ANEXO C:
CÓDIGO Y ARCHIVOS DE LA APLICACIÓN CON CAMSHIFT

C.1.- Introducción:

A lo largo del capítulo 4 se mostraron todos los contenidos de la aplicación para la detección de caídas programada con el método Camshift, incluyendo los diagramas de flujo, estructura interna, funciones utilizadas en el código, resultados obtenidos y un ejemplo de la aplicación en funcionamiento.

En este anexo, vamos a incluir el código y el resto de archivos relevantes utilizados para la implementación del sistema.

C.2.- Código fuente

En este apartado incluiremos los archivos que se han utilizado en el entorno de programación: cabeceras.h, principal.cpp y funciones.cpp, que presentaremos a continuación.

C.2.1.- Archivo cabeceras.h

Este archivo contiene la declaración de las estructuras de datos que hemos utilizado para optimizar el funcionamiento del programa así como todas las funciones que luego emplearemos en el resto del código.

```
////////////////////////////////////
//   ARCHIVO CABECERAS.H   //
////////////////////////////////////
#include "cvaux.h"
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <math.h>

#define CONTADOR_MAX 5000
#define BARRA_HORIZ 10
#define BARRA_VERTIC 30

//Estructura de datos utilizada para la tarea de seguimiento
typedef struct Track
{
        IplImage* original;
        IplImage* imagenhsv;
        IplImage* imagenhue;
        IplImage* mask;
        IplImage* imagenhist;
        CvHistogram* histograma;
        IplImage* backproject;
        IplImage* mascaraflow;
        CvPoint origin;
        CvRect objeto;
        int objseleccionado;
        int seguir;
        CvRect ventanatrack;
        CvBox2D track_box;
        CvConnectedComp track_comp;
}
Track;

//Estructura de datos utilizada para el cálculo del flujo óptico y para la fase de entrenamiento
typedef struct OptFlow
{
        IplImage* imagenGris;
        IplImage* actual;
        IplImage* anterior;
        IplImage* piramidal;
        IplImage* ant_piramidal;
        IplImage* flujo_mov;
        IplImage* flujo_global;
        char* estado;
        int inicio_seg;
        int cont;
        int salto;
        CvPoint2D32f* puntos0;
        CvPoint2D32f* puntos1;
        float compxTotal;
        float compyTotal;
        float magnitud;
        FILE* fichero;
```

```
}
OptFlow;

//Declaración de las funciones propias utilizadas en el código
Track* CvCreateTrack(CvCapture* flujo);
OptFlow* CvCreateOptFlow(CvCapture* flujo);
void calcula_flujo(OptFlow* paramflow, IplImage* mascara, int entrenar);
void seguimiento(Track* paramtrack);
CvScalar hsv2rgb( float hue );
void on_mouse( int event, int x, int y, int flags, void* param );
void creaMascara(CvRect rect1, IplImage* mascara);
void pintaFlecha(CvPoint centro, CvPoint extremo, CvScalar color, int grosor, IplImage* imagen);
```

