ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

TEMA: "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA CONTABILIZACIÓN DE LA CANTIDAD DE OBJETOS QUE CIRCULAN POR EL ESPACIO CAPTADO POR UNA CÁMARA UTILIZANDO LIBRERÍAS DE OPENCV"

AUTORES:
ESTEFANÍA DAYANA MULLO LÓPEZ
CARLOS ANDRÉS MORENO MOLINA

DIRECTOR: ING. EDDIE GALARZA Z.
CODIRECTOR: ING. JOSÉ BUCHELI A.

LATACUNGA

2016

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Certificamos que el trabajo de titulación, ***"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA CONTABILIZACIÓN DE LA CANTIDAD DE OBJETOS QUE CIRCULAN POR EL ESPACIO CAPTADO POR UNA CÁMARA UTILIZANDO LIBRERÍAS DE OPENCV"*** realizado por los señores: ***ESTEFANÍA DAYANA MULLO LÓPEZ y CARLOS ANDRÉS MORENO MOLINA***, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, por lo tanto nos permitimos acreditarlo y autorizar a los señores: ***ESTEFANÍA DAYANA MULLO LÓPEZ y CARLOS ANDRÉS MORENO MOLINA*** para que lo sustenten públicamente.

Latacunga, 14 de Septiembre del 2016

----------	----------
Ing. Eddie Galarza	Ing. José Bucheli
DIRECTOR	CODIRECTOR

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORÍA DE RESPONSABILIDAD

Nosotros, ***ESTEFANÍA DAYANA MULLO LÓPEZ***, con cédula de identidad N° 050378918-2 y ***CARLOS ANDRÉS MORENO MOLINA*** con cédula de identidad N° 050350329-4, declaramos que este trabajo de titulación **"*DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA CONTABILIZACIÓN DE LA CANTIDAD DE OBJETOS QUE CIRCULAN POR EL ESPACIO CAPTADO POR UNA CÁMARA UTILIZANDO LIBRERÍAS DE OPENCV*"** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaramos que este trabajo es de nuestra autoría, en virtud de ello nos declaramos responsables del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 14 de Septiembre del 2016

Estefanía Dayana Mullo López

C.C.: 050378918-2

Carlos Andrés Moreno Molina

C.C.: 050350329-4

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Nosotros, ***ESTEFANÍA DAYANA MULLO LÓPEZ*** y ***CARLOS ANDRÉS MORENO MOLINA***, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **"*DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA CONTABILIZACIÓN DE LA CANTIDAD DE OBJETOS QUE CIRCULAN POR EL ESPACIO CAPTADO POR UNA CÁMARA UTILIZANDO LIBRERÍAS DE OPENCV*"** cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

Latacunga, 14 de Septiembre del 2016

----------	----------
Estefanía Dayana Mullo López	Carlos Andrés Moreno Molina
C.C.: 050378918-2	C.C.: 050350329-4

DEDICATORIA

Dedico este trabajo de titulación en primer lugar a Dios por darme cada día la oportunidad de adquirir conocimientos, a mi madre por ser el pilar fundamental de mi vida, e inculcarme cada día a ser una persona de buenos valores y enseñarme siempre que la educación será el escudo para enfrentarme al mundo y la vida, a mi padre que a la distancia siempre tenía un mensaje de motivación para seguir adelante, a mi hermano Álex quién con sus palabras y ejemplo profesional, me mostró que todo lo que nos propongamos podemos cumplirlo, y que el camino del conocimiento aún está por recorrerlo, a ti Andrés mi compañero de trabajo, por cada ayuda mutua que nos dimos y por cada palabra de aliento que nos llevó a concluir con nuestro objetivo de forma satisfactoria, y por último mi pequeñita Arleth que llegaste en el momento más preciso a cambiar mis días obscuros por aquellos de luz, alegría y amor puro.

Dayana

Dedico este trabajo a mi querido padre, Jorge Gustavo Moreno Clavijo, por ser mi mayor ejemplo de humildad y superación, por haberme enseñado a afrontar la vida y porque eres la motivación que me impulsa a alcanzar mis metas.

Andrés

AGRADECIMIENTO

Gracias infinitas a ti mi Dios por tus bendiciones por tu amor por todos los instrumentos que me diste para llegar hasta donde ahora estoy, a mi madre por la vida por ser quién día a día me enseñó a ser una mujer de bien por cada palabra que me llevó siempre por las sendas correctas de vida, a mi hermano Álex por tus consejos y alientos para no desvanecer ante cualquier circunstancia difícil que se presentaba.

A ti Andrés gracias por siempre tener la manera más adecuada de mostrarme que yo puedo, que siempre existe una solución para cada problema, por la paciencia que me dabas esos días que las cosas se complicaban, por tu comprensión y amor.

Por último, quiero agradecer a mis profesores en especial a mi tutor Ing. Eddie por guiarnos de la mejor manera posible para ejecutar el trabajo y corregir nuestras fallas siempre motivándonos a desarrollar el proyecto de la manera más adecuada y correcta, también a mi cotutor Ing. José gracias por siempre estar preocupado en que desarrollemos esto de la mejor manera cumpliendo siempre con nuestros objetivos.

Dayana

AGRADECIMIENTO

Agradezco a mi amada familia por su apoyo incondicional en cada etapa de mi vida, en especial a mi padre y mis hermanos, Myriam y Javier, porque sus consejos y sabiduría han sido y serán mi mayor fortaleza para alcanzar mis metas.

De igual forma a ti Dayana, gracias por tu ayuda, por tu tenacidad para mejorar las cosas y más que nada gracias porque he aprendido de ti la valentía para superar las dificultades.

Por último, agradezco a nuestros tutores, Ing. Eddie e Ing. José, por su paciencia y su acertada manera de guiarnos en la ejecución de este proyecto.

Andrés

ÍNDICE DE CONTENIDOS

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En el presente trabajo se realizó el estudio, diseño e implementación de un sistema para la contabilización de la cantidad de objetos que circulan por el espacio captado por una cámara utilizando librerías de OpenCV. El proyecto consiste en la elaboración de un sistema que presenta, de manera gráfica y numérica al usuario, la cantidad de objetos que transitan, los mismos se podrán visualizar en la pantalla del computador por medio de una aplicación que muestra en tiempo real la imagen de video que está siendo procesada así también como la cantidad de objetos que están siendo sometidos a conteo, a más de esto, la aplicación desarrollada da la potestad al usuario de utilizar diferentes herramientas entre las que se puede recalcar la posibilidad de seleccionar la fuente de video para el procesamiento, ya sea un archivo de video, las imágenes captadas por la propia cámara del computador o un dispositivo externo, también permite elegir la dirección en la que se contarán los objetos (sentido de conteo), además de poder visualizar las diferentes etapas de procesamiento que se está realizando a cada fotograma. La aplicación contará con una consola de control para pausar o continuar el video, a más de esto se muestra el resultado y sentido de la contabilización, el tiempo transcurrido en la misma, la fecha y el promedio de los elementos sometidos a conteo. La interfaz de usuario del sistema se encuentra desarrollada en el software Visual Studio C++, que para nuestro proyecto actúa como el compilador, y con el uso de las librerías de la biblioteca OpenCV se puede procesar en tiempo real los fotogramas de video.

PALABRAS CLAVE:

- **PROCESAMIENTO DE IMÀGENES**
- **LIBRERIA OPENCV**
- **ELECTRÓNICA**

ABSRACT

In the present paper the study, design and implementation of a system for counting the number of objects that moves around the space captured for a camera using OpenCV library was performed. The project involves the development of a system that shows, in graphical and numerical way, the amount of objects that moves, the objects will be displayed on a computer screen through an application that shows, in real time, the frames of video that are being processed and at the same time shows the number of objects that are being counted, to more of this, the developed application allows to user to use some tools such as select the video source for example a video file, the photograms captured for internal camera of computer or use an external device, the application also allows to set the direction for counting objects and can to view the different stages of processing that is being performed at each frame. The application has a control console where we can pause or play the video, to more of this it displays the direction and the results of counting, the time elapsed, date and the average of objects counted. The user interface has been developed in Visual Studio C++, that for our project is the compiler, and with the use of the algorithms of OpenCV library it can process video frames in real time.

KEYWORDS:

- **IMAGE PROCESSING**
- **OPENCV LIBRARY**
- **ELECTRONIC**

INTRODUCCIÓN

El presente trabajo se encuentra distribuido en cuatro capítulos, en los cuales se detalla cada una de las actividades que se realizaron y los temas que involucra cada etapa de desarrollo del proyecto, a continuación, se especifica cómo se encuentra estructurado cada uno de estos.

CAPÌTULO I

En el primer capítulo se presenta el desarrollo del marco teórico, en este se incluye toda la información respecto al diseño del sistema, las herramientas empleadas tanto en hardware como en software para la creación del mismo, y por último el tratamiento por el cual pasan las imágenes al ser sometidas a un procesamiento.

CAPÍTULO II

El segundo capítulo abarca las consideraciones de diseño del sistema, el procedimiento que se tomó al desarrollar el mismo, así como también la implementación, en este se incluye la sintaxis de programación más importante que tuvo relevancia en el tratamiento de las imágenes.

CAPÍTULO III

Se realiza la implementación del sistema, además de las pruebas de funcionamiento de la aplicación bajo diferentes parámetros de evaluación, posterior a esto se muestra en tablas el análisis comparativo respecto al desempeño del sistema.

CAPÍTULO IV

En éste último capítulo se presenta las conclusiones y recomendaciones a las cuales se llegó tras la finalización del diseño y la implementación del sistema.

CAPÍTULO I

GENERALIDADES

1.1.Introducción

En el presente capítulo se analizan varios temas de relevancia en el desarrollo del sistema a realizarse entre los que podemos mencionar la visión artificial conjuntamente con las técnicas que involucran la misma, describiendo una por una la función que cumplen dentro del proceso.

Para la adquisición de las imágenes se emplea un dispositivo óptico con características adecuadas para la captación de fotogramas ajustados al proceso. Dentro del desarrollo del proyecto se encuentra el procesamiento digital de imágenes sobre las cuales se va a trabajar, dando un formato adecuado para la extracción de información, siguiendo un ciclo de etapas las cuales cumplen con tareas específicas.

Las imágenes provistas por el dispositivo óptico son procesadas con librerías de la biblioteca de OpenCV porque presenta un sinnúmero de funciones, métodos, algoritmos, etc., congregados en grupos, formando así su estructura, con la gran ventaja de ser de código abierto y permitiendo su depuración en cualquier compilador C/C++.

Para la implementación de la interfaz de usuario se utilizó el compilador Visual Studio puesto que es el depurador C/C++ oficial de Microsoft y ofrece un entorno de desarrollo amigable al usuario al momento de diseñar HMIs.

1.2.Visión artificial

La visión artificial o visión por computador es un campo de la Inteligencia Artificial que permite a las máquinas ver, extraer información de las imágenes digitales, resolver alguna tarea o entender la escena que están visionando (Maduell, 2010). El procedimiento comúnmente empleado para obtener información de las imágenes digitales se muestra en la figura 1.

Figura 1: Procedimiento para obtener información de imágenes digitales

El primer paso es la adquisición de imágenes del mundo real mediante un dispositivo con un sensor óptico y un sistema que convierta la señal del sensor en información digital para que pueda ser procesada por un computador. El segundo paso consiste en procesar secuencialmente las imágenes digitales. En visión por computador se distinguen dos niveles de procesamiento: procesamiento de imágenes de bajo nivel y de alto nivel (Martínez, 2004).

Los métodos de bajo nivel utilizan muy poco conocimiento del contenido de la imagen; algunos ejemplos son binarización de imágenes, filtros para eliminar el ruido, extracción de bordes, segmentación, etc. El procesamiento de alto nivel está basado en el conocimiento y es común utilizar algoritmos de Inteligencia Artificial (Martínez, 2004).

Figura 2: Niveles de procesamiento en visión artificial

El tercer y último paso consiste en extraer solo la información necesaria de la imagen procesada acorde a la aplicación que se desee llevar a cabo, por ejemplo, en base a la posición de los objetos en movimiento poder contabilizar el número personas que ingresa o sale en un sitio determinado.

1.2.1.Cámara Digital

La cámara digital es un dispositivo electrónico que tiene por objetivo capturar imágenes y transformarlas a un formato entendible por los computadores. En lo que se refiere a visión artificial, la información provista por la cámara, es la base sustentable para el procesamiento digital de imágenes, es por esta razón que la cámara digital es el elemento principal en visión por computador, prácticamente se puede decir que son los ojos de todo el sistema.

Las cámaras web y la mayoría de cámaras de vídeo se pueden conectar directamente a los ordenadores por medio de sus periféricos de entrada y podemos capturar fotogramas en tiempo real (Maduell, 2010).

1.2.2.Cámara FACECAM 321

Para la ejecución de este proyecto se empleará una cámara web denominada VGA Plug&Play Facecam 321 la misma que brinda la facilidad de adquirir imágenes en perfecto estado e idóneas para el desarrollo de este propósito. La Facecam 321 es una cámara web automática y sencilla de emplear ya que no necesita de la instalación de algún driver para que pueda funcionar de manera correcta y eficaz en los distintos sistemas operativos que se empleen para el desarrollo de alguna aplicación (Genius, 2011).

