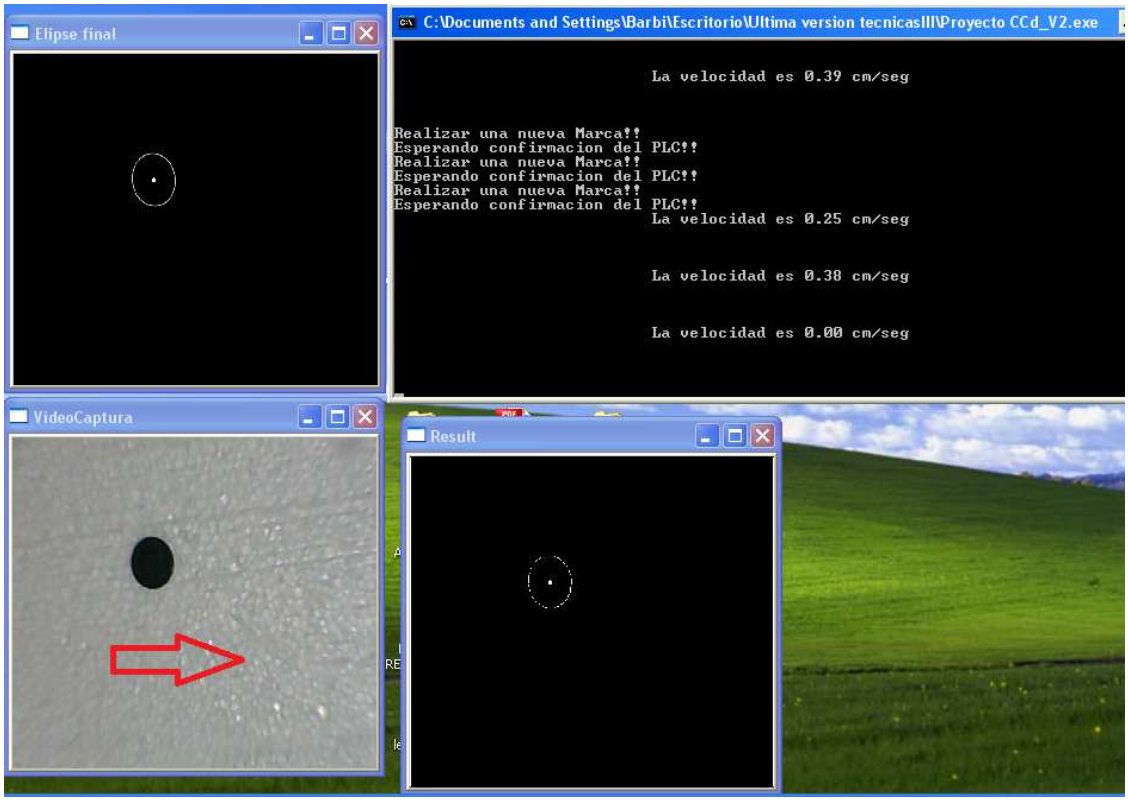
Elipse final
C:\Documents and Settings\Barbi\Escritorio\Ultima version tecnicasIII\Proyecto CCd_V2.exe
La velocidad es 0.39 cm/seg
Realizar una nueva Marca!!
Esperando confirmacion del PLC!!
Realizar una nueva Marca!!
Esperando confirmacion del PLC!!
Realizar una nueva Marca!!
Esperando confirmacion del PLC!!
La velocidad es 0.25 cm/seg
La velocidad es 0.38 cm/seg
La velocidad es 0.00 cm/seg
VideoCaptura
Result

fig 13

INCONVENIENTES

A la hora de efectuar las capturas con la cámara nos encontramos con ciertos inconvenientes como por ejemplo la luminosidad requerida para obtener una imagen adecuada para el proceso de tratamiento y posterior análisis. Para ello se realizan una serie de cálculos detallados a continuación.

Aclaración: Los datos no son estándar sino que varían dependiendo de donde se procederá a emplazar el proyecto.

ESTUDIO DE LUMINOSIDAD

Las cámaras requieren cierto nivel de luz para producir imágenes de buena calidad. Este nivel de luz se mide en foot-candles o lux. Un foot-candle o pié-candela es la medida de la intensidad de la luz producida por una vela a un metro de distancia. Aunque producirán imágenes aceptables en las condiciones de luz más pobres, la mayoría de las cámaras requieren un nivel de luz de 150 a 200 foot-candles para producir una fotografía o video de calidad óptima.
Para ello se realiza un estudio de luminosidad para determinar si la cantidad de lux en el plano de trabajo es la óptima para el empleo de nuestra cámara.

$$E_{med.} = \frac{\Phi_t}{A} \cdot \eta \cdot f_m$$

Donde:

E_{med} : Nivel medio de iluminación [Lux]

Φ_t : Flujo luminoso total de todas las luminarias [Lúmen]

A : Superficie total del plano de trabajo [m^2]

η : Factor de utilización para el plano de trabajo

f_m : Factor de mantenimiento

Para independizarnos de la luminosidad en el sector donde se va a disponer la cámara, incorporaremos un reflector que nos proporcionará la luz adecuada para la captura de las imágenes.