C.2.2.- Archivo principal.cpp

Este archivo contiene la definición de la función main(), a partir de la cual llamaremos al resto de funciones que hacen todas las tareas de creación e inicialización de las estructuras de datos, gestión de eventos con el ratón, seguimiento de los objetos de interés, cálculo del flujo del movimiento, gestión de las opciones del menú, grabación de vídeos, muestra de resultados y sobre todo detección de caídas.

```
////////////////////////////////////
//   ARCHIVO PRINCIPAL.CPP   //
////////////////////////////////////
#include "cvaux.h"
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include "cabeceras.h"


int main( int argc, char** argv )
{
        //Variables para la visualización de los resultados, grabación y entrenamiento
        int verhist = 1;
        int verglobal=0;
        int verflujo= 1;
        int grabar=0;
        int entrenar=0;

        //Variables para la detección de caídas
        int posAntesDeCaer=0;
        int enSuelo=0;
        bool caida=false;
        bool velocidadAlta=false;

        //Variables y estructuras de datos para captura de imágenes, seguimiento y cálculo de
        //flujo óptico
        CvCapture*      flujocamara = NULL;
        CvVideoWriter* video=NULL;
```

```
Track* trackcabeza=NULL;
Track* tracktorso=NULL;
OptFlow* flechas=NULL;
int trama,error;

//Inicialización de las estructuras
flujocamara = cvCaptureFromCAM(-1);
trackcabeza=CvCreateTrack(flujocamara);
tracktorso=CvCreateTrack(flujocamara);
flechas=CvCreateOptFlow(flujocamara);

 if((!trackcabeza->original)||(!tracktorso->original))
{
        printf("Revise por favor la camara. La conexion no es correcta \n");
        getchar();
        exit(0);
}

//Menú principal de la aplicación
printf( "Funcionamiento del programa: \n"
        "\tESC - Salir del programa\n"
        "\tc - Finalizar el seguimiento del objeto\n"
        "\th - ON-OFF del histograma del objeto\n"
        "\tf - ON-OFF del flujo parcial de los objetos en seguimiento\n"
        "\tg - ON-OFF del flujo global\n"
        "\te - Para comenzar/finalizar el entrenamiento\n"
        "\ts - Para comenzar/finalizar la grabacion de la prueba\n"
        "Para comenzar el seguimiento, seleccione el objeto con el raton \n" );

cvNamedWindow( "Histograma de la cabeza", 1 );
cvNamedWindow("Histograma del torso",1);
cvNamedWindow( "Ventana principal", 1 );
cvNamedWindow("Flujo optico de los objetos en seguimiento", 1);

//Bucle principal de la aplicación
for(trama=1;trackcabeza->original; trackcabeza->original = cvQueryFrame(flujocamara),
trama++ )
{
        //Gestión de los eventos de ratón
        if(trackcabeza->seguir>0)
        //Ya se ha seleccionado la cabeza y queremos seleccionar el cuerpo
        {
                cvSetMouseCallback( "Ventana principal", on_mouse, tracktorso );
        }
        else
        {
                cvSetMouseCallback("Ventana principal", on_mouse, trackcabeza);
        }

        cvZero(trackcabeza->mascaraflow);

        //Realizamos el seguimiento de los elementos de interés que se hayan
        //seleccionado en el ratón
        seguimiento(trackcabeza);
        //Creamos la máscara para calcular el flujo óptico sólo en la ventana donde se
        //encuentra el objeto que estamos siguiendo
        creaMascara(trackcabeza->ventanatrack, trackcabeza->mascaraflow);
        seguimiento(tracktorso);
        //trackcabeza->mascaraflow contendrá tanto el área de la cabeza como el
        //del cuerpo
        creaMascara(tracktorso->ventanatrack, trackcabeza->mascaraflow);
        cvCvtColor( trackcabeza->original, flechas->imagenGris, CV_BGR2GRAY );
```

```
if(trackcabeza->seguir>0)
{
	calcula_flujo(flechas, trackcabeza->mascaraflow, entrenar);
}

//Zona de gestión de detección de caídas
if(trackcabeza->track_box.center.y > posAntesDeCaer)
{
	posAntesDeCaer=trackcabeza->track_box.center.y;
}
if(!velocidadAlta && flechas->compyTotal<-12)
{
	velocidadAlta=true;
}

if(velocidadAlta && trackcabeza->track_box.center.y <= (posAntesDeCaer/4))
{
	enSuelo++;
	if(enSuelo>=150)
		caida=true;
}
else if(velocidadAlta && trackcabeza->track_box.center.y > 3*(posAntesDeCaer/4))
{
	velocidadAlta=false;
	printf("NOS HEMOS RECUPERADO DE UNA CAIDA\n");
	enSuelo=0;
}

if(caida)
{
	printf("CAIDA DETECTADA. ALARMA!!!\n");
	caida=false;
	velocidadAlta=false;
	enSuelo=0;
}

//Bloque donde se muestran los resultados obtenidos
cvShowImage( "Histograma de la cabeza", trackcabeza->imagenhist );
cvShowImage( "Histograma del torso", tracktorso->imagenhist );
cvFlip(flechas->flujo_mov, flechas->flujo_mov,0);
cvFlip(flechas->flujo_global, flechas->flujo_global, 0);
cvShowImage( "Flujo optico de los objetos en seguimiento", flechas->flujo_mov );
cvShowImage("Flujo optico global", flechas->flujo_global);

//Gestión del menú principal. Análisis de las teclas pulsadas
int c = cvWaitKey(10);
if( c == 27 )
	break;

switch( c )
{
case 'c':  //Reseteamos los objetos que estamos siguiendo
	if(tracktorso->seguir>0)
	{
		tracktorso->seguir=0;
	}
	else
	{
		trackcabeza->seguir=0;
	}
break;
case 'h':   //Activamos o desactivamos las vistas de los histogramas
verhist ^= 1;
```

```
if( !verhist )
{
	cvDestroyWindow( "Histograma de la cabeza" );
	cvDestroyWindow("Histograma del torso");
}
else
{
	cvNamedWindow( "Histograma de la cabeza", 1 );
	cvNamedWindow( "Histograma del torso", 1 );
}

break;
case 'f':
//Activar o desactivar la vista del flujo óptico parcial (con todos los vectores de
//desplazamiento
verflujo ^= 1;
if( !verflujo )
{
	cvDestroyWindow( "Flujo optico de los objetos en seguimiento" );
}
else
{
	cvNamedWindow( "Flujo optico de los objetos en seguimiento", 1 );
}
break;
case 'g':   //Activar o desactivar la vista del flujo óptico global
verglobal ^=1;
if(!verglobal)
{
	cvDestroyWindow("Flujo optico global");
}
else
{
	cvNamedWindow("Flujo optico global",1);
}
break;
case 'e':
//Activar o desactivar la fase de entrenamiento (donde escribiremos el valor del
//flujo global (magnitud y componentes x e y) en un archivo de texto
if(entrenar)
{
	entrenar=0;
	printf("Fin del entrenamiento\n");
}
else
{
	entrenar=1;
	printf("Comienzo del entrenamiento");
}
break;
case 's':
//Comenzar o finalizar la grabación de un video de la ventana principal durante
//la ejecución de la aplicación
double fps=cvGetCaptureProperty( flujocamara, CV_CAP_PROP_FPS );
double alto=cvGetCaptureProperty(flujocamara,CV_CAP_PROP_FRAME_HEIGHT);
double ancho=cvGetCaptureProperty(flujocamara,CV_CAP_PROP_FRAME_WIDTH);
double fourcc=cvGetCaptureProperty(flujocamara,CV_CAP_PROP_FOURCC);

if(grabar)
{
	grabar=0;
	cvReleaseVideoWriter( &video );
	printf("Grabacion parada\n");
```

```
                }
                else
                {
                        grabar=1;
                        video=cvCreateVideoWriter( "grabacion.avi", -1, 30, cvSize(ancho,alto),
                        1 );
                        printf("Grabando");
                }
                break;
                default:
                        break;
        }

        //Si la grabación está activa añadimos un nuevo frame a la secuencia de vídeo
        if(grabar)
        {
                error=cvWriteFrame( video, trackcabeza->original );
                printf(".");
        }
}

//Liberación de los recursos del sistema
cvReleaseCapture( &flujocamara );
cvDestroyWindow("Ventana principal");
cvDestroyWindow("Flujo optico de los objetos en movimiento");
cvDestroyWindow("Flujo optico global");
cvDestroyWindow("Histograma del torso");
cvDestroyWindow("Histograma de la cabeza");

return 0;
}
```