Tabla 1

Especificaciones técnicas de la cámara FaceCam321

Especificaciones Técnicas	
Sensor de Imagen	Lente VGA píxel CMOS
Tipo de Lente	Foco Manual
Definición	8Mega Píxeles (MP)
Interfaz	USB 2.0
Formato de Archivo	MIJPEG / WMV
Resolución	640 x 480 píxeles
Resolución de video	VGA: 30fps
UVC (Plug & Play)	Si

Figura 3: FaceCam 321
Fuente: (Genius, 2011)

1.3.Procesamiento digital de imágenes

El procesamiento de imágenes hace referencia a los algoritmos aplicados al contenido digital de una imagen para modificarla, eliminar datos innecesarios, filtrarla y más, con el fin de obtener información relevante acorde a la aplicación que se le desee dar.

1.3.1.Formación de Imágenes

Una imagen, también llamado fotograma, puede ser representada como una matriz bidimensional o un arreglo de matrices bidimensionales de la forma:

$$I(x, y)$$

El valor del elemento $I(x, y)$ representa la intensidad de color en la posición (x, y) del plano del fotograma.

Las imágenes a escala de grises serán representadas por una sola matriz bidimensional; mientras que los fotogramas a color serán representados

por un arreglo de tres matrices bidimensionales donde cada una representa los valores de intensidad de los colores primarios (rojo, verde y azul). Las dimensiones de la matriz que conforman la imagen dependerán de la resolución de la misma y cada elemento de la matriz es denominado pixel. En la siguiente imagen se muestra un ejemplo de la representación matricial de una imagen a escala de grises.

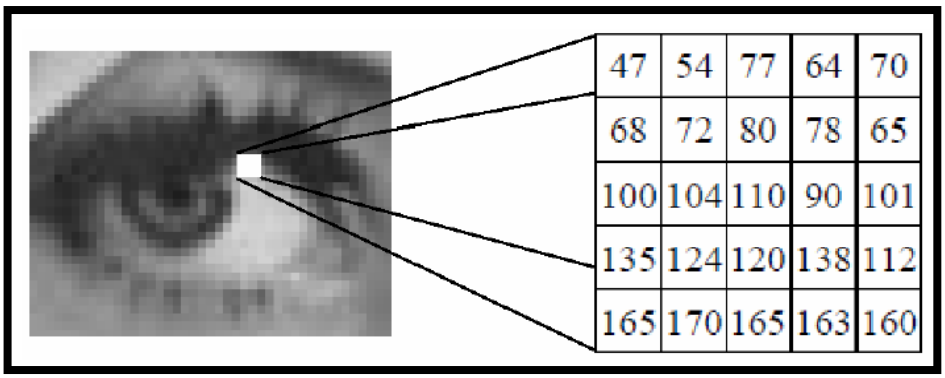

Figura 4: Representación matricial de una imagen a escala de grises
Fuente: (Justo & Aguirre, 2009)

1.3.2.Video

Un video es una secuencia de fotogramas tomados en intervalos de tiempo constante, su representación matemática sería de la forma:

$$I(x, y)(n)$$

El valor del elemento $I(x, y)$ corresponde a la intensidad de color del pixel en la posición (x, y) del plano correspondiente al fotograma n.

El intervalo de tiempo en el cual se puede reproducir un fotograma es una característica innata de todos los archivos de video y su valor se denomina *frames por segundo (fps)* siendo 60 fps el valor más empleando.

1.3.3.Etapas del Procesamiento Digital de Imágenes

Las etapas por las que debe pasar la información de un fotograma en procesamiento de imágenes se describen en la figura 5.

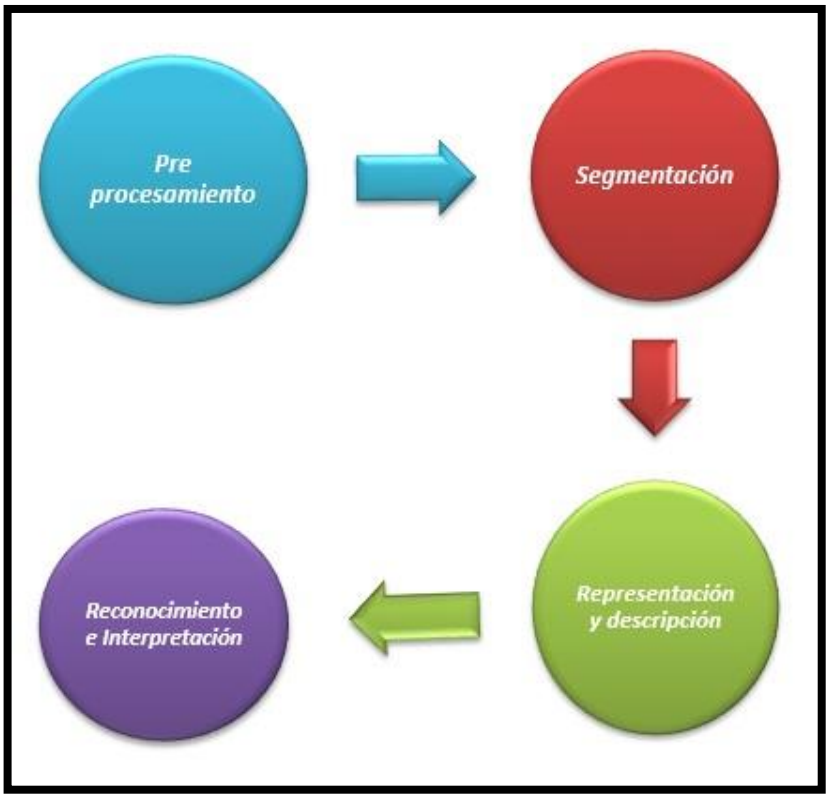

Figura 5: Etapas del procesamiento digital de imágenes

1.3.3.1.Pre procesamiento

El objetivo de esta etapa es mejorar la calidad de información de la imagen para aumentar la probabilidad de éxito en los siguientes pasos. Algunos ejemplos de métodos utilizados en esta etapa son aumentar el contraste, remover el ruido, etc. (Martínez, 2004).

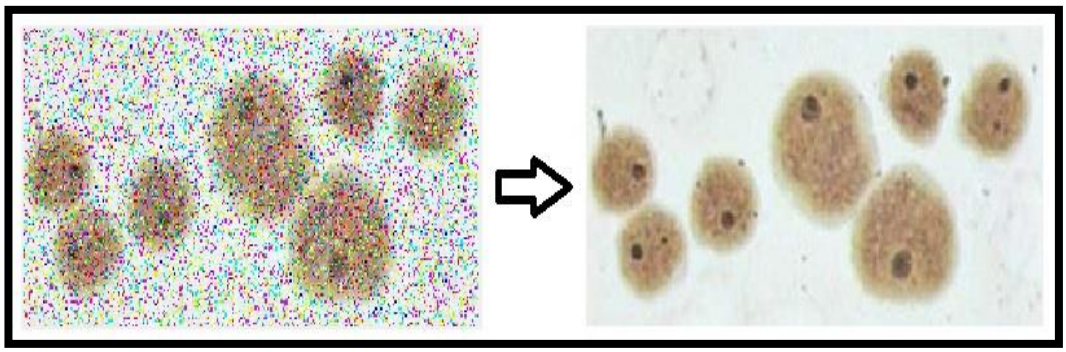

Figura 6: Pre procesamiento de imágenes

1.3.3.2.Segmentación

La segmentación de imágenes es la etapa del procesamiento que consiste en descartar la información innecesaria de una imagen para obtener solo las regiones con cierta homogeneidad respecto a una propiedad como puede ser el color, textura, movimiento, entre otros.

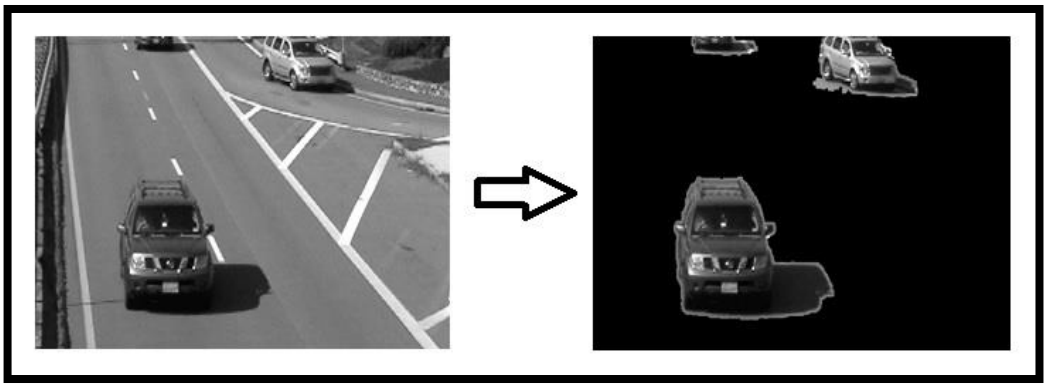

Figura 7: Segmentación

En general, la segmentación es una de las tareas más difíciles en el procesamiento de imágenes y de esta etapa depende el eventual éxito o fracaso del análisis de la imagen. En lo que se refiere a algoritmos implementados para la segmentación existen varias metodologías, entre las más comunes están: técnicas basadas en detección de frontera,

técnicas de umbral y técnicas basadas en agrupamiento de píxeles (Martínez, 2004).

En las técnicas de segmentación por detección de frontera o bordes, la segmentación de una imagen se da mediante algoritmos basados, principalmente, en la derivada de la intensidad de pixeles adyacentes. Bajo estas circunstancias los pixeles segmentados serán los que presenten una alta variación de intensidad, es por esto que al aplicar esta técnica se obtendrán los contornos de una imagen puesto que son las regiones que presentan mayor variación de intensidad entre pixeles vecinos.

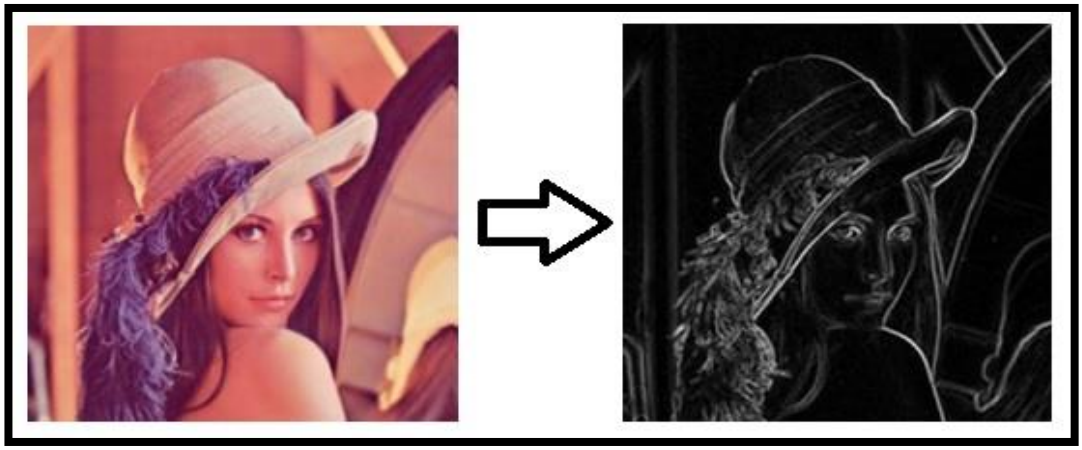

Figura 8: Segmentación por detección de bordes
Fuente: (Bradski & Kaehler, 2008)

Las técnicas de umbral segmentan la imagen pixel por pixel, es decir verifican si el pixel está dentro de un rango específico (threshold) para que pueda ser segmentado de otro modo su información es eliminada.

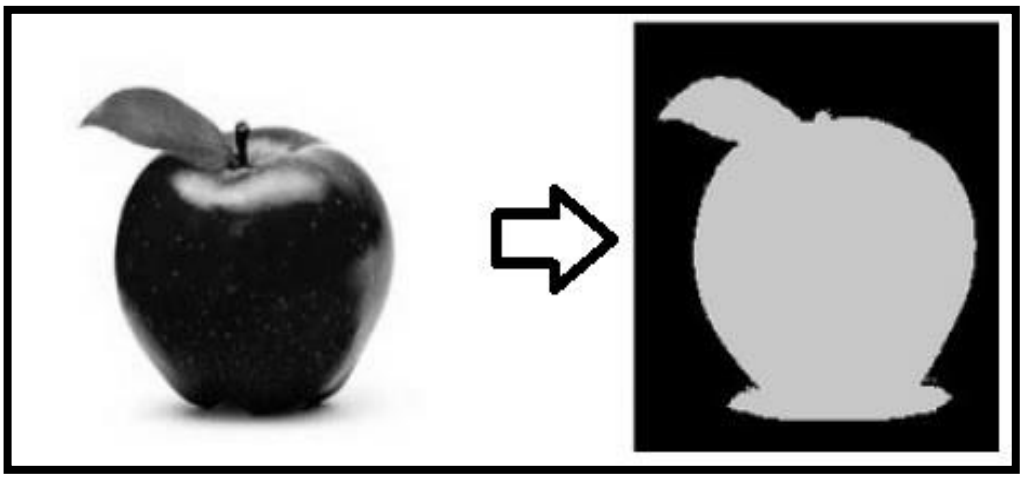

Figura 9: Segmentación por umbralización
Fuente: (Bradski & Kaehler, 2008)

Por otra parte, en los métodos basados en agrupamiento de píxeles o método de regiones la imagen es dividida en regiones agrupando pixeles vecinos con niveles de intensidad similares (Martínez, 2004).

1.3.3.3.Representación y descripción

La descripción, también llamada selección de características, extrae información cuantitativa o características relevantes para diferenciar un objeto de los demás (Martínez, 2004).