Optamos por la elección de un reflector cuyas características son las siguientes:

fig 14

- 16 Leds de alta potencia
- No genera interferencias con otros dispositivos electrónicos
- La iluminación proporcionada es homogénea y cerrada
- 2200 lúmenes de luz intensa
- Voltaje de 9 – 33 Volts
- Protección contra sobrecalentamiento gracias a un sensor térmico integrado
- Protección contra sobrecalentamiento

Realizaremos el estudio de luminosidad:

- Factor de utilización de 0,77
- Factor de mantenimiento de 0,6
- Superficie de trabajo 4m2
- Flujo del reflector 2200 lúmenes

Realizando los cálculos: Emed = 231 Lux

Los niveles de iluminación a utilizar deben estar de acuerdo con la tarea a realizar. La Comisión Internacional de Iluminación, recomienda los valores en función de las distintas actividades, los cuales se pueden consultar en diversos Manuales de Alumbrado.
A modo orientativo, damos a continuación algunos valores generales los cuales nos servirán para determinar si la luz de nuestro lugar de trabajo es el correcto:

- Zonas exteriores de circulación 20 Lux
- Circulación en industrias, depósitos 150 Lux
- Trabajos manuales simples 300 Lux
- Trabajos de oficina 500 a 700 Lux
- Trabajos finos manuales 1000 Lux

Con esto podemos comprobar que la iluminación es la correcta y no tendrían que producirse problemas para la captura de imágenes, aun si no existiese otra fuente de luz, mas que el reflector.