C.2.3.- Archivo funciones.cpp

Este archivo contiene todas las funciones que se llaman desde main() y que se encargan de realizar cada una de las tareas básicas de la aplicación.

```
////////////////////////////////////
//   ARCHIVO FUNCIONES.CPP   //
////////////////////////////////////

#include "cvaux.h"
#include "cv.h"
#include "highgui.h"
#include <time.h>
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include "cabeceras.h"

//Creación e inicialización de la estructura de datos para el seguimiento
Track* CvCreateTrack(CvCapture* flujo)
{
        int hdims = 16;
        float hranges_arr[] = {0,180};
        float* hranges = hranges_arr;
        Track* inicia=0;
        inicia = (Track *)cvAlloc( sizeof( *inicia ));
```

```
	inicia->objseleccionado=0;
	inicia->original = cvQueryFrame(flujo);
	inicia->seguir=0;
	inicia->imagenhsv = cvCreateImage( cvGetSize(inicia->original), 8, 3 );
	inicia->imagenhue = cvCreateImage( cvGetSize(inicia->original), 8, 1 );
	inicia->mask = cvCreateImage( cvGetSize(inicia->original), 8, 1 );
	inicia->mascaraflow=cvCreateImage( cvGetSize(inicia->original), 8, 1 );
	inicia->ventanatrack.x=0;
	inicia->ventanatrack.y=0;
	inicia->ventanatrack.height=0;
	inicia->ventanatrack.width=0;

	inicia->imagenhist = cvCreateImage( cvSize(320,200), 8, 3 );
	inicia->histograma = cvCreateHist( 1, &hdims, CV_HIST_ARRAY, &hranges, 1 );
	inicia->backproject = cvCreateImage( cvGetSize(inicia->original), 8, 1 );
	cvZero( inicia->imagenhist );
	return inicia;
}

//Creación en inicialización de la estructura de datos para el cálculo del flujo óptico
OptFlow* CvCreateOptFlow(CvCapture* flujo)
{
	OptFlow* inicia=0;
	IplImage* original=0;
	int i,j;
	inicia = (OptFlow*)cvAlloc( sizeof( *inicia ));
	inicia->cont=0;
	inicia->inicio_seg=0;
	inicia->salto=10;
	inicia->compxTotal=0;
	inicia->compyTotal=0;
	inicia->magnitud=0;
	inicia->estado = (char*)cvAlloc(CONTADOR_MAX);
	inicia->puntos0 = (CvPoint2D32f*)cvAlloc(CONTADOR_MAX*sizeof((inicia->puntos0)[0]));
	inicia->puntos1 = (CvPoint2D32f*)cvAlloc(CONTADOR_MAX*sizeof((inicia->puntos0)[0]));

	original = cvQueryFrame(flujo);

	//Para poder estudiar el movimiento de la imagen no podemos estudiar todos los
	//puntos, lo que haremos será centrarnos en puntos de la imagen separados entre sí por
	//una distancia o "salto" de 10 píxeles.

	for(i = 0; i < (original->height); i+=(inicia->salto))
	{
		for(j = 0; j < (original->width); j+=(inicia->salto))
		{
			(inicia->puntos0)[inicia->cont] = cvPoint2D32f(j, i);
			(inicia->puntos1)[inicia->cont] = cvPoint2D32f(j, i);
			(inicia->cont)++;
			// Contaremos el número total de puntos que vamos a considerar al
			//final.(Dependerá del tamaño de la imagen)
		}
	}
	inicia->inicio_seg=inicia->cont;

	inicia->imagenGris = cvCreateImage( cvGetSize(original), 8, 1 );
	inicia->actual = cvCreateImage( cvGetSize(original), 8, 1 );
	inicia->anterior = cvCreateImage( cvGetSize(original), 8, 1 );
	inicia->piramidal = cvCreateImage( cvGetSize(original), 8, 1 );
	inicia->ant_piramidal = cvCreateImage( cvGetSize(original), 8, 1 );
	inicia->flujo_mov = cvCreateImage( cvGetSize(original), 8, 1 );
	inicia->flujo_global = cvCreateImage( cvSize(320,200), 8, 3 );
	inicia->fichero=fopen("velocidades.txt","w+");
```

```
	return inicia;
}

//Función que calcula el flujo óptico de la zona de la imagen que no sea nula en la mascara. El
//resultado se almacena en una estructura del tipo OptFlow. Si estamos en fase de
//entrenamiento los resultados obtenidos se escriben en un archivo de texto.
void calcula_flujo(OptFlow* paramflow, IplImage* mascara, int entrenar)
{
	int i,j,contador;
	CvPoint centro=cvPoint(0,0);
	CvPoint extremo=cvPoint(0,0);
	CvRect rectTemp=cvRect(0,0,0,0);
	int bandera=0;
	float* componentex=NULL;
	float *componentey=NULL;


	componentex = (float*)cvAlloc(CONTADOR_MAX*sizeof(componentex[0]));
	componentey = (float*)cvAlloc(CONTADOR_MAX*sizeof(componentey[0]));

	//Suavizamos la imagen para mejorar la precisión
	cvSmooth(paramflow->imagenGris, paramflow->imagenGris, CV_GAUSSIAN, 3, 0, 0);
	cvSmooth(paramflow->imagenGris, paramflow->imagenGris, CV_GAUSSIAN, 3, 0, 0);
	cvSmooth(paramflow->imagenGris, paramflow->imagenGris, CV_GAUSSIAN, 3, 0, 0);

	//Aplicamos la máscara para que el flujo óptico se calcule sólo en los puntos
	//pertenecientes al objeto bajo seguimiento.
	cvAnd( paramflow->imagenGris, mascara, paramflow->imagenGris, 0 );

	//La imagen que en el bucle anterior era la actual, ahora es la imagen anterior.
	cvCopy(paramflow->actual, paramflow->anterior, 0);
	cvCopy(paramflow->imagenGris, paramflow->actual, 0);

	//Actualizamos también las imágenes que vamos a utilizar para el método iterativo
	//piramidal LK
	cvCopy(paramflow->piramidal, paramflow->ant_piramidal, 0);

	//Calculamos el flujo del movimiento utilizando el método iterativo piramidal LK con el
	//parámetro level=0 para que así el sistema se comporte como el método general de
	//Lucas-Kanade
	cvCalcOpticalFlowPyrLK(paramflow->anterior, paramflow->actual, paramflow->ant_piramidal,
	paramflow->piramidal, paramflow->puntos0, paramflow->puntos1, paramflow->cont,
	cvSize(paramflow->salto, paramflow->salto), 0, paramflow->estado, 0, cvTermCriteria(
	CV_TERMCRIT_ITER | CV_TERMCRIT_EPS,20,0.03), bandera);

	//Eliminamos el posible ruido que se produzca en la imagen.
	for(i = 0; i <paramflow->inicio_seg; i++)
	{
		if(abs((int)((paramflow->puntos0)[i].x - (paramflow->puntos1)[i].x)) > 50 | |
		abs((int)((paramflow->puntos0)[i].y - (paramflow->puntos1)[i].y)) > 50)
		{
			(paramflow->puntos1)[i] = (paramflow->puntos0)[i];
		}
	}

	//Actualizamos el valor de las banderas
	bandera |= CV_LKFLOW_PYR_A_READY;

	//Refrescamos la imagen del flujo del movimiento que estamos representando
	for (i = 0; i < (paramflow->flujo_mov)->height; i++)
	{
		for (j = 0; j < (paramflow->flujo_mov)->width; j++)
		{
```

```
                    ((paramflow->flujo_mov)->imageData + (paramflow->flujo_mov)->widthStep*i)
                    [j] = (char)255;  //Ponemos toda la imagen en blanco

            }