Esta etapa permite hacer un filtrado o una clasificación rigurosa de los objetos obtenidos en la segmentación como paso previo al reconocimiento e interpretación de los mismos. Es en este punto donde se deben discriminar las regiones que no concuerdan con el objetivo del sistema, para nuestro caso se realizó un filtrado en base al área de los objetos segmentados y así podemos eliminar el análisis de regiones con un tamaño muy reducido o muy elevado las cuales pueden aparecer por efectos de ruido en la imagen.

1.3.3.4.Reconocimiento e interpretación

La última etapa del procesamiento de imágenes es el reconocimiento e interpretación. El reconocimiento es el proceso que asigna un identificador a un objeto basado en la información provista por sus descriptores. La interpretación es asignar un significado al objeto reconocido. En general, los sistemas de procesamiento que incluyen reconocimiento e interpretación están asociados a aplicaciones de análisis de imágenes cuyo objetivo es la extracción automática de información (Martínez, 2004).

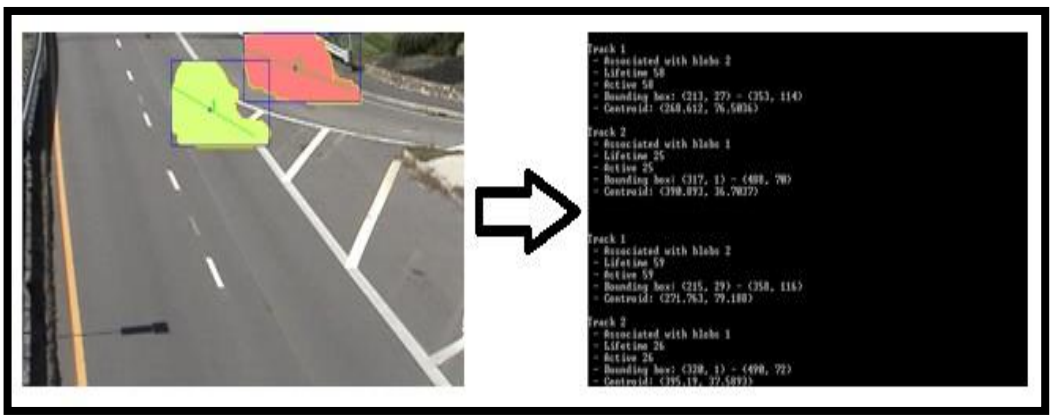

Figura 10: Reconocimiento e interpretación de objetos en una imagen

1.4.OpenCV

OpenCV (Open Source Computer Vision library, *Librerías de Código Abierto para Visión por Computador*) es un conjunto de librerías escritas en C, C++, Python y Java de código libre orientada a visión por computador. Tienen la característica de ser multiplataforma lo cual permite su ejecución en diferentes sistemas operativos como Windows, Linux, MacOS, iOS y Android (Itseez, 2014).

OpenCV es, principalmente, una biblioteca de alto nivel que implementa algoritmos para: procesamiento de imágenes de bajo nivel (filtrado,

morfología, transformaciones geométricas, histogramas), procesamiento de imágenes de alto nivel (segmentación, restauración de imágenes), procesamiento de contornos y geometría computacional, seguimiento de objetos, flujo óptico, calibración de cámaras, entre otros (Itseez, 2015).

La principal característica de la librería, además de su funcionalidad, es el desempeño. Los algoritmos están basados en estructuras de datos dinámicos y más de la mitad de las funciones fueron optimizadas a nivel de ensamblador para tomar ventaja de la arquitectura de los procesadores Intel (Collazos, 2009).

1.4.1.Características generales de OpenCV

- Posee infraestructura para aplicaciones de visión artificial.
- Contiene herramientas para la interpretación de imágenes.
- Posee una licencia BSD lo cual permite utilizar y modificar el código.
- Librerías que funcionan para C/C++, por ser una biblioteca completa en funciones para realizar operaciones con imágenes.
- Permite efectuar el estudio y seguimiento de objetos en movimiento.
- Utilizada en aplicaciones como la detección de intrusos en videos, monitorización de equipamientos, ayuda a navegación de robots, etc.
- Permite la ejecución de código en diferentes sistemas operativos.

1.4.2.Estructura de OpenCV

Esta biblioteca se divide en nueve módulos, en donde se encuentran clasificadas las librerías dependiendo de su utilidad:

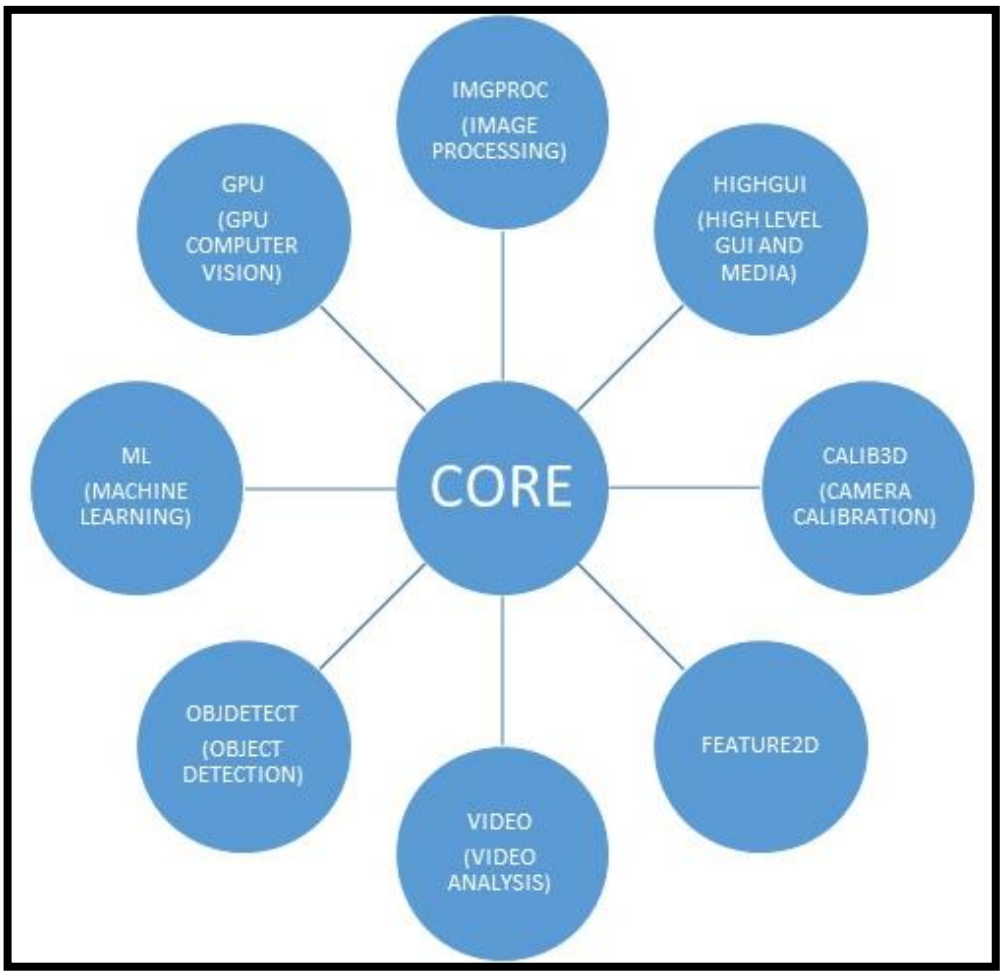

Figura 11: Estructura de la biblioteca OpenCV

A continuación, explicaremos los modulos de OpenCV en los cuales se encuentran las librerías implementadas en nuestro sistema:

1.4.2.1.Core

En este grupo se encuentran las clases, estructuras y algoritmos básicos que usan las demás funciones. La clase *Mat* es la más usada puesto que sirve para crear objetos de tipo matriz en los cuales se puede almacenar la información de un fotograma, campos vectoriales, histogramas, entre otros, además la clase Mat permite realizar operaciones matriciales.

1.4.2.2.Imgproc

De las siglas Image Processing, en este grupo están implementadas las funciones principales para procesamiento de imágenes y algoritmos de visión. Las funciones más relevantes de este módulo en nuestro proyecto se presentan a continuación:

- blur(): filtrado de imágenes
- erode(), dilate(): erosión y dilatación de imágenes
- threshold(): binarización de imágenes
- findContours(): detección de contornos de una imagen

1.4.2.3.Highgui

En este módulo se encuentran las funciones con todo lo relacionado a la sencilla interfaz gráfica de OpenCV y las funciones que permitan importar imágenes y video. Los algoritmos de este grupo que más incidencia tuvieron en nuestro sistema son:

- VideoCapture: un objeto de esta clase sirve para la captura de vídeo a partir de archivos de vídeo, secuencias de imágenes o cámaras.
- namedWindow(): permite crear una ventana para la visualización de fotogramas.
- imshow(): permite visualizar un objeto tipo Mat en una ventana.

1.4.2.4.Video

Las librerías de este módulo permiten realizar análisis de videos entre las más destacadas tenemos la sustracción de fondo, extracción de movimiento, seguimiento de objetos, entre otros.

Para nuestro proyecto hicimos uso de la clase *BackgroundSubstractorMOG2* que permite sustraer el fondo de un video mediante un modelo de segmentación denominado Mixture of Gauss el cual consiste en la actualización del fondo del video usando combinaciones de distribuciones de Gauss.

1.5.Visual Studio

Microsoft Visual Studio es un conjunto de herramientas para crear software, desde la fase de diseño pasando por las fases de diseño de la interfaz de usuario, codificación, pruebas, depuración, análisis de la calidad y el rendimiento del código, implementación en los clientes y recopilación de telemetría de uso. Estas herramientas están diseñadas para trabajar juntas de la forma más eficiente posible y todas se exponen a través del Entorno de desarrollo integrado (IDE) de Visual Studio (Microsoft Developer Network, 2012).

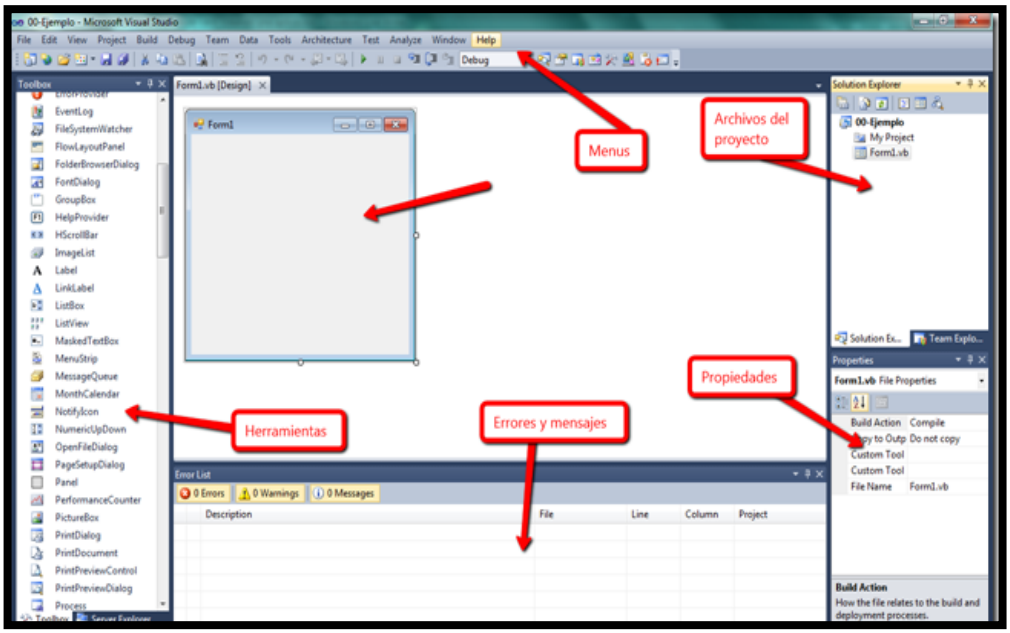

Figura 12: Microsoft Visual C++ 2010 Express Edition

Fuente: (Tony Valderrama, Visual BASIC. net)

Entre las diferentes secciones que componen la ventana principal de trabajo de Visual Studio están las siguientes:

- Superficie de diseño de Visual Studio Extensions for Windows Workflow Foundation: Se emplea para establecer un flujo de trabajo arrastrando actividades de la ventana de herramientas hacia el área de diseño.
- Cuadro de herramientas: Facilita al usuario configurar el entorno de trabajo a gusto del mismo, en el cual se puede ubicar botones, cajas de texto, etc., mediante el uso de los controles ubicados en este cuadro.
- Propiedades: Se utiliza para configurar las propiedades del objeto que se ha seleccionado, por ejemplo: color, tamaño, nombre, etc.
- Explorador de soluciones: Muestra los archivos que componen el proyecto en forma jerárquica y facilita acceso a las características que ayudan a administrar los proyectos.
- Ventanas de depuración: En esta se presentan los mensajes de información de la depuración del programa a la hora de compilarlo (Microsoft, 2015).

De forma predeterminada, Visual Studio proporciona compatibilidad con C y C++, lo cual lo convierte en el compilador ideal para trabajar con la biblioteca OpenCV en la plataforma Windows.

1.5.1.Características

- Visual C++ hace uso extensivo del framework Microsoft Foundation Classes (o simplemente MFC), el cual es un conjunto de clases C++ para el desarrollo de aplicaciones en Windows.
- El IDE cuenta con herramientas como el IntelliSense, RemoteDebuging, Editar y Continuar, y Texto Resaltado.
- Herramientas de codificación eficaces que permite escribir código, compilar y corregir problemas.

