Carrera: ***Maestría en Ciencias de la Ingeniería***
Materia: ***Visión Artificial***
Profesor: ***Dr. Eddie Helbert Clemente Torres***

Resumen de habilidades desarrolladas *Fecha:* 16-Feb-2016

Alumnos: Zúñiga Alvarez Francisco Fernando *No.Control:* G04650183
Pozos Flores Angel Manrique *No.Control:* M07211505

Examen

Dado un video donde exista un objeto en movimiento, construir una nueva secuencia de imágenes en donde se vea la estela del objeto en movimiento.

Para cumplir el objetivo desarrollamos el siguiente código.

```
#include <opencv2\opencv.hpp>
#include <stdlib.h>
#include <cv.hpp>
#include <cxcore.hpp>
#include <highgui.h>
#include <iostream>

using namespace std;
using namespace cv;



//Función Threshold
IplImage* GetThresholdedImage(IplImage* img)
{
	// Convierte la imagen en una imagen HSV
	IplImage* imgHSV = cvCreateImage(cvGetSize(img), 8, 3);
	cvCvtColor(img, imgHSV, CV_BGR2HSV);

	IplImage* imgThreshed = cvCreateImage(cvGetSize(img), 8, 1);
	cvInRangeS(imgHSV, cvScalar(20, 100, 100), cvScalar(30, 255, 255), imgThreshed);
	cvReleaseImage(&imgHSV);

	return imgThreshed;
}



int main()
{
	// inicia la captura desde la cámara
	CvCapture* capture = 0;
	capture = cvCaptureFromCAM(1);

	// Si no encuentra un dispositivo muestra un error y se cierra
	if (!capture)
	{
		printf("No se pudo iniciar la captura...");
		return -1;
	}

	// Se utilizarán 2 ventanas
	cvNamedWindow("video");
	cvNamedWindow("thresh");

	// Esta imagen almacenara los datos de
	// la posición rastreada de la pelota
	IplImage* imgScribble = NULL;

	// Un ciclo infinito
	while (true)
	{
		// almacenara un frame capturado de la cámara
		IplImage* frame = 0;
		frame = cvQueryFrame(capture);
```

```
// si no puede capturar un cuadro... se cierra
if (!frame)
	break;

// si es el primer cuadro, es necesario inicializarlo
if (imgScribble == NULL)
{
	imgScribble = cvCreateImage(cvGetSize(frame), 8, 3);
}

// almacena el threshold amarillo de la imagen (amarillo=blanco, resto = negro)
IplImage* imgYellowThresh = GetThresholdedImage(frame);

// calcula los momentos para estimar la posición de la pelota
CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
cvMoments(imgYellowThresh, moments, 1);

// los valores reales de los momentos
double moment10 = cvGetSpatialMoment(moments, 1, 0);
double moment01 = cvGetSpatialMoment(moments, 0, 1);
double area = cvGetCentralMoment(moments, 0, 0);

// almacenando la última y la posición actual de la pelota
static int posX = 0;
static int posY = 0;

int lastX = posX;
int lastY = posY;

posX = moment10 / area;
posY = moment01 / area;

//muestra la posición en la línea de comandos
printf("position (%d,%d)", posX, posY);

// Se dibuja una línea solo si es una posición valida
if (lastX>0 && lastY>0 && posX>0 && posY>0)
{

	// Dibuja el contorno amarillo del objeto
	cvLine(imgScribble, cvPoint(posX, posY), cvPoint(lastX, lastY), cvScalar(0,
	255, 255), 70);
}

// agrega el rastro a la imagen...
cvAdd(frame, imgScribble, frame);
cvShowImage("thresh", imgYellowThresh);
cvShowImage("video", frame);

// espera por una tecla
int c = cvWaitKey(60);
if (c != -1)
{
	// si se presiona termina el programa
	break;
}
```

```
			// libera la imagen del threshold y los momentos
			cvReleaseImage(&imgYellowThresh);
			delete moments;
		}

		//libera la cámara
		cvReleaseCapture(&capture);
		return 0;
}
```

Como resultado se muestra una secuencia de imágenes, donde se puede ver su funcionamiento siguiendo el movimiento de una pelota.