    }

    //Ahora dibujamos el flujo del movimiento. Lo dibujaremos con flechas. A mayor
    //velocidad del movimiento, mayor será el tamaño de la flecha.
    for (i = 0; i < paramflow->inicio_seg; i++)
    {
            //Dibujaremos sólo velocidades intermedias, ni muy lentas ni muy rápidas.
            if((abs((int)((paramflow->puntos0)[i].x - (paramflow->puntos1)[i].x)) >= 5 ||
            abs((int)((paramflow->puntos0)[i].y - (paramflow->puntos1)[i].y)) >= 5) &&
            (abs((int)((paramflow->puntos0)[i].x - (paramflow->puntos1)[i].x)) <= 40 &&
            abs((int)((paramflow->puntos0)[i].y - (paramflow->puntos1)[i].y)) <= 40))
            {
                    pintaFlecha(cvPointFrom32f((paramflow->puntos0)[i]), cvPointFrom32f
                    ((paramflow->puntos1)[i]), CV_RGB(0,0,0), 1, paramflow->flujo_mov);
            }
    }

    //Calculamos el flujo global
    cvZero(paramflow->flujo_global);
    for (i = 0; i < paramflow->inicio_seg; i++)
    {
            //Calculamos los vectores 2D suponiendo que todos tuviesen el mismo origen.

            componentex[i]=(paramflow->puntos1)[i].x - (paramflow->puntos0)[i].x;
            componentey[i]=(paramflow->puntos1)[i].y - (paramflow->puntos0)[i].y;

    }
    paramflow->compxTotal=0;
    paramflow->compyTotal=0;
    contador=0;
    for (i = 0; i < paramflow->inicio_seg; i++)
    {
            paramflow->compxTotal+=componentex[i];
            paramflow->compyTotal+=componentey[i];
            if((abs(componentex[i])!=0)&&(abs(componentey[i])!=0))
                    contador++;
    }
    paramflow->compxTotal=(paramflow->compxTotal)/contador;
    paramflow->compyTotal=(paramflow->compyTotal)/contador;
    rectTemp=cvRect(0,0,paramflow->flujo_global->width,paramflow->flujo_global->height );
    centro = cvPoint((rectTemp.x + rectTemp.width/2),(rectTemp.y + rectTemp.height/2));

    //Calculamos la magnitud y la escribimos en un archivo, junto con compxTotal y
    //compyTotal para realizar las pruebas.
    paramflow->magnitud = abs(sqrt((paramflow->compxTotal*paramflow->compxTotal)+
    (paramflow->compyTotal*paramflow->compyTotal)));

    //Para pintarlo lo multiplicaremos por 5 para que se vea mejor en la ventana de flujo
    //global.
    extremo=cvPoint(centro.x+5*(int)(paramflow->compxTotal),centro.y+5*(int)(paramflow->
    compyTotal));
    pintaFlecha(centro, extremo, CV_RGB(255,0,0), 4, paramflow->flujo_global);

    if(entrenar)
    {
            paramflow->fichero=fopen("velocidades.txt","a");
            fprintf(paramflow->fichero, "\nMagnitud: %f",paramflow->magnitud);
            fprintf(paramflow->fichero,"\tX: %f", paramflow->compxTotal);
            fprintf(paramflow->fichero,"\tY: %f", paramflow->compyTotal);
```

```
                fclose(paramflow->fichero);
        }
}

//Función para realizar el seguimiento de los objetos seleccionados utilizando el algoritmo
//Camshift
void seguimiento(Track* paramtrack)
{
        int i, anchobin;
        int vmin = 10, vmax = 256, smin = 30;
        int hdims = 16;
        float hranges_arr[] = {0,180};
        float* hranges = hranges_arr;

        cvCvtColor( paramtrack->original, paramtrack->imagenhsv, CV_BGR2HSV );
        //Pasamos del dominio RGB al dominio HSV para poder despues separar los planos del
        //color, la saturacion y el brillo.
        if( paramtrack->seguir )//Es distinto de 0 cuando ya tenemos seleccionado el objeto.
        {
                //Recordamos del artículo camshift que nosotros vamos a trabajar con el plano
                //HUE (color) y para que no haya mucho ruido es necesario tener un rango de
                //valores de los otros dos planos (saturacion y brillo) adecuado. Tomaremos una
                //máscara donde sólo tomaremos aquellos píxeles del plano HUE donde los
                //planos S y V tengan un valor dentro del rango estipulado.

                cvInRangeS( paramtrack->imagenhsv, cvScalar(0,smin,MIN(vmin,vmax),0),
                cvScalar(180,256,MAX(vmin,vmax),0), paramtrack->mask );

                //Separamos los planos HSV para trabajar sólo con el plano HUE.
                cvSplit( paramtrack->imagenhsv, paramtrack->imagenhue, 0, 0, 0 );

                if( paramtrack->seguir < 0 )
                //Si "seguimiento" es negativo significa que ya tenemos el objeto seleccionado y
                //que tenemos que inicializar todos los parametros para poder comenzar con el
                //tracking.
                {
                        float valormax = 0.f;

                        cvSetImageROI( paramtrack->imagenhue, paramtrack->objeto );
                        cvSetImageROI( paramtrack->mask, paramtrack->objeto );
                        cvCalcHist( &(paramtrack->imagenhue), paramtrack->histograma, 0,
                        paramtrack->mask );
                        cvGetMinMaxHistValue( paramtrack->histograma, 0, &valormax, 0, 0 );
                        cvConvertScale(paramtrack->histograma->bins,paramtrack->histograma->
                        bins, valormax ? 255. / valormax : 0., 0 );
                        cvResetImageROI( paramtrack->imagenhue );
                        cvResetImageROI( paramtrack->mask );
                        paramtrack->ventanatrack = paramtrack->objeto;
                        paramtrack->seguir=1;

                        cvZero( paramtrack->imagenhist );
                        anchobin = paramtrack->imagenhist->width / hdims;

                        //Pintamos el histograma del plano HUE. (Este bucle nos lo deberiamos
                        //saltar si no queremos ver el histograma, para reducir la carga
                        //computacional)
                        for( i = 0; i < hdims; i++ )
                        {
                                int val=cvRound(cvGetReal1D(paramtrack->histograma->bins,i)*
                                paramtrack->imagenhist->height/255 );
                                CvScalar color = hsv2rgb(i*180.f/hdims);
```

```