- El lenguaje de programación está basado en C++ y es compatible en la mayor parte de su código con este lenguaje, de igual forma que su sintaxis es exactamente igual.
- Depuración avanzada que permite diagnosticar problemas sin abandonar el flujo de trabajo.
- Visual C++ es un entorno integrado de desarrollo que permite la programación orientada a objetos (POO) conjuntamente con el sistema de desarrollo SDK (también denominado API) de Windows.
- Visual C++ incluye la librería de clases MFC (Microsoft Foundation Classes) que permite crear y gestionar de manera intuitiva componentes típicos de Windows.

Al ser un entorno integrado Visual C++ incluye, entre otras, las siguientes herramientas de desarrollo:

- Editor de texto
- Compilador/Enlazador
- Depurador
- Visor de datos y dependencias (Browser). Visual Studio Community. (2015).

CAPÍTULO II

DISEÑO E IMPLEMENTACIÓN

2.1.Introducción

En el presente capítulo se analizan 3 temas importantes sobre los cuales está basado el diseño y la implementación del sistema, dentro de las consideraciones de diseño se describen los elementos tanto de hardware como de software que se han empleado para la elaboración del proyecto así como también la importancia de la utilización de los mismos, posterior a esto, en lo que se refiere al diagrama de flujo del sistema, se presenta el proceso que se sigue para la elaboración del plan, tomando en cuenta cada uno de los pasos que conforman este punto así como cuál es el procedimiento que se realiza en cada uno de ellos.

Como último se incluye el diseño de la interfaz del usuario del sistema, en el cual se presenta el panel interactivo que se mostrará al usuario, donde el mismo tiene la facilidad de conocer como está actuando el proceso y la contabilización de los objetos que se desee.

2.2.Consideraciones de diseño

2.2.1.Elementos de Hardware

2.2.1.1.Cámara

El sistema basado en visión artificial requiere de un instrumento que le permita capturar imágenes, las cuales son la información a manipularse por

lo que se emplea un dispositivo fácil y sencillo de utilizarse con características óptimas para la adquisición de fotogramas sin requerir de que tenga una alta resolución que involucre costos elevados y con características de plug and play, que significa que no tiene la necesidad de realizar una instalación de software para la utilización del dispositivo, a más de esto se requiere de una cantidad de fotogramas por segundo para obtener un video adecuado para el tratamiento de la información.

2.2.1.2.Computador

El computador es el elemento principal dentro del sistema ya que en conjunto con el software empleado dentro del mismo cumple con la tarea de procesar la información que es adquirida mediante el uso de una cámara y convertida en datos a ser procesados, por lo que la capacidad de procesamiento debe ser de alta velocidad, ya que el sistema ejecuta la contabilización de los elementos en tiempo real. En forma adicional posee una memoria RAM de 8GB que es muy importante debido a que el programa requiere de una cantidad mínima de 80MB para cumplir con la tarea de manera eficiente, con lo que se brinda al usuario una alta capacidad de respuesta. El sistema operativo utilizado es Windows 10 de 64 bits, el mismo que reúne las características necesarias para cumplir con el objetivo del diseño y la creación del sistema.

2.2.2.Elementos de Software

2.2.2.1.Visual Studio

Es un entorno de desarrollo creado para trabajar en Windows y soporta varios lenguajes de programación, por lo que para la elaboración del sistema se maneja C++, el mismo permite realizar la compilación de la

información que se está manipulando, permite el trabajo con objetos dinámicos lo cual es una característica muy importante debido al tratamiento que se está dando con las librerías de OpenCV; se desarrolla bajo el paradigma C++ debido a que resulta fácil el acoplamiento con OpenCV, a más de que permite el diseño de la interfaz entre el usuario y el sistema para que este pueda manipular y conocer el desarrollo y funcionamiento del mismo, conociendo de manera gráfica y numérica la cantidad de objetos que se está contabilizando en tiempo real.

2.2.2.2.OpenCV

La biblioteca de OpenCV, como su nombre lo indica posee librerías creadas para aplicaciones de machine learning y visión artificial, este último es la cualidad que se tomó en cuenta para usarla en el desarrollo del sistema, el cual está basado en visión artificial. Los algoritmos que posee son los adecuados para el tratamiento que se realiza con la información como por ejemplo la identificación de objetos, clasificación y acciones humanas en videos, encontrar la similitud en imágenes, etc. y a más de una característica importante en el procesamiento de imágenes como es el tracking y otros algoritmos que más adelante se analizarán con mayor detalle, su código de programación está escrito en C++ y C# lo que le permite operar de manera eficiente con Visual Studio.

2.2.2.3.Librería cvblob

La librería cvblob es empleada en el desarrollo de proyectos para visión por computadora, con la cualidad de que permite detectar las regiones conectadas en imágenes procesadas digitalmente y en formato binario, está escrita en C y C++ lo cual permite que se involucren fácilmente con

Visual Studio y OpenCV, esta librería posee características similares que la hacen compatible con esta última.

2.3.Diagrama de flujo del sistema

A continuación, se muestra en un diagrama de flujo los pasos a seguir para implementar un sistema que realiza la contabilización de objetos en movimiento utilizando técnicas de visión artificial.

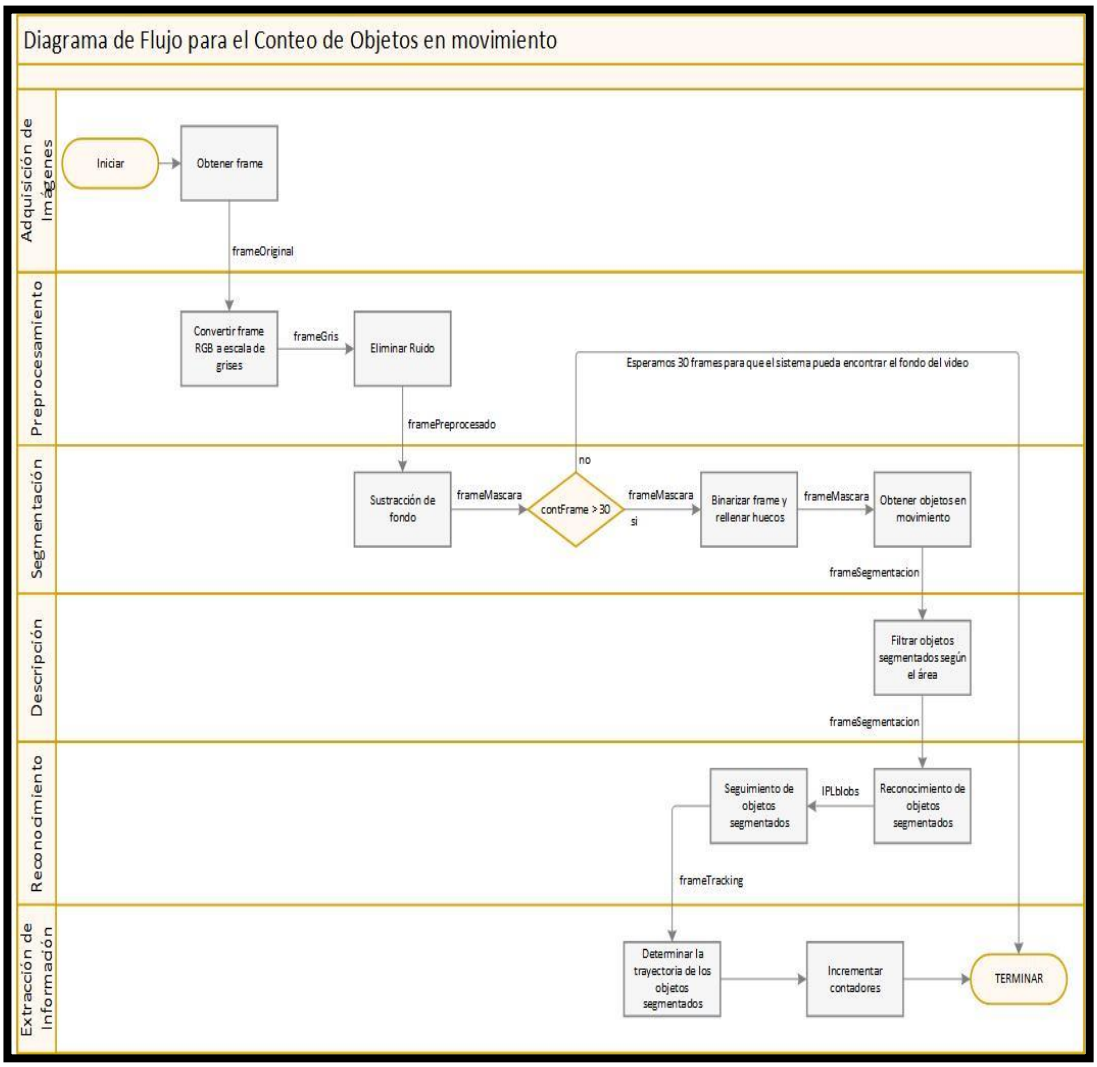

Figura 13: Diagrama de flujo del sistema desarrollado para la contabilización de objetos

Para comprender de mejor forma como se encuentra estructurado el diagrama de flujo mostrado en la figura 13, se detalla en los siguientes puntos cada una de las etapas descritas en el mismo.

2.3.1. Adquisición de Imágenes

Para obtener imágenes, ya sea de un archivo, video o una cámara de video, la librería de OpenCV tiene implementado la clase VideoCapture, la misma que posee métodos para obtener frames de un archivo de video, de la webcam del computador o a su vez de una cámara externa. La secuencia a seguir para obtener los fotogramas del video es la siguiente:

- Crear un objeto tipo VideoCapture con la dirección del archivo de video o el número de cámara conectado al computador.
- Configurar los parámetros del video como son las dimensiones del fotograma y el número de tramas por segundo que queremos recibir.
- Almacenar la información del fotograma en un objeto tipo Mat.
- Eliminar el objeto VideoCapture al finalizar el video para liberar la memoria.

Un ejemplo sobre la forma de obtener fotogramas es la siguiente:

```
Mat frame;
VideoCapture video(0);
video.set(CV_CAP_PROP_FPS, 60.0);
…… lazo ……
video.read(frame);
…… fin lazo ……
video.release();
```

2.3.2.Pre procesamiento

Para optimizar el procesamiento de imágenes en necesario convertir las imágenes con formato RGB (red, green, blue) a escala de grises puesto que se trabaja con una sola matriz y de esta forma reducir el tiempo de procesamiento, por lo que para esto existe la función cvtColor. Una aplicación se indica a continuación:

cvtColor(frameRGB, frameGris, CV_RGB2GRAY);

Donde el frameRGB corresponde a la matriz de valores de la imagen original sin tener ningún tipo de tratamiento, el frameGris corresponde a los valores de la imagen en escala de grises.

Si se desea asegurar que el tratamiento de la información sobre la cual se está trabajando sea adecuada para las etapas posteriores, es necesario eliminar el ruido que se presenta en las imágenes, en este caso OpenCV posee una variedad de filtros digitales donde su uso dependerá de la aplicación en la cual se le emplee.

Un ejemplo en el que se indica cómo se utiliza el filtro blur se muestra a continuación:

blur(frameGris, framePreprocesado, cv::Size(3, 3));

Donde el frameGris corresponde a la matriz de valores de la imagen en escala de grises, el framePreprocesado corresponde a la matriz de valores de la imagen que ya ha sido sometida al proceso del filtro.

2.3.3.Segmentación

OpenCV posee diversas técnicas para realizar la segmentación de objetos de una imagen, por lo que la implementación de cada uno de ellos

dependerá del tipo de objetos que se desee segmentar. Debido a las características del proyecto que se desarrolla en el presente trabajo, la sustracción de fondo es el algoritmo principal para poder analizar los objetos en movimiento.

2.3.3.1.Sustracción de Fondo

Los algoritmos para sustracción de fondo de un video permiten hallar una matriz de puntos que servirá de modelo de la escena llamada máscara de fondo. Procesando la desviación de cada pixel de los fotogramas entrantes con la máscara de fondo se obtienen los objetos en movimiento.

El proceso básico de un algoritmo de sustracción de fondo consiste en una resta absoluta entre la trama actual del video y la máscara de fondo, seguidamente se aplica un umbral a toda la matriz resultante para darle un formato binario a la imagen, de esta forma las regiones de color blanco corresponden a los objetos en movimiento y el resto de la imagen tendrá una tonalidad oscura, por lo que la imagen resultante es conocida como máscara de movimiento.

$$mascaraMovimiento = |frameActual - mascaraFondo| \quad (1)$$

$$mascaraMovimiento(i,j) = \begin{cases} 1, & mascaraMovimento(i,j) > umbral \\ 0, & mascaraMovimento(i,j) \leq umbral \end{cases}$$

El algoritmo de sustracción de fondo tiene un procedimiento básico puesto que es vulnerable a errores específicamente cuando el fondo de la escena varía, por ejemplo, las variaciones de luminosidad que se presentan en el transcurso del día. Existen técnicas de sustracción de fondo más robustas para compensar estos errores, la más común es Gaussian Mixture Background Segmentation la cual implementa algoritmos adaptativos

basados en desviaciones estándar para actualizar la máscara de fondo continuamente en cada trama.

OpenCV posee la clase BackgroundSubtractorMOG2 para segmentar objetos en movimiento, para lo cual se debe crear una instancia de clase y llamar al método operator() el cual recibe el frame actual y entrega la máscara de movimiento. En el siguiente ejemplo se presenta la sustracción de fondo de un video, donde frameActual representa el fotograma actual de video y mascaraMovimiento es el resultado de la sustracción de fondo como se aprecia en la figura 14.

```
BackgroundSubtractorMOG2 bs;
bs.operator(frameActual, mascaraMovimiento, -1);
```

Figura 14: Sustracción de Fondo del video procesado

Para completar el proceso de segmentación existe la etapa de rellenado de huecos, ya que como se puede apreciar en la figura 14, en la sustracción de fondo, las regiones de los objetos en movimiento poseen puntos que no fueron segmentados de manera correcta.