CODIGO FUENTE

```
#ifdef _CH_
#pragma package <opencv>
#endif

#ifndef _EiC
#include "cv.h"
#include "highgui.h"
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#endif

#define LIMITE_INFERIOR 500
#define LIMITE_SUPERIOR 550
#define ANCHO 35

int CalculoVelocidad(CvPoint centro);

int slider_pos = 70;
IplImage *data_img=NULL;
IplImage *image02=NULL, *image04=NULL, *image05=NULL, *img_result=NULL;
int x;
int               N=0; // contador para ciclos
int               contador = 0;
int               flag_nueva_marca = 0;
int               val_fin = 0;
static            float       p=640;
float             velocidad = 0;
int               punto_interes = 0;
int               val_ini = 0;
clock_t           t_ini, t_fin;
int               calculando = 0;
float             buf_punto[2] = {0};
int               k = 0;

CvPoint center;              //Coordenadas del punto de interes
CvPoint aux_centro;          // Auxiliar de Coordenadas



/*
/ Funcion para identificar la marca a seguir
*/
void process_image(int h)
{
    int dist_real=0;
    int dist_rel=0;
    double secs;
    int var;

    CvMemStorage* stor;
    CvSeq* cont;
    CvBox2D32f* box;
```

```
    CvPoint* PointArray;
    CvPoint2D32f* PointArray2D32f;

    // Create dynamic structure and sequence.
    stor = cvCreateMemStorage(0);
    cont = cvCreateSeq(CV_SEQ_ELTYPE_POINT, sizeof(CvSeq),
sizeof(CvPoint) , stor);

    // Threshold the source image. This needful for cvFindContours().
    cvThreshold( data_img, image02, slider_pos, 255,
CV_THRESH_BINARY );

    // Find all contours.
    cvFindContours( image02, stor, &cont, sizeof(CvContour),
                    CV_RETR_LIST, CV_CHAIN_APPROX_NONE, cvPoint(0,0));

    // Clear images. IPL use.
    cvZero(image02);
    cvZero(image04);
    cvZero(image05);

    // This cycle draw all contours and approximate it by ellipses.
    for(;cont;cont = cont->h_next)
    {
        int i;                                  // Indicador de ciclo.
        int count = cont->total;    // Numero de puntos de contorno
        CvSize size;

        // Number point must be more than or equal to 6 (for
cvFitEllipse_32f).
        if( count < 6 )
            continue;

        // Alloc memory for contour point set.
        PointArray = (CvPoint*)malloc( count*sizeof(CvPoint) );
        PointArray2D32f=
(CvPoint2D32f*)malloc( count*sizeof(CvPoint2D32f) );

        // Alloc memory for ellipse data.
        box = (CvBox2D32f*)malloc(sizeof(CvBox2D32f));

        // Get contour point set.
        cvCvtSeqToArray(cont, PointArray, CV_WHOLE_SEQ);

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

        // Fits ellipse to current contour.
        cvFitEllipse(PointArray2D32f, count, box);

        // Dibujo el contorno.

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

        // Convierto la informacion de la elipse de float a integer.
        center.x = cvRound(box->center.x);
```

```
        center.y = cvRound(box->center.y);

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

        if((size.width >10 && size.height >10)&& (size.width <150 &&
size.height <150) )
            {
                  cvEllipse(image05, center, size,box->angle, 0,
360,CV_RGB(0,0,255), 1, CV_AA, 0);

            cvCircle( image04,cvPoint(center.x,center.y), 2,
CV_RGB(255,255,255),2 );//dibuja circulo en centro

                  cvCircle( image05,cvPoint(center.x,center.y), 2,
CV_RGB(255,255,255),2 );

                  if(k != 0 )
                  {
                        var = center.x - aux_centro.x;
                  }

                  if(var > 100 && var < 200    )
                  {
                        printf(" ATENCION!! - - EXISTEN DOS MARCAS MUY
PROXIMAS!!!\n");

                  }
                  if(center.x < LIMITE_INFERIOR )
                  {
                        k+=1;
                        aux_centro.x = center.x;  // Auxiliar de
coordenada de interes
                        contador = 0;
                        CalculoVelocidad(center);

                  }

                  if(center.x > LIMITE_SUPERIOR)
                  {
                     contador++;
                     CalculoVelocidad(center);
                     if( contador > 25  )
                     {
                              printf("Realizar una nueva Marca!!\n");
                              printf("Esperando confirmacion del
PLC!!\n");
                              fflush(stdout);
                              k = 0;
                              getch();    //Simula Señal proveniente
del plc debido a una nueva marca
                              contador = 0;
                        }
                        return;
                   }

        }
        // Free memory.
        free(PointArray);
        free(PointArray2D32f);
```

```
        free(box);
    }

      // Show image. HighGUI use.
      cvShowImage( "Result", image04 );
    cvShowImage ( "Elipse final", image05);
}

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;
    char file[500];
    k = 0;

      printf("       SISTEMA MEDIDOR DE VELOCIDAD DE COLADA CONTINUA
\n\n");
    printf("       DETECCION DE COLADA CONTINUA ENTRANTE \n\n");
    printf("       Realizando Marca...    \n\n\n");
    fflush(stdout);

    capture = cvCaptureFromCAM(0);
    /*
            PODRIA CONFIGURARSE PARA QUE TENGA UN AJUSTE RESPECTO A LA
ALTURA DE LA CAMARA.
      */
      //printf("Ingrese dist maxima recorrida en centimetros: \n ");
    //scanf("%d",&x);
    if(!cvGrabFrame(capture))
    {
      printf("ERROR - - No hay Imagen a tomar\n");
      printf("Verificar el estado de la Camara\n");
      printf("%p\n",capture);
      fflush(stdout);
      exit(-1);
    }
    //Propiedades de la imagen
    cvQueryFrame (capture); // llamada esto es necesario para obtener
correcta
                            // Captura de propiedades
    frameH = (int) cvGetCaptureProperty(capture,
CV_CAP_PROP_FRAME_HEIGHT);
    frameW = (int) cvGetCaptureProperty(capture,
CV_CAP_PROP_FRAME_WIDTH);

    cvNamedWindow( "VideoCaptura",0);
    cvNamedWindow("Elipse final", 0);
```

```
    cvNamedWindow("Result", 0);

    while(1)
    {
      //capturamos la imagen
      cvGrabFrame(capture);
      imagen_orig = cvRetrieveFrame(capture);
            cvShowImage("VideoCaptura",imagen_orig);
            cvMoveWindow("VideoCaptura", 0, 400);    //cargo posicion
de pantalla

            //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();

            // Parametros de finalizacion
            char c = cvWaitKey(33);
            if( c == 27 )
        {
            break;
        }
    }
    cvQueryFrame(capture);

    //Libera la fuente de la captura
    cvReleaseCapture(&capture);
    cvDestroyWindow("VideoCaptura");
    cvDestroyWindow("contorno");
    cvDestroyWindow("Aproximacion por elipse");
    return (0);
}

int CalculoVelocidad(CvPoint centro)
{
    int dist_real=0;
    int dist_rel=0;
    double secs;
    time_t tiempo;
    struct tm *tlocal = localtime(&tiempo);
    char output[128];


    if (N == 0) t_ini = clock();          // tomo el tiempo de inicio

    if(center.x < LIMITE_INFERIOR)
    {
       N++;
         if(calculando == 0)
         {
          val_ini = center.x;
          calculando = 1;
       }

         if(N >= 20 && calculando == 1)
        {
             N = 0;
             calculando = 0;
```

```
            t_fin = clock();  // tomo el tiempo de final
            secs = (double)(t_fin - t_ini) / CLOCKS_PER_SEC; //o sino/
tiempo=getMilisegundos(t2-t1);
            val_fin = center.x;

            if(val_fin > val_ini)
            {
                  dist_rel= val_fin - val_ini;
                  dist_real= (dist_rel*ANCHO)/640; // en cm
            }
            else
            {
                  dist_rel= val_ini - val_fin;
                  dist_real= (dist_rel*ANCHO)/640; // en cm
            }

            velocidad= (dist_real)/(secs);
            printf("                              La velocidad es %0.2f
cm/seg\n\n\n\n",velocidad);

       }
   }
    return(1);
}
```