                                cvRectangle( paramtrack->imagenhist, cvPoint(i*anchobin,
                                paramtrack->imagenhist->height), cvPoint((i+1)*anchobin,
                                paramtrack->imagenhist->height - val), color, -1, 8, 0 );
                        }
                }

                cvCalcBackProject( &(paramtrack->imagenhue), paramtrack->backproject,
                paramtrack->histograma );
                cvAnd( paramtrack->backproject, paramtrack->mask, paramtrack->backproject,
                0 );
                cvCamShift(paramtrack->backproject,paramtrack->ventanatrack, cvTermCriteria
                ( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ), &paramtrack->track_comp,
                &paramtrack->track_box );
                paramtrack->ventanatrack = paramtrack->track_comp.rect;

                if( paramtrack->original->origin )
                        paramtrack->track_box.angle = -paramtrack->track_box.angle;

                cvEllipseBox( paramtrack->original, paramtrack->track_box, CV_RGB(255,0,0), 3,
                CV_AA, 0 );

        }

        if( paramtrack->objseleccionado && paramtrack->objeto.width > 0 && paramtrack->
        objeto.height > 0 )
        {
                cvSetImageROI( paramtrack->original, paramtrack->objeto );
                cvXorS( paramtrack->original, cvScalarAll(255), paramtrack->original, 0 );
                cvResetImageROI( paramtrack->original );
        }

        if (paramtrack->seguir==0)
        {
                cvZero(paramtrack->imagenhist);
        }
        cvShowImage( "Ventana principal", paramtrack->original );
}

//Función para pasar del espacio HSV al espacio RGB a partir del valor hue exclusivamente
CvScalar hsv2rgb( float hue )
{
   int rgb[3], p, sector;
   static const int sector_data[][3]=
      {{0,2,1}, {1,2,0}, {1,0,2}, {2,0,1}, {2,1,0}, {0,1,2}};
   hue *= 0.033333333333333333333333333333333f;
   sector = cvFloor(hue);
   p = cvRound(255*(hue - sector));
   p ^= sector & 1 ? 255 : 0;

   rgb[sector_data[sector][0]] = 255;
   rgb[sector_data[sector][1]] = 0;
   rgb[sector_data[sector][2]] = p;

   return cvScalar(rgb[2], rgb[1], rgb[0],0);
}

//Función que se va a ejecutar cuando ocurra algún evento de ratón. Es una función propia del
//sistema que nosotros hemos sobreescrito y así hemos conseguido que haga otras tareas
//específicas de la aplicación
void on_mouse( int event, int x, int y, int flags, void* param )
{
        if( !((Track*)param)->original )
      return;
```

```
        if( ((Track*)param)->original->origin )
                y = ((Track*)param)->original->height - y;

        if( ((Track*)param)->objseleccionado )
        {
                ((Track*)param)->objeto.x = MIN(x,((Track*)param)->origin.x);
                ((Track*)param)->objeto.y = MIN(y,((Track*)param)->origin.y);
                ((Track*)param)->objeto.width = ((Track*)param)->objeto.x + CV_IABS(x -
                ((Track*)param)->origin.x);
                ((Track*)param)->objeto.height = ((Track*)param)->objeto.y + CV_IABS(y -
                ((Track*)param)->origin.y);

                ((Track*)param)->objeto.x = MAX( ((Track*)param)->objeto.x, 0 );
                ((Track*)param)->objeto.y = MAX( ((Track*)param)->objeto.y, 0 );
                ((Track*)param)->objeto.width = MIN( ((Track*)param)->objeto.width,
                ((Track*)param)->original->width );
                ((Track*)param)->objeto.height = MIN( ((Track*)param)->objeto.height,
                ((Track*)param)->original->height );
                ((Track*)param)->objeto.width -= ((Track*)param)->objeto.x;
                ((Track*)param)->objeto.height -= ((Track*)param)->objeto.y;
        }

        switch( event )
        {
        case CV_EVENT_LBUTTONDOWN:
                ((Track*)param)->origin = cvPoint(x,y);
                ((Track*)param)->objeto = cvRect(x,y,0,0);
                ((Track*)param)->objseleccionado = 1;
        break;
        case CV_EVENT_LBUTTONUP:
                ((Track*)param)->objseleccionado = 0;
                if( ((Track*)param)->objeto.width > 0 && ((Track*)param)->objeto.height > 0 )
                        ((Track*)param)->seguir = -1;
        break;
        }
}

//Función que utilizamos para crear una máscara que filtré a que zonas de la imagen general se
//les va a calcular el flujo óptico. La imagen resultado será distinta de cero sólo en aquellos
//píxeles que caigan dentro del rectángulo que se le pasa como parámetro
void creaMascara(CvRect rect1, IplImage* mascara)
{
                int ptx1,ptx2,pty1,pty2;
                CvPoint esquina1,esquina2;

                ptx1=rect1.x;
                pty1=rect1.y;
                esquina1=cvPoint(ptx1,pty1);

                ptx2=ptx1 + rect1.width;
                pty2=pty1 + rect1.height;
                esquina2=cvPoint(ptx2,pty2);

                cvRectangle( mascara, esquina1, esquina2, CV_RGB( 255, 255, 255 ), CV_FILLED,
                8, 0);
}

//Función que pinta una flecha sobre la imagen resultado tras especificarle el origen (centro), el
//extremo, su color y su grosor.
void pintaFlecha(CvPoint centro, CvPoint extremo, CvScalar color, int grosor, IplImage* imagen)
{
        CvPoint p1,p2,p3,p4,p5;
```

```
            //Dibujamos la flecha
        //Primero dibujamos el palo de la flecha
        cvLine(imagen, centro, extremo, color, grosor, 8, 0);

        //Ahora dibujamos la cabeza de la flecha

        p1 = cvPoint((int)(centro.x +0.7*(extremo.x - centro.x)),(int)(centro.y +0.7*(extremo.y -
        centro.y)));

        p2 = cvPoint((int)(centro.x + 0.7*(extremo.x - centro.x)),(int)(extremo.y));

        p3 = cvPoint((int)(extremo.x),(int)(centro.y + 0.7*(extremo.y - centro.y)));

        p4 = cvPoint((int)(p1.x + 0.5*(p2.x-p1.x)),(int)(p1.y + 0.5*(p2.y-p1.y)));

        p5 = cvPoint((int)(p1.x + 0.5*(p3.x-p1.x)),(int)(p1.y + 0.5*(p3.y-p1.y)));

        cvLine(imagen, extremo,p4, color, grosor, 8, 0);
        cvLine(imagen, extremo,p5, color, grosor, 8, 0);
}
```