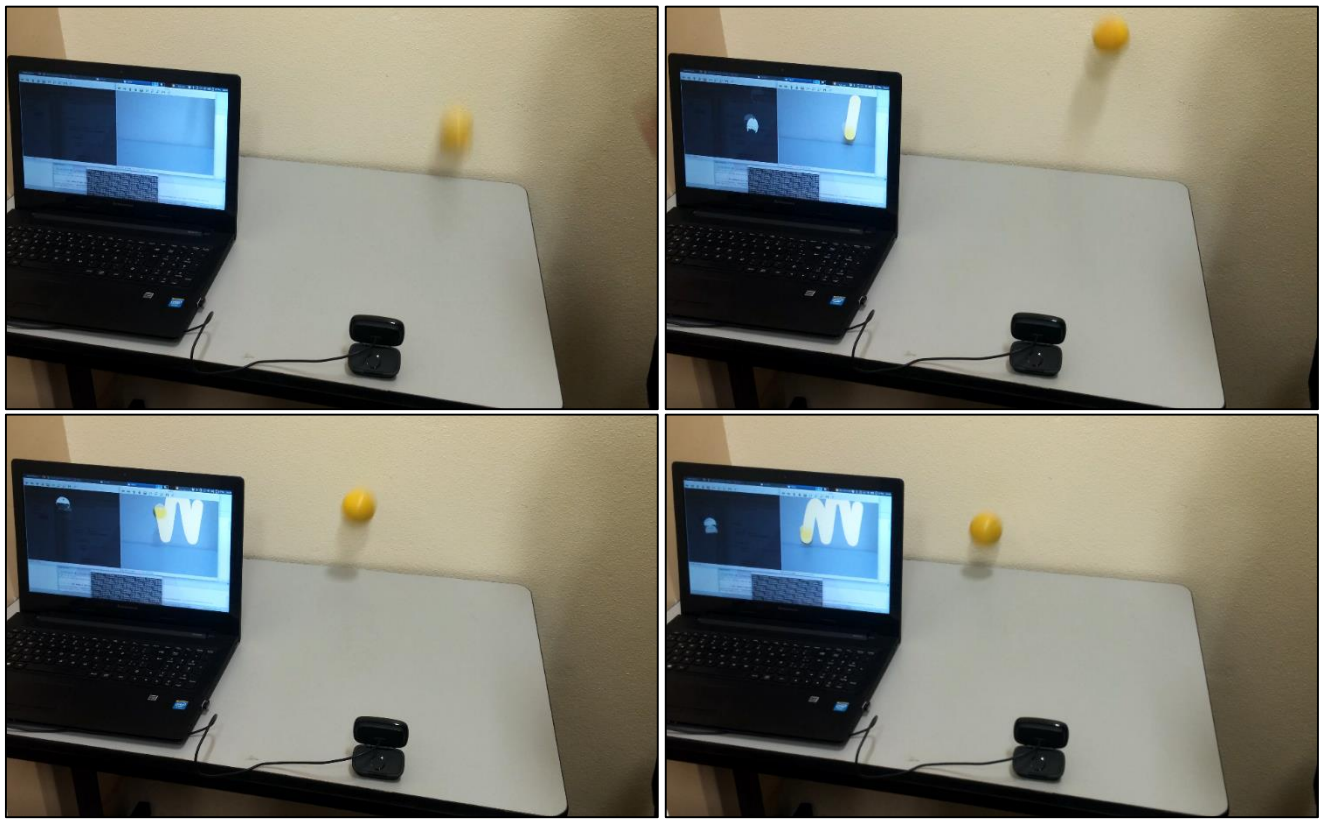

Figura 1. Secuencia de seguimiento de una pelota

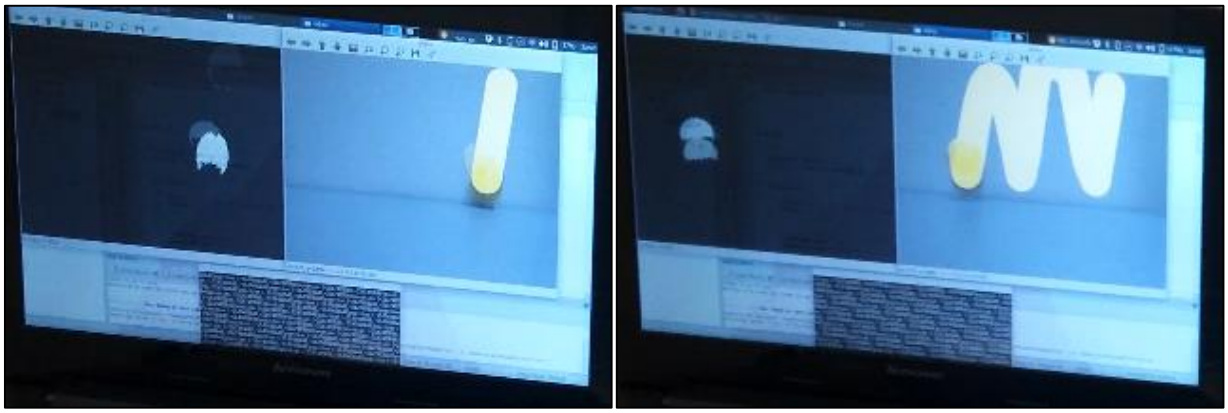

Figura 2. Silueta de la pelota en movimiento

Conclusiones

Este algoritmo toma el thresholding del objeto que se desea analizar y muestra la silueta de las posiciones donde pasa, tomamos la idea de hacer esto basándonos en la tarea 3, donde nosotros definimos el threshold del color amarillo de la pelota y las demás componentes fueron eliminadas manipulando el objeto en formato HSV.

Esto con el fin de que pudiera dejar el rastro de su silueta color amarillo cuando pasaba frente a la cámara, tomando las posiciones de los momentos anteriores y actuales y mostrándolos en pantalla, también mostramos en consola dichas posiciones y con la función cvLine escribimos dichas posiciones en tiempo real sobre la pantalla, generando con ello la silueta del objeto.

Decidimos que el algoritmo corriera en tiempo real ya que tuvimos bastantes problemas para poder procesar con video cargado, pues por errores de formato y además de no conocer a detalle la forma en la que trabajan algunas de las funciones, en ocasiones los videos cargados que queríamos utilizar mostraban variaciones como solo mostrar algunos de los colores o simplemente nos daba error a la hora de compilarlo y correrlo, al hacerlo en tiempo real como se decidió implementar no se mostraron errores y la velocidad de procesamiento del objeto y video fue muy buena como se muestra en las figuras 1 y 2.

Referencias

[1] http://opencv-srf.blogspot.mx/2011/09/object-detection-tracking-using-contours.html

[2] http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html

[3] http://shervinemami.info/blobs.html

[4] G. Bradski, A. Kaehler, Learning OpenCV, Tracking and Motion, Cap. 10., pg. 316.