Para corregir estos errores se debe aplicar técnicas basadas en detección de fronteras para encontrar los bordes de las regiones segmentadas y posteriormente rellenar el interior. Las librerías de OpenCV proveen de las funciones necesarias para cumplir con la etapa de rellenado de huecos las mismas que son:

- findContours(): realiza la detección de contornos de la imagen de entrada, retornando la matriz de puntos de cada borde encontrado, además de la jerarquía de los mismos.
- drawContours(): dibuja líneas uniendo los puntos de los bordes del contorno.

En la figura 15 se puede observar el resultado del rellenado de huecos tras la sustracción de fondo.

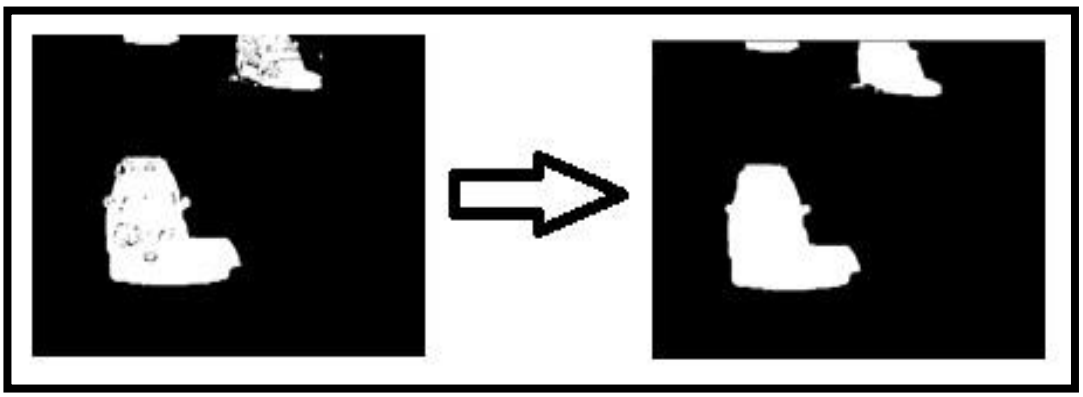

Figura 15: Procedimiento de rellenado de huecos en el tratamiento de las imágenes procesadas

Finalmente, ya con la máscara de movimiento bien definida, el último paso de la segmentación consiste en eliminar el fondo de la imagen para discriminar la información que no se utiliza y quedarnos con los objetos en movimiento para lo cual se realiza una multiplicación binaria entre la máscara de movimiento y el frame original.

En el siguiente ejemplo se implementa una multiplicación binaria entre el fotograma actual de video (frameOriginal) y la máscara de movimiento (mascaraMovimiento) para obtener como resultados los objetos segmentados (frameSegmentacion):

bitwise_and(frameOriginal, mascaraMovimiento, frameSegmentacion);

Figura 16: Segmentación de la imagen procesada

2.3.4.Descripción

En esta etapa se filtrarán los objetos según el área, es decir solo los objetos que posean un área congruente con el área de la trama actual serán considerados en la etapa posterior, gracias a esta etapa se reducirá la probabilidad de errores producido por el ruido en la imagen (objetos con área muy pequeña) o por mala detección de la máscara de fondo (objetos con área muy grande).

El filtrado de objetos según el área puede ser implementada en la misma etapa de rellenado de huecos gracias a la función contourArea().

A continuación, se presenta en ejemplo para obtener el área de los contornos de una imagen:

```
findContours(mascaraMovimiento, contornos, jerarquia,
CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);

for(size_t i=0; i < contornos.size(); ++i){

int area = contourArea(contornos[i]);

}
```

En el vector contornos se guardan los bordes de las regiones encontradas en el fotograma mascaraMovimiento y gracias a la función contourArea se puede extraer el área de dichas regiones.

2.3.5.Reconocimiento

Esta es la etapa previa a la extracción de información y es aquí donde se debe etiquetar y realizar un seguimiento de los objetos segmentados para poder determinar la trayectoria de los mismos. El etiquetado (labeling) consiste en asignar un identificador a cada objeto segmentado y este a su vez permanecerá constante durante toda la aparición de dicho objeto en el video, es decir, el ID del objeto será el mismo en todos los fotogramas que aparezca dicho objeto.

Gracias al etiquetado, el seguimiento (tracking) de objetos se puede implementar, de esta forma se puede conocer en cada fotograma la posición y la etiqueta (ID) de cada objeto segmentado, esto evita que un objeto pueda ser contado varias veces en una misma trama.

La librería *cvblob* contiene algoritmos para realizar el etiquetado y seguimiento de objetos y los mismos implementan algoritmos de la biblioteca OpenCV facilitando su importación al código de programa. A continuación, se muestra un ejemplo de la implementación de dicha librería:

CvBlobs infoObjetos;

CvTracks seguimientoObjetos;

unsigned int resultados = cvLabel(frameSegmentacion, frameEtiquetado, infoObjetos);

cvUpdateTracks(infoObjetos, seguimientoObjetos, 2.0, 200);

La función cvLabel() realiza el etiquetado de los objetos segmentados y almacena el ID, los momentos, los valores máximos y mínimos en X, los valores máximos y mínimos en Y y el centro de masa de cada uno en infoObjetos que es una instancia de la clase *CvBlobs*.

La función cvUpdateTracks(), realiza el seguimiento de los objetos etiquetados en todo el video y vuelve asignar un ID a cada uno, con la

diferencia que el mismo permanecerá siempre constante hasta cuando transcurran 200 frames de inactividad del objeto.

2.3.6.Extracción de Información

Una vez que conocemos la posición y el nombre de cada objeto segmentado en cada fotograma del video, se puede determinar la trayectoria del mismo aplicando una máquina de estados de la siguiente forma:

Como primer paso dividimos el fotograma en tres zonas (zona 1, zona 2 y zona 3) como se observa en la figura 17, y el estado actual de cada objeto corresponderá al valor de la zona en el que se encuentre su centro de masa.

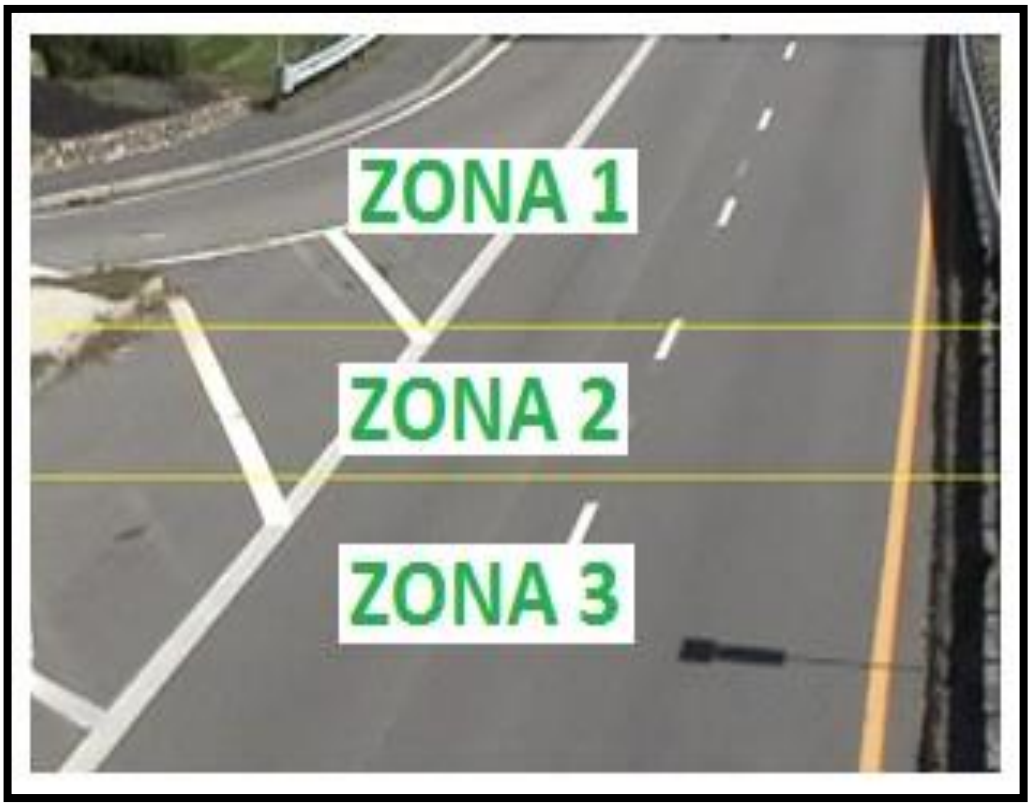

Figura 17: Zonas de división del fotograma.

La máquina de estados analiza el estado actual de cada objeto segmentado y el estado del mismo en el fotograma anterior; de existir un cambio de estado se incrementará los contadores, en la figura 18 se puede observar cuál es el proceso que realiza la máquina de estados para determinar la trayectoria.

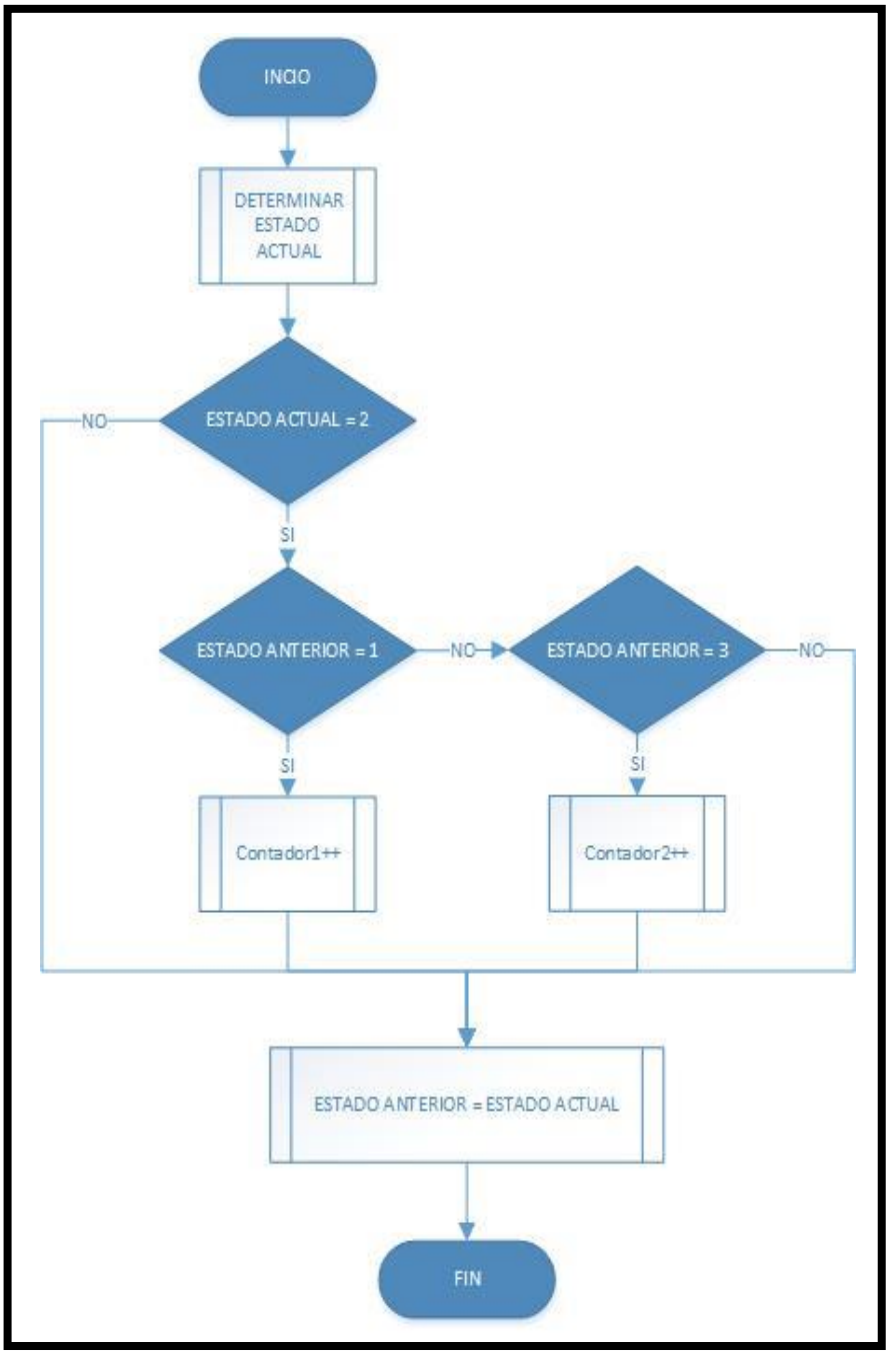

Figura 18: Máquina de Estados para determinar la trayectoria

2.4.Diseño de la interfaz de usuario

La interfaz de usuario del sistema para la contabilización de la cantidad de objetos que circulan por el espacio captado por una cámara se implementó en el software Microsoft Visual Studio C++ 2010 puesto que es el compilador C/C++ oficial de Microsoft.

A continuación se describen los pasos para la creación del proyecto:

2.4.1.Creación de un nuevo proyecto

Para crear un nuevo proyecto debemos debemos seleccionar File/New/Project. Se abre una ventana de Nuevo Proyecto en la cual seleccionamos la plantilla CLR y escogemos un proyecto tipo Windows Forms Application puesto que este tipo de archivos nos permite crear interfaces de usuario.