C.3.- Archivos de entrenamiento

Una de las tareas que se realizan dentro de la aplicación para la detección de caídas en lo que hemos llamado el proceso de "entrenamiento", consistente en guardar los datos del flujo óptico en un archivo llamado "velocidades.txt" para su posterior análisis. (Ver capítulo 4 de la memoria).

En este apartado del anexo C introduciremos todos los archivos de entrenamiento que hemos utilizado para implementar posteriormente el bloque de detección de caídas.

- Caminando hacia la derecha

  El siguiente archivo muestra los resultados recogidos cuando el individuo monitorizado caminaba delante de la cámara hacia la derecha. En este caso debemos fijarnos en la columna X y ver que en la mitad del archivo sus valores son positivos y más altos.

Magnitud: 1.468560	X: 0.538674	Y: 1.366198
Magnitud: 0.560630	X: 0.272364	Y: -0.490024
Magnitud: 0.813385	X: 0.508722	Y: 0.634663
Magnitud: 0.588234	X: 0.425613	Y: -0.406046
Magnitud: 0.247211	X: 0.004182	Y: -0.247176
Magnitud: 0.739384	X: -0.692991	Y: -0.257783
Magnitud: 0.536828	X: -0.208135	Y: -0.494837
Magnitud: 1.122235	X: -0.480246	Y: 1.014285
Magnitud: 0.720044	X: -0.623617	Y: -0.359951
Magnitud: 3.304136	X: -3.205635	Y: 0.800760
Magnitud: 4.247765	X: -2.962498	Y: 3.044193
Magnitud: 2.699028	X: 2.154197	Y: -1.626095

Magnitud: 7.952993	X: 6.406253	Y: -4.712750
Magnitud: 10.932403	X: 10.911811	Y: -0.670686
Magnitud: 20.191422	X: 20.124352	Y: -1.644383
Magnitud: 15.747050	X: 15.562123	Y: -2.406225
Magnitud: 8.401192	X: 8.081139	Y: -2.296786
Magnitud: 16.359251	X: 16.101891	Y: -2.890364
Magnitud: 9.337334	X: 8.487645	Y: -3.891745
Magnitud: 13.968642	X: 13.920696	Y: -1.156359
Magnitud: 14.195039	X: 13.221289	Y: 5.166881
Magnitud: 9.905160	X: 8.970634	Y: -4.199989
Magnitud: 13.105165	X: 13.094182	Y: -0.536420
Magnitud: 7.488142	X: 7.388402	Y: -1.218114
Magnitud: 8.349691	X: 8.094889	Y: 2.046978
Magnitud: 11.174879	X: 9.353848	Y: -6.114201
Magnitud: 13.083258	X: 12.499627	Y: -3.864058
Magnitud: 8.017443	X: 8.014261	Y: -0.225834
Magnitud: 15.907577	X: 10.810344	Y: -11.669938
Magnitud: 11.123585	X: 10.910153	Y: -2.168569
Magnitud: 7.597823	X: 6.218141	Y: -4.365963
Magnitud: 5.617374	X: 1.939486	Y: 5.271934
Magnitud: 6.649305	X: -0.275342	Y: -6.643601
Magnitud: 6.583995	X: 3.596356	Y: -5.514999
Magnitud: 3.120320	X: 3.046705	Y: -0.673781
Magnitud: 3.927038	X: -0.006877	Y: -3.927032
Magnitud: 9.162764	X: 6.672994	Y: 6.279124
Magnitud: 8.382149	X: -2.917745	Y: -7.857937
Magnitud: 3.857533	X: 0.043747	Y: -3.857285
Magnitud: 3.372196	X: -1.555796	Y: -2.991856
Magnitud: 10.096255	X: -0.873117	Y: -10.058431

- Caminando hacia la izquierda

En este caso el individuo monitorizado caminaba hacia la izquierda en línea recta. Podemos ver cómo al final del archivo los valores de la componente X son valores negativos a una velocidad alta.

Magnitud: 2.807162	X: 0.154020	Y: 2.802933
Magnitud: 2.153896	X: -0.360015	Y: 2.123595
Magnitud: 1.999270	X: 1.043415	Y: 1.705393
Magnitud: 2.251868	X: -1.461324	Y: -1.713313
Magnitud: 3.136569	X: 2.037284	Y: 2.384857
Magnitud: 0.541842	X: -0.081897	Y: -0.535617
Magnitud: 0.675930	X: -0.414537	Y: -0.533891
Magnitud: 1.510040	X: 1.492190	Y: 0.231492
Magnitud: 1.757297	X: 1.674513	Y: -0.533011
Magnitud: 1.352310	X: 1.147190	Y: 0.716029
Magnitud: 2.740215	X: 2.645215	Y: -0.715271
Magnitud: 2.261547	X: 1.696124	Y: -1.495914
Magnitud: 6.615939	X: 6.178232	Y: -2.366452
Magnitud: 0.993436	X: -0.928518	Y: -0.353225
Magnitud: 5.248528	X: -5.128435	Y: 1.116336
Magnitud: 5.329375	X: -5.246970	Y: 0.933568
Magnitud: 12.812160	X: -12.739141	Y: 1.365921
Magnitud: 10.522974	X: -10.505765	Y: 0.601575
Magnitud: 14.950202	X: -14.819407	Y: 1.973254
Magnitud: 17.604574	X: -17.202826	Y: 3.739497
Magnitud: 14.030187	X: -13.816848	Y: 2.437387
Magnitud: 13.990543	X: -13.979901	Y: 0.545567
Magnitud: 9.599620	X: -9.577799	Y: -0.646889

Magnitud: 16.944189	X: -16.661234	Y: -3.083641
Magnitud: 15.309711	X: -14.483500	Y: 4.961398
Magnitud: 13.652371	X: -13.556150	Y: 1.618028
Magnitud: 7.888581	X: -7.757256	Y: -1.433418
Magnitud: 11.257977	X: -11.257867	Y: -0.049633
Magnitud: 9.286100	X: -9.255980	Y: 0.747319
Magnitud: 6.086011	X: -6.075154	Y: 0.363359
Magnitud: 9.778582	X: -9.727675	Y: -0.996493
Magnitud: 11.382223	X: -11.089823	Y: 2.563365

- Agachándose y levantándose:

  En este archivo el individuo primero se agacha y luego se levanta, pudiendo ver las variaciones principalmente en la columna Y, donde primero son valores grandes negativos (cuando se mueve hacia abajo) y a continuación son valores grandes positivos (cuando se mueve hacia arriba).

Magnitud: 1.100057	X: 0.159825	Y: -1.088384
Magnitud: 1.087581	X: -0.842902	Y: 0.687275
Magnitud: 0.982504	X: 0.921069	Y: 0.341975
Magnitud: 0.588908	X: -0.587953	Y: 0.033526
Magnitud: 2.144949	X: -1.394219	Y: 1.630019
Magnitud: 0.815884	X: 0.127746	Y: -0.805821
Magnitud: 0.911255	X: 0.851316	Y: 0.325034
Magnitud: 2.091236	X: -0.390309	Y: -2.054489
Magnitud: 3.368507	X: 3.318645	Y: -0.577435
Magnitud: 1.963989	X: 1.108640	Y: -1.621163
Magnitud: 2.525550	X: -2.152172	Y: 1.321574
Magnitud: 5.991036	X: -5.934486	Y: -0.821209
Magnitud: 12.147485	X: -9.531168	Y: -7.531150
Magnitud: 14.121705	X: -1.375588	Y: -14.054548
Magnitud: 19.787205	X: -6.451525	Y: -18.705915
Magnitud: 15.652926	X: -3.364563	Y: -15.287047
Magnitud: 19.008823	X: -0.899203	Y: -18.987543
Magnitud: 9.316585	X: 4.278950	Y: -8.275828
Magnitud: 14.761410	X: 3.507355	Y: 14.338677
Magnitud: 17.701809	X: 2.562311	Y: 17.515383
Magnitud: 20.410053	X: 2.133992	Y: 20.298185
Magnitud: 20.857897	X: 6.352516	Y: 19.866993
Magnitud: 8.266241	X: 2.110594	Y: 7.992255
Magnitud: 8.442638	X: 0.396286	Y: 8.433332
Magnitud: 13.706350	X: -3.899615	Y: -13.139903
Magnitud: 0.531556	X: -0.384601	Y: -0.366925
Magnitud: 2.618180	X: -2.418546	Y: -1.002746

- Caidas

  Finalmente introduciremos varios archivos donde hemos monitorizado caídas hacia la derecha. (El sentido de la caída no es importante, pues lo importante es el valor de la componente Y que es la que vamos a utilizar en el bloque de detección).

Magnitud: 2.096749 X: 1.432293 Y: 1.531304
Magnitud: 1.001429 X: -0.485890 Y: -0.875654
Magnitud: 2.164265 X: -1.041314 Y: -1.897290
Magnitud: 1.913873 X: 1.737504 Y: 0.802489
Magnitud: 1.973792 X: -1.956308 Y: 0.262133
Magnitud: 2.443596 X: 0.517718 Y: -2.388122
Magnitud: 3.901063 X: 0.780508 Y: 3.822186
Magnitud: 2.204567 X: 0.464712 Y: 2.155031
Magnitud: 1.437782 X: 0.665849 Y: -1.274309
Magnitud: 17.081806 X: 4.335157 Y: -16.522545
Magnitud: 21.904276 X: 7.361527 Y: -20.630201
Magnitud: 21.356203 X: 8.935193 Y: -19.397158
Magnitud: 15.562628 X: 9.877689 Y: -12.026081
Magnitud: 24.526310 X: -1.590014 Y: -24.474716
Magnitud: 3.136529 X: -2.841638 Y: 1.327745
Magnitud: 1.087060 X: 1.087052 Y: 0.004048
Magnitud: 4.528768 X: -2.108114 Y: -4.008191
Magnitud: 0.232033 X: 0.051300 Y: -0.226291
Magnitud: 0.365510 X: 0.003964 Y: 0.365489
Magnitud: 0.352104 X: 0.283773 Y: 0.208448
Magnitud: 0.010297 X: -0.010220 Y: -0.001255
Magnitud: 2.055579 X: -0.635498 Y: 1.954878
Magnitud: 2.576385 X: 1.981014 Y: -1.647222
Magnitud: 2.701202 X: -2.400807 Y: 1.237990
Magnitud: 1.080513 X: 0.785236 Y: 0.742234
Magnitud: 2.545384 X: 1.719277 Y: -1.876984
Magnitud: 2.967975 X: 0.717481 Y: 2.879947
Magnitud: 2.078212 X: 1.241235 Y: -1.666823
Magnitud: 4.020632 X: 3.653966 Y: 1.677502
Magnitud: 13.286957 X: 13.008237 Y: -2.707211
Magnitud: 4.756941 X: 4.215223 Y: -2.204629
Magnitud: 16.225399 X: -3.188236 Y: -15.909077
Magnitud: 15.370746 X: -1.057351 Y: -15.334335
Magnitud: 26.030607 X: 2.614192 Y: -25.899006
Magnitud: 17.688150 X: 14.613181 Y: -9.966225
Magnitud: 14.118281 X: 13.849752 Y: -2.740480
Magnitud: 8.743793 X: -2.527719 Y: -8.370457
Magnitud: 0.137303 X: -0.002075 Y: -0.137287
Magnitud: 0.004869 X: 0.001149 Y: -0.004731
Magnitud: 0.006100 X: -0.001218 Y: 0.005977