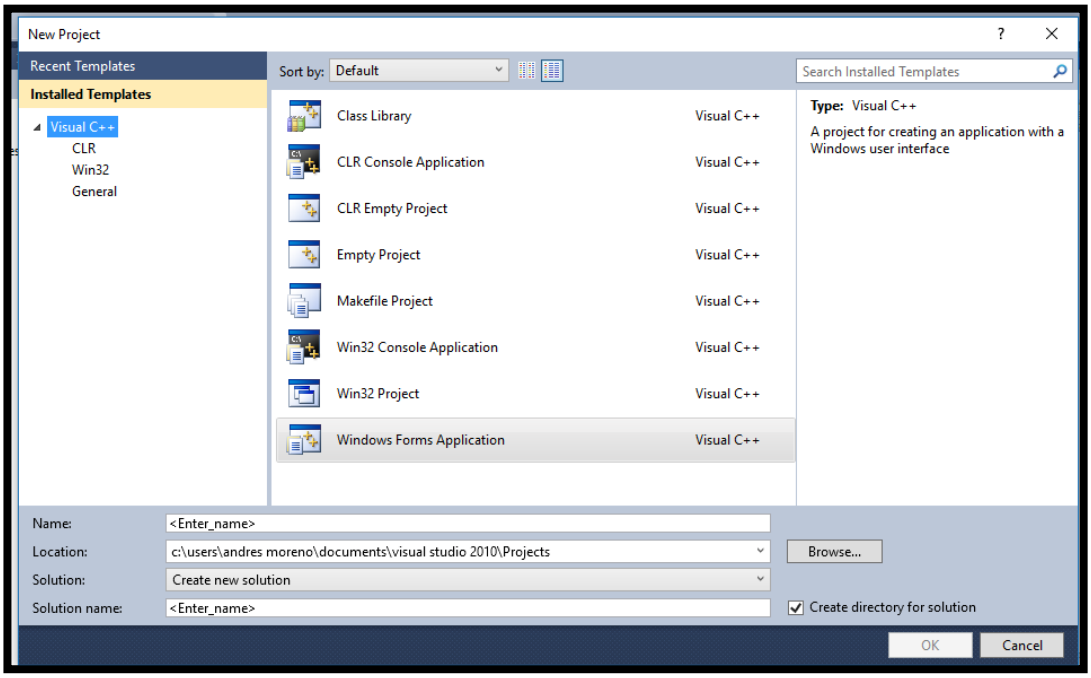

Figura 19: Creación de un proyecto con interfaz de usuario

Se creará un nuevo proyecto con varios archivos, pero el que nos interesa es el archivo de cabecera Forms.h puesto que es el formulario donde se diseñará la interfaz de usuario y se codificará la programación del proyecto. Dicho archivo lo podemos visualizar de dos formas, la primera es gráficamente donde podremos añadir las herramientas de control y monitoreo de la aplicación como son botones, campos de texto, contenedores de imágenes entre otros y la segunda forma de visualización es el archivo en sí mismo, es decir, la programación que se codifica automáticamente al añadir las herramientas de visualización.

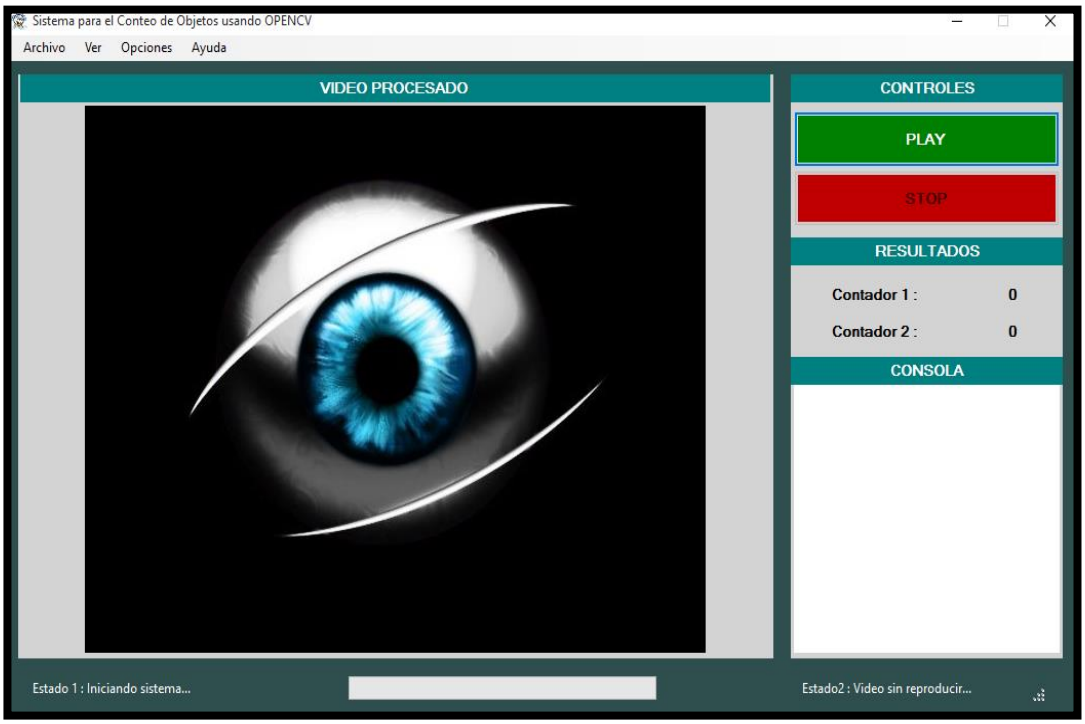

Figura 20: Interfaz de Usuario del Sistema

Como se aprecia en la figura 20 la interfaz de usuario del sistema desarrollado contiene todos los elementos necesarios para su fácil utilización e interpretación de resultados. Para esto la pantalla principal se ha dividido en cuatro partes:

Barra de Herramientas: consta de cuatro menús desplegables donde el usuario puede seleccionar la fuente del video, la etapa del procesamiento

que desee visualizar, el tipo de conteo a realizar, y una breve ayuda del prototipo, se visualiza en la figura 21.

Figura 21: Barra de Herramientas de la Interfaz del Sistema

Barra de Estado: en esta sección el usuario podrá conocer cómo está funcionando el sistema, es decir, si se está procesando algún video, y el estado que se desea reproducir del mismo, como se aprecia en la figura 22.

Figura 22: Barra de Estado del Sistema

Barra de Controles y Resultados: En esta sección se encuentran ubicados los botones que se utilizan para controlar la reproducción del video y posee áreas en una pequeña consola para mostrar los resultados de la contabilización de los objetos, la misma que se observa en la figura 23.

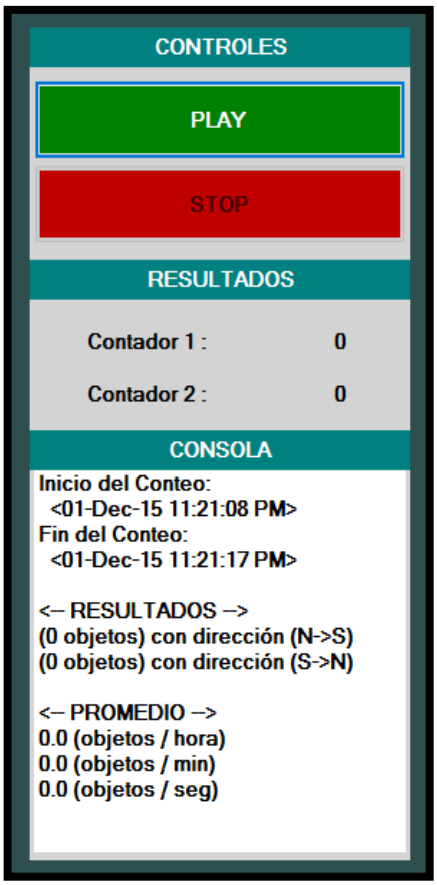

Figura 23: Barra de Controles y Resultados de la Interfaz del Sistema

Espacio de Video: es el lugar asignado para la impresión, en tiempo real, de los fotogramas procesados, se indica en la figura 24.

Figura 24: Área que presenta los fotogramas que se están procesando

A continuación, se muestra la codificación del formulario para obtener esta interfaz de usuario.

```
void InitializeComponent(void) {

System::ComponentModel::ComponentResourceManager^  resources =
(gcnew System::ComponentModel::ComponentResourceManager
(Form1::typeid));

this->item_CamaraExterna = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->status_Estado = (gcnew System::Windows::Forms::StatusStrip());

this->label_Estado1 = (gcnew
System::Windows::Forms::ToolStripStatusLabel());

this->progressBar_Estado = (gcnew
System::Windows::Forms::ToolStripProgressBar());
```

```
this->label_Estado2 = (gcnew
System::Windows::Forms::ToolStripStatusLabel());

this->panel_Principal = (gcnew System::Windows::Forms::Panel());

this->menu_Principal = (gcnew System::Windows::Forms::MenuStrip());

this->item_Archivo = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_CamaraPC = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_Demo1 = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_Demo2 = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_Salir = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_Ver = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->item_Preprocesamiento = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_Segmentacion = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_Tracking = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_VideoProcesado = (gcnew
System::Windows::Forms::ToolStripMenuItem());
```

```
this->item_Opciones = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_ConteoVertical = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_ConteoHorizontal = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_Ayuda = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_VerAyuda = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->item_Acerca = (gcnew
System::Windows::Forms::ToolStripMenuItem());

this->panel_Imagen = (gcnew System::Windows::Forms::Panel());

this->label_Video = (gcnew System::Windows::Forms::Label());

this->picture_Imagen = (gcnew System::Windows::Forms::PictureBox());

this->panel_Botones = (gcnew System::Windows::Forms::Panel());

this->label_Consola = (gcnew System::Windows::Forms::Label());

this->label_C2 = (gcnew System::Windows::Forms::Label());

this->label_VC2 = (gcnew System::Windows::Forms::Label());

this->label_VC1 = (gcnew System::Windows::Forms::Label());

this->label_C1 = (gcnew System::Windows::Forms::Label());

this->button_Stop = (gcnew System::Windows::Forms::Button());
```

```
this->button_Play = (gcnew System::Windows::Forms::Button());

this->label_TituloConsola = (gcnew System::Windows::Forms::Label());

this->label_Resultados = (gcnew System::Windows::Forms::Label());

this->label_Controles = (gcnew System::Windows::Forms::Label());

this->status_Estado->SuspendLayout();

this->panel_Principal->SuspendLayout();

this->menu_Principal->SuspendLayout();

this->panel_Imagen->SuspendLayout();

(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>picture_Imagen))->BeginInit();

this->panel_Botones->SuspendLayout();

this->SuspendLayout();

//

// item_CamaraExterna

//

this->item_CamaraExterna->Name = L"item_CamaraExterna";

this->item_CamaraExterna->Size = System::Drawing::Size(156, 22);

this->item_CamaraExterna->Text = L"Cámara externa";

this->item_CamaraExterna->Click += gcnew System::EventHandler(this,
&Form1::item_CamaraExterna_Click);

//
```

```
// status_Estado

//

this->status_Estado->Anchor =
static_cast<System::Windows::Forms::AnchorStyles>((System::Windows::
Forms::AnchorStyles::Left |
System::Windows::Forms::AnchorStyles::Right));

this->status_Estado->AutoSize = false;

this->status_Estado->BackColor =
System::Drawing::Color::DarkSlateGray;

this->status_Estado->Dock = System::Windows::Forms::DockStyle::None;

this->status_Estado->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(3) {this->label_Estado1,

this->progressBar_Estado, this->label_Estado2});

this->status_Estado->Location = System::Drawing::Point(20, 572);

this->status_Estado->Name = L"status_Estado";

this->status_Estado->Size = System::Drawing::Size(992, 27);

this->status_Estado->TabIndex = 0;

this->status_Estado->Text = L"statusStrip1";

//

// label_Estado1

//

this->label_Estado1->AutoSize = false;
```

```
this->label_Estado1->ForeColor = System::Drawing::Color::White;

this->label_Estado1->Margin = System::Windows::Forms::Padding(0, 3,
10, 2);

this->label_Estado1->Name = L"label_Estado1";

this->label_Estado1->Size = System::Drawing::Size(300, 22);

this->label_Estado1->Text = L"Estado 1 : Iniciando sistema...";

this->label_Estado1->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;

//

// progressBar_Estado

//

this->progressBar_Estado->Name = L"progressBar_Estado";

this->progressBar_Estado->Size = System::Drawing::Size(300, 21);

//

// label_Estado2

//

this->label_Estado2->AutoSize = false;

this->label_Estado2->ForeColor = System::Drawing::Color::White;

this->label_Estado2->Margin = System::Windows::Forms::Padding(10, 3,
0, 2);

this->label_Estado2->Name = L"label_Estado2";
```

```
this->label_Estado2->RightToLeft =
System::Windows::Forms::RightToLeft::No;

this->label_Estado2->Size = System::Drawing::Size(300, 22);

this->label_Estado2->Text = L"Estado2 : Video sin reproducir...";

this->label_Estado2->TextAlign =
System::Drawing::ContentAlignment::MiddleRight;

//

// panel_Principal

//

this->panel_Principal->BackColor =
System::Drawing::Color::DarkSlateGray;

this->panel_Principal->Controls->Add(this->menu_Principal);

this->panel_Principal->Controls->Add(this->panel_Imagen);

this->panel_Principal->Controls->Add(this->panel_Botones);

this->panel_Principal->Controls->Add(this->status_Estado);

this->panel_Principal->Location = System::Drawing::Point(0, -1);

this->panel_Principal->Name = L"panel_Principal";

this->panel_Principal->Size = System::Drawing::Size(1039, 613);

this->panel_Principal->TabIndex = 1;

//

// menu_Principal
```

```
//
this->menu_Principal->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(4) {this->item_Archivo,
this->item_Ver, this->item_Opciones, this->item_Ayuda});
this->menu_Principal->Location = System::Drawing::Point(0, 0);
this->menu_Principal->Name = L"menu_Principal";
this->menu_Principal->Size = System::Drawing::Size(1039, 24);
this->menu_Principal->TabIndex = 1;
//
// item_Archivo
//
this->item_Archivo->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(5) {this->item_CamaraPC,
this->item_CamaraExterna, this->item_Demo1, this->item_Demo2, this-
>item_Salir});
this->item_Archivo->Name = L"item_Archivo";
this->item_Archivo->Size = System::Drawing::Size(60, 20);
this->item_Archivo->Text = L"Archivo";
//
// item_CamaraPC
//
```

```
this->item_CamaraPC->Checked = true;

this->item_CamaraPC->CheckState =
System::Windows::Forms::CheckState::Checked;

this->item_CamaraPC->Name = L"item_CamaraPC";

this->item_CamaraPC->Size = System::Drawing::Size(156, 22);

this->item_CamaraPC->Text = L"Cámara PC";

this->item_CamaraPC->Click += gcnew System::EventHandler(this,
&Form1::item_CamaraPC_Click);

//

// item_Demo1

//

this->item_Demo1->Name = L"item_Demo1";

this->item_Demo1->Size = System::Drawing::Size(156, 22);

this->item_Demo1->Text = L"Demo 1";

this->item_Demo1->Click += gcnew System::EventHandler(this,
&Form1::item_Demo1_Click);

//

// item_Demo2

//

this->item_Demo2->Name = L"item_Demo2";

this->item_Demo2->Size = System::Drawing::Size(156, 22);
```

```
this->item_Demo2->Text = L"Demo 2";

this->item_Demo2->Click += gcnew System::EventHandler(this,
&Form1::item_Demo2_Click);

//

// item_Salir

//

this->item_Salir->Name = L"item_Salir";

this->item_Salir->Size = System::Drawing::Size(156, 22);

this->item_Salir->Text = L"Salir";

//

// item_Ver

//

this->item_Ver->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(4) {this-
>item_Preprocesamiento,

this->item_Segmentacion, this->item_Tracking, this-
>item_VideoProcesado});

this->item_Ver->Name = L"item_Ver";

this->item_Ver->Size = System::Drawing::Size(35, 20);

this->item_Ver->Text = L"Ver";

//
```

```
// item_Preprocesamiento

//

this->item_Preprocesamiento->Name = L"item_Preprocesamiento";

this->item_Preprocesamiento->Size = System::Drawing::Size(173, 22);

this->item_Preprocesamiento->Text = L"Pre procesamiento";

this->item_Preprocesamiento->Click += gcnew
System::EventHandler(this, &Form1::item_Preprocesamiento_Click);

//

// item_Segmentacion

//

this->item_Segmentacion->Name = L"item_Segmentacion";

this->item_Segmentacion->Size = System::Drawing::Size(173, 22);

this->item_Segmentacion->Text = L"Segmentación";

this->item_Segmentacion->Click += gcnew System::EventHandler(this,
&Form1::item_Segmentacion_Click);

//

// item_Tracking

//

this->item_Tracking->Name = L"item_Tracking";

this->item_Tracking->Size = System::Drawing::Size(173, 22);

this->item_Tracking->Text = L"Tracking";
```

```
this->item_Tracking->Click += gcnew System::EventHandler(this,
&Form1::item_Tracking_Click);

//

// item_VideoProcesado

//

this->item_VideoProcesado->Checked = true;

this->item_VideoProcesado->CheckState =
System::Windows::Forms::CheckState::Checked;

this->item_VideoProcesado->Name = L"item_VideoProcesado";

this->item_VideoProcesado->Size = System::Drawing::Size(173, 22);

this->item_VideoProcesado->Text = L"Video Procesado";

this->item_VideoProcesado->Click += gcnew System::EventHandler(this,
&Form1::item_VideoProcesado_Click);

//

// item_Opciones

//

this->item_Opciones->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->item_ConteoVertical,

this->item_ConteoHorizontal});

this->item_Opciones->Name = L"item_Opciones";

this->item_Opciones->Size = System::Drawing::Size(69, 20);
```

```
this->item_Opciones->Text = L"Opciones";

//

// item_ConteoVertical

//

this->item_ConteoVertical->Checked = true;

this->item_ConteoVertical->CheckState =
System::Windows::Forms::CheckState::Checked;

this->item_ConteoVertical->Name = L"item_ConteoVertical";

this->item_ConteoVertical->Size = System::Drawing::Size(169, 22);

this->item_ConteoVertical->Text = L"Conteo vertical";

this->item_ConteoVertical->Click += gcnew System::EventHandler(this,
&Form1::item_ConteoVertical_Click);

//

// item_ConteoHorizontal

//

this->item_ConteoHorizontal->Name = L"item_ConteoHorizontal";

this->item_ConteoHorizontal->Size = System::Drawing::Size(169, 22);

this->item_ConteoHorizontal->Text = L"Conteo horizontal";

this->item_ConteoHorizontal->Click += gcnew System::EventHandler(this,
&Form1::item_conteoHorizontal_Click);

//
```

```
// item_Ayuda
//
this->item_Ayuda->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->item_VerAyuda,
this->item_Acerca});
this->item_Ayuda->Name = L"item_Ayuda";
this->item_Ayuda->Size = System::Drawing::Size(53, 20);
this->item_Ayuda->Text = L"Ayuda";
//
// item_VerAyuda
//
this->item_VerAyuda->Name = L"item_VerAyuda";
this->item_VerAyuda->Size = System::Drawing::Size(127, 22);
this->item_VerAyuda->Text = L"Ver Ayuda";
//
// item_Acerca
//
this->item_Acerca->Name = L"item_Acerca";
this->item_Acerca->Size = System::Drawing::Size(127, 22);
this->item_Acerca->Text = L"Acerca";
```

```
//
// panel_Imagen
//
this->panel_Imagen->BackColor = System::Drawing::Color::LightGray;
this->panel_Imagen->Controls->Add(this->label_Video);
this->panel_Imagen->Controls->Add(this->picture_Imagen);
this->panel_Imagen->Location = System::Drawing::Point(10, 36);
this->panel_Imagen->Name = L"panel_Imagen";
this->panel_Imagen->Size = System::Drawing::Size(736, 523);
this->panel_Imagen->TabIndex = 2;
//
// label_Video
//
this->label_Video->BackColor = System::Drawing::Color::Teal;
this->label_Video->ForeColor = System::Drawing::Color::White;
this->label_Video->Location = System::Drawing::Point(2, 0);
this->label_Video->Name = L"label_Video";
this->label_Video->Size = System::Drawing::Size(731, 25);
this->label_Video->TabIndex = 1;
this->label_Video->Text = L"VIDEO PROCESADO";
```

```
this->label_Video->TextAlign =
System::Drawing::ContentAlignment::MiddleCenter;

//

// picture_Imagen

//

this->picture_Imagen->BackColor = System::Drawing::Color::LightGray;

this->picture_Imagen->Image = (cli::safe_cast<System::Drawing::Image^
>(resources->GetObject(L"picture_Imagen.Image")));

this->picture_Imagen->Location = System::Drawing::Point(3, 28);

this->picture_Imagen->Name = L"picture_Imagen";

this->picture_Imagen->Size = System::Drawing::Size(730, 490);

this->picture_Imagen->SizeMode =
System::Windows::Forms::PictureBoxSizeMode::CenterImage;

this->picture_Imagen->TabIndex = 0;

this->picture_Imagen->TabStop = false;

//

// panel_Botones

//

this->panel_Botones->BackColor = System::Drawing::Color::LightGray;

this->panel_Botones->Controls->Add(this->label_Consola);

this->panel_Botones->Controls->Add(this->label_C2);
```

```
this->panel_Botones->Controls->Add(this->label_VC2);
this->panel_Botones->Controls->Add(this->label_VC1);
this->panel_Botones->Controls->Add(this->label_C1);
this->panel_Botones->Controls->Add(this->button_Stop);
this->panel_Botones->Controls->Add(this->button_Play);
this->panel_Botones->Controls->Add(this->label_TituloConsola);
this->panel_Botones->Controls->Add(this->label_Resultados);
this->panel_Botones->Controls->Add(this->label_Controles);
this->panel_Botones->Location = System::Drawing::Point(763, 36);
this->panel_Botones->Name = L"panel_Botones";
this->panel_Botones->Size = System::Drawing::Size(265, 523);
this->panel_Botones->TabIndex = 3;
//
// label_Consola
//
this->label_Consola->BackColor = System::Drawing::Color::White;
this->label_Consola->Location = System::Drawing::Point(3, 278);
this->label_Consola->Name = L"label_Consola";
this->label_Consola->Size = System::Drawing::Size(259, 240);
this->label_Consola->TabIndex = 2;
```

```
//
// label_C2
//
this->label_C2->AutoSize = true;
this->label_C2->Location = System::Drawing::Point(37, 222);
this->label_C2->Name = L"label_C2";
this->label_C2->Size = System::Drawing::Size(91, 16);
this->label_C2->TabIndex = 2;
this->label_C2->Text = L"Contador 2 :";
//
// label_VC2
//
this->label_VC2->Location = System::Drawing::Point(134, 222);
this->label_VC2->Name = L"label_VC2";
this->label_VC2->Size = System::Drawing::Size(91, 16);
this->label_VC2->TabIndex = 2;
this->label_VC2->Text = L"0";
this->label_VC2->TextAlign =
System::Drawing::ContentAlignment::MiddleRight;
//
```

```
// label_VC1
//
this->label_VC1->Location = System::Drawing::Point(134, 189);
this->label_VC1->Name = L"label_VC1";
this->label_VC1->Size = System::Drawing::Size(91, 16);
this->label_VC1->TabIndex = 2;
this->label_VC1->Text = L"0";
this->label_VC1->TextAlign =
System::Drawing::ContentAlignment::MiddleRight;
//
// label_C1
//
this->label_C1->AutoSize = true;
this->label_C1->Location = System::Drawing::Point(37, 189);
this->label_C1->Name = L"label_C1";
this->label_C1->Size = System::Drawing::Size(91, 16);
this->label_C1->TabIndex = 2;
this->label_C1->Text = L"Contador 1 :";
//
// button_Stop
```

```
//
this->button_Stop->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast
<System::Byte>(192)),
static_cast<System::Int32>(static_cast<System::Byte>(0)),
static_cast<System::Int32>(static_cast<System::Byte>(0)));
this->button_Stop->Enabled = false;
this->button_Stop->ForeColor = System::Drawing::Color::White;
this->button_Stop->Location = System::Drawing::Point(3, 86);
this->button_Stop->Name = L"button_Stop";
this->button_Stop->Size = System::Drawing::Size(259, 50);
this->button_Stop->TabIndex = 3;
this->button_Stop->Text = L"STOP";
this->button_Stop->UseVisualStyleBackColor = false;
this->button_Stop->Click += gcnew System::EventHandler(this,
&Form1::button_Stop_Click);
//
// button_Play
//
this->button_Play->BackColor = System::Drawing::Color::Green;
this->button_Play->ForeColor = System::Drawing::Color::White;
```

```
this->button_Play->Location = System::Drawing::Point(3, 33);

this->button_Play->Name = L"button_Play";

this->button_Play->Size = System::Drawing::Size(259, 50);

this->button_Play->TabIndex = 2;

this->button_Play->Text = L"PLAY";

this->button_Play->UseVisualStyleBackColor = false;

this->button_Play->Click += gcnew System::EventHandler(this,
&Form1::button_Play_Click);

//

// label_TituloConsola

//

this->label_TituloConsola->BackColor = System::Drawing::Color::Teal;

this->label_TituloConsola->ForeColor = System::Drawing::Color::White;

this->label_TituloConsola->Location = System::Drawing::Point(0, 253);

this->label_TituloConsola->Name = L"label_TituloConsola";

this->label_TituloConsola->Size = System::Drawing::Size(268, 25);

this->label_TituloConsola->TabIndex = 2;

this->label_TituloConsola->Text = L"CONSOLA";

this->label_TituloConsola->TextAlign =
System::Drawing::ContentAlignment::MiddleCenter;

//
```

```
// label_Resultados
//
this->label_Resultados->BackColor = System::Drawing::Color::Teal;
this->label_Resultados->ForeColor = System::Drawing::Color::White;
this->label_Resultados->Location = System::Drawing::Point(0, 146);
this->label_Resultados->Name = L"label_Resultados";
this->label_Resultados->Size = System::Drawing::Size(268, 25);
this->label_Resultados->TabIndex = 2;
this->label_Resultados->Text = L"RESULTADOS";
this->label_Resultados->TextAlign =
System::Drawing::ContentAlignment::MiddleCenter;
//
// label_Controles
//
this->label_Controles->BackColor = System::Drawing::Color::Teal;
this->label_Controles->ForeColor = System::Drawing::Color::White;
this->label_Controles->Location = System::Drawing::Point(0, 0);
this->label_Controles->Name = L"label_Controles";
this->label_Controles->Size = System::Drawing::Size(268, 25);
this->label_Controles->TabIndex = 2;
```

```
this->label_Controles->Text = L"CONTROLES";

this->label_Controles->TextAlign =
System::Drawing::ContentAlignment::MiddleCenter;

//

// Form1

//

this->AutoScaleDimensions = System::Drawing::SizeF(9, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::DarkSlateGray;

this->ClientSize = System::Drawing::Size(1039, 610);

this->Controls->Add(this->panel_Principal);

this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif",
9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,

static_cast<System::Byte>(0)));

this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedSingle;

this->Icon = (cli::safe_cast<System::Drawing::Icon^  >(resources-
>GetObject(L"$this.Icon")));

this->MainMenuStrip = this->menu_Principal;

this->Margin = System::Windows::Forms::Padding(4);

this->MaximizeBox = false;
```

```
this->Name = L"Form1";

this->Text = L"Sistema para el Conteo de Objetos usando OPENCV";

this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);

this->status_Estado->ResumeLayout(false);

this->status_Estado->PerformLayout();

this->panel_Principal->ResumeLayout(false);

this->panel_Principal->PerformLayout();

this->menu_Principal->ResumeLayout(false);

this->menu_Principal->PerformLayout();

this->panel_Imagen->ResumeLayout(false);

(cli::safe_cast<System::ComponentModel::ISupportInitialize^ >(this-
>picture_Imagen))->EndInit();

this->panel_Botones->ResumeLayout(false);

this->panel_Botones->PerformLayout();

this->ResumeLayout(false);

}
```

2.4.2.Programación del menú Archivo

El menú Archivo consta de cinco ítems seleccionables, los cuatro primeros permiten escogen la fuente del video a procesar y le ultimo cierra la aplicación. Las fuentes de video disponibles son la cámara del

computador, una cámara externa conectada al computador, y dos archivos de video.