****

Magnitud: 1.006063 X: 0.271189 Y: 0.968824
Magnitud: 0.585328 X: -0.211201 Y: -0.545896
Magnitud: 0.338361 X: 0.059154 Y: 0.333150
Magnitud: 2.668411 X: 1.235229 Y: -2.365296
Magnitud: 0.883562 X: -0.446956 Y: 0.762175
Magnitud: 0.421651 X: 0.417127 Y: 0.061600
Magnitud: 0.750459 X: 0.743782 Y: -0.099883
Magnitud: 2.431611 X: 0.615931 Y: 2.352310
Magnitud: 1.484984 X: -1.111766 Y: -0.984457
Magnitud: 1.867152 X: 1.751681 Y: 0.646429
Magnitud: 0.461856 X: 0.461817 Y: -0.006028
Magnitud: 3.148563 X: 0.938078 Y: 3.005572
Magnitud: 2.939767 X: 2.749598 Y: 1.040163
Magnitud: 0.998479 X: 0.214413 Y: -0.975186
Magnitud: 2.771448 X: 2.725786 Y: -0.501017
Magnitud: 3.728824 X: 3.231400 Y: 1.860694
Magnitud: 5.787602 X: 5.724151 Y: -0.854655
Magnitud: 7.050607 X: 6.312953 Y: 3.139693

Magnitud: 4.759448 X: 4.530247 Y: 1.459178
Magnitud: 8.313344 X: 5.481571 Y: -6.250126
Magnitud: 14.698734 X: 5.654259 Y: -13.567687
Magnitud: 11.376502 X: 2.545855 Y: -11.087986
Magnitud: 11.091211 X: 3.395599 Y: -10.558640
Magnitud: 6.663713 X: 2.204287 Y: -6.288576
Magnitud: 5.564795 X: 3.608950 Y: -4.235850
Magnitud: 6.502960 X: 6.157846 Y: -2.090315
Magnitud: 8.279779 X: 6.899513 Y: -4.577278

****

Magnitud: 1.345667 X: -1.301285 Y: -0.342749
Magnitud: 0.901313 X: -0.710263 Y: 0.554879
Magnitud: 0.909192 X: -0.089887 Y: 0.904738
Magnitud: 1.994702 X: -1.526997 Y: -1.283401
Magnitud: 0.758329 X: -0.053277 Y: 0.756455
Magnitud: 2.262830 X: -1.784591 Y: 1.391271
Magnitud: 2.183005 X: -2.075235 Y: 0.677429
Magnitud: 4.894014 X: -4.884053 Y: 0.312084
Magnitud: 3.415247 X: -3.021536 Y: -1.591926
Magnitud: 4.071490 X: -2.287345 Y: -3.368247
Magnitud: 8.174546 X: 2.361447 Y: -7.826032
Magnitud: 19.604855 X: 0.672011 Y: -19.593334
Magnitud: 0.373925 X: 0.031087 Y: 0.372630
Magnitud: 16.062475 X: 2.059757 Y: -15.929862
Magnitud: 8.993740 X: 8.988207 Y: -0.315447
Magnitud: 11.189619 X: 10.768157 Y: -3.042098
Magnitud: 10.409554 X: 4.236816 Y: -9.508322
Magnitud: 13.192039 X: 12.684807 Y: -3.622922
Magnitud: 13.613721 X: 13.610525 Y: 0.294965
Magnitud: 6.420521 X: -2.148064 Y: -6.050530
Magnitud: 4.114828 X: -4.078700 Y: 0.544079
Magnitud: 2.915766 X: -0.666088 Y: -2.838665
Magnitud: 8.800695 X: 6.147394 Y: 6.297761
Magnitud: 1.469494 X: 0.939614 Y: -1.129839
Magnitud: 5.737048 X: -4.695362 Y: -3.296558
Magnitud: 3.621485 X: -3.614799 Y: -0.219951

****

Magnitud: 0.006983 X: 0.002917 Y: 0.006345
Magnitud: 0.009132 X: -0.001338 Y: -0.009033
Magnitud: 0.004107 X: 0.002615 Y: 0.003167
Magnitud: 0.000962 X: -0.000422 Y: -0.000865
Magnitud: 0.006392 X: 0.003936 Y: 0.005036
Magnitud: 0.003552 X: -0.003499 Y: -0.000616
Magnitud: 0.000528 X: 0.000181 Y: 0.000496
Magnitud: 7.181561 X: -1.790602 Y: -6.954751
Magnitud: 3.587649 X: 0.335080 Y: 3.571966
Magnitud: 0.552366 X: 0.538264 Y: -0.124017
Magnitud: 1.718366 X: -0.975220 Y: 1.414825
Magnitud: 2.051868 X: 1.729924 Y: -1.103415
Magnitud: 4.687149 X: 0.780699 Y: -4.621674
Magnitud: 2.248416 X: 1.486757 Y: -1.686692
Magnitud: 2.593735 X: -1.254116 Y: 2.270387
Magnitud: 4.997261 X: 2.393361 Y: -4.386848
Magnitud: 2.607078 X: 0.054341 Y: 2.606511
Magnitud: 8.539289 X: 8.410937 Y: -1.474987
Magnitud: 9.411078 X: 9.385674 Y: -0.691017

Magnitud: 13.696712	X: 11.126191	Y: -7.987978
Magnitud: 7.740927	X: 6.817200	Y: -3.667115
Magnitud: 11.617105	X: 9.176367	Y: -7.124002
Magnitud: 18.108568	X: 13.246213	Y: -12.347392
Magnitud: 16.249292	X: -2.529271	Y: -16.051239
Magnitud: 2.638503	X: -1.221503	Y: -2.338723
Magnitud: 0.780988	X: -0.241906	Y: -0.742579
Magnitud: 1.718108	X: -1.569768	Y: -0.698373