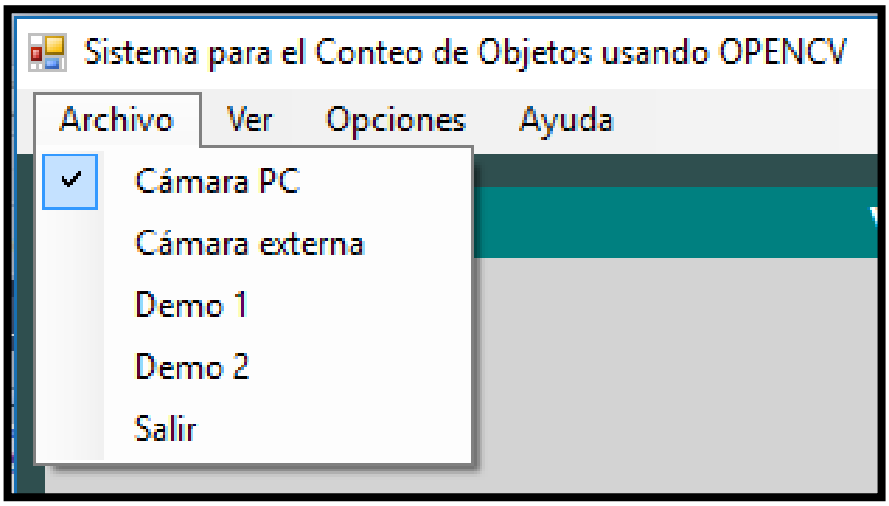

Figura 25: Menú Archivo

Para que el programa pueda identificar que fuente escoger se ha creado una variable global llamada *fuenteVideo* la misma que cambiará de valor dependiendo el ítem seleccionado, en las siguientes líneas se muestra la programación de los ítems del menú Archivo.

```
const static char _CAMARA0_=0, _CAMARA1_=1, _DEMO1_=2,
_DEMO2_=3;

char fuenteVideo;



private: System::Void item_CamaraPC_Click(System::Object^  sender,
System::EventArgs^  e) {

fuenteVideo = _CAMARA0_;

this->item_CamaraPC->Checked = true;

this->item_CamaraExterna->Checked = false;
```

```
this->item_Demo1->Checked = false;

this->item_Demo2->Checked = false;

}


private: System::Void item_CamaraExterna_Click(System::Object^
sender, System::EventArgs^  e) {

fuenteVideo = _CAMARA1_;

this->item_CamaraPC->Checked = false;

this->item_CamaraExterna->Checked = true;

this->item_Demo1->Checked = false;

this->item_Demo2->Checked = false;

}


private: System::Void item_Demo1_Click(System::Object^  sender,
System::EventArgs^  e) {

fuenteVideo = _DEMO1_;

this->item_CamaraPC->Checked = false;

this->item_CamaraExterna->Checked = false;

this->item_Demo1->Checked = true;

this->item_Demo2->Checked = false;

}
```

```
private: System::Void item_Demo2_Click(System::Object^  sender,
System::EventArgs^  e) {

fuenteVideo = _DEMO2_;

this->item_CamaraPC->Checked = false;

this->item_CamaraExterna->Checked = false;

this->item_Demo1->Checked = false;

this->item_Demo2->Checked = true;

}
```

2.4.3.Programación del menú Ver

El menú Ver permite al usuario escoger la etapa de procesamiento que dese monitorear en la aplicación, para esto existen cuatro opciones las cuales son la etapa de pre procesamiento, la etapa de segmentación, la etapa de seguimiento (Tracking) y la etapa final del procesamiento, siendo esta opción la que se definirá por defecto al iniciar el programa.

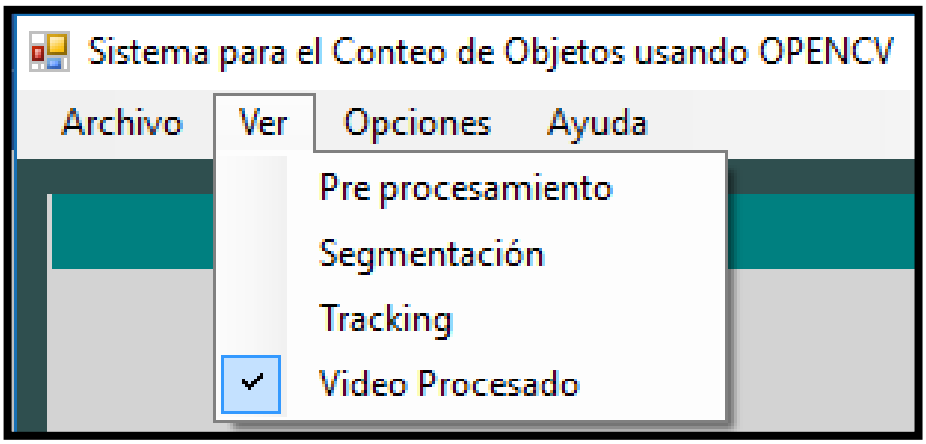

Figura 26: Menú Ver

Para que el programa pueda identificar que etapa del procesamiento se va a mostrar en la interfaz se ha creado una variable global llamada *fuenteImagen* la misma que cambiará de valor dependiendo el ítem seleccionado, en las siguientes líneas se muestra la programación de los ítems del menú Ver.

```
const static char _PREPROCESADO_=0, _SEGMENTACION_=1,
_TRACKING_=2, _PROCESADO_=3;

char fuenteImagen;



private: System::Void item_Preprocesamiento_Click(System::Object^
sender, System::EventArgs^  e) {

fuenteImagen = _PREPROCESADO_;

this->item_Preprocesamiento->Checked = true;

this->item_Segmentacion->Checked = false;

this->item_Tracking->Checked = false;

this->item_VideoProcesado->Checked = false;
```

```
this->label_Video->Text = "PREPROCESAMIENTO";

}


private: System::Void item_Segmentacion_Click(System::Object^  sender,
System::EventArgs^  e) {

fuenteImagen = _SEGMENTACION_;

this->item_Preprocesamiento->Checked = false;

this->item_Segmentacion->Checked = true;

this->item_Tracking->Checked = false;

this->item_VideoProcesado->Checked = false;

this->label_Video->Text = "SEGMENTACION";

}


private: System::Void item_Tracking_Click(System::Object^  sender,
System::EventArgs^  e) {

fuenteImagen = _TRACKING_;

this->item_Preprocesamiento->Checked = false;

this->item_Segmentacion->Checked = false;

this->item_Tracking->Checked = true;

this->item_VideoProcesado->Checked = false;

this->label_Video->Text = "TRACKING";
```

```
}

private: System::Void item_VideoProcesado_Click(System::Object^
sender, System::EventArgs^  e) {

fuenteImagen = _PROCESADO_;

this->item_Preprocesamiento->Checked = false;

this->item_Segmentacion->Checked = false;

this->item_Tracking->Checked = false;

this->item_VideoProcesado->Checked = true;

this->label_Video->Text = "VIDEO PROCESADO";

}
```

2.4.4.Programación del menú Opciones

El menú Opciones se creó con la finalidad de que el usuario le ayude al sistema a identificar la orientación del conteo puesto que la aplicación lo que hace es verificar la trayectoria de los centros de masa de los objetos segmentados, entonces es importante que el sistema conozca en qué dirección se trasladan los mismos.

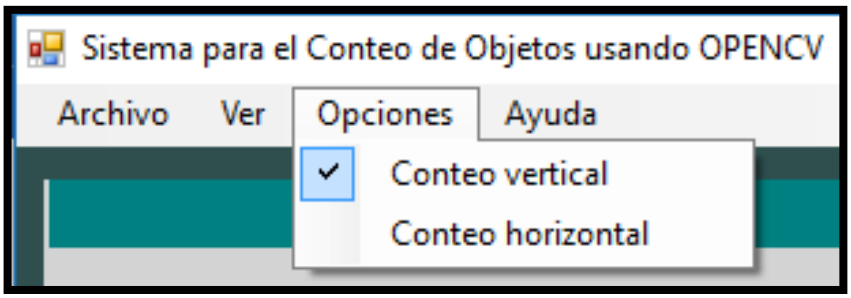

Figura 27: Menú Opciones

Para que el programa pueda identificar la dirección del movimiento de los objetos segmentados se ha creado una variable global llamada *tipoConteo* la misma que cambiará de valor dependiendo el ítem seleccionado, en las siguientes líneas se muestra la programación de los ítems del menú Opciones.

```
char tipoConteo;

const static char _CONTEO_V_=0, _CONTEO_H_=1;



private: System::Void item_ConteoVertical_Click(System::Object^  sender,
System::EventArgs^  e) {

tipoConteo = _CONTEO_V_;

this->item_ConteoVertical->Checked = true;

this->item_ConteoHorizontal->Checked = false;

contador1=0, contador2=0;

this->label_VC1->Text = contador1.ToString();

this->label_VC2->Text = contador2.ToString();

}
```

```
private: System::Void item_conteoHorizontal_Click(System::Object^
sender, System::EventArgs^  e) {

tipoConteo = _CONTEO_H_;

this->item_ConteoVertical->Checked = false;

this->item_ConteoHorizontal->Checked = true;

contador1=0, contador2=0;

this->label_VC1->Text = contador1.ToString();

this->label_VC2->Text = contador2.ToString();

}
```

2.4.5.Programación de los botones de Control

Como ya se mencionó la barra de Controles posee dos botones encargados de la reproducción del video como se muestra en la figura 28.

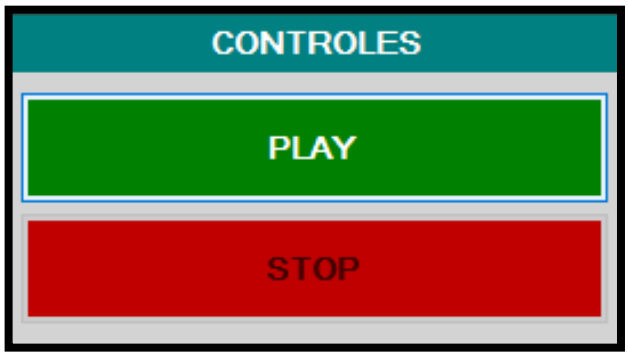

Figura 28: Barra de Controles

El botón PLAY se encarga de reproducir, pausar y reanudar la ejecución del video, mientras que el botón STOP detendrá completamente el video y mostrará los resultados en la consola. A continuación, se muestra la programación de los mismos.

```
VideoCapture video;

double frameWidth, frameHeight, lim1Width, lim2Width, lim1Height,
lim2Height, numFrames, contFrames;

char estado_track[200], _estado[200];

char camara_video, play_pause;

char tipoConteo;

int contador1=0, contador2=0;

double areaMin=0, areaMax=0;


private: System::Void button_Play_Click(System::Object^  sender,
System::EventArgs^  e) {

switch(play_pause){

case 0:

reproducirVideo();

break;

case 1:

pausarVideo();

break;
```

```
case 2:

reanudarVideo();

break;

}

}


private: System::Void button_Stop_Click(System::Object^  sender,
System::EventArgs^  e) {

this->finalizarVideo();

}


private: System::Void reproducirVideo(){

this->label_Estado1->Text = "Estado 1 : Abriendo video...";

this->button_Play->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast
<System::Byte>(192)),
static_cast<System::Int32>(static_cast<System::Byte>(64)),

static_cast<System::Int32>(static_cast<System::Byte>(0)));

this->button_Play->Text = L"PAUSE";

play_pause = 1;
```

```
this->item_CamaraPC->Enabled = false;

this->item_CamaraExterna->Enabled = false;

this->item_Demo1->Enabled = false;

this->item_Demo2->Enabled = false;

this->item_ConteoVertical->Enabled = false;

this->item_ConteoHorizontal->Enabled = false;

switch(fuenteVideo){

case _CAMARA0_:

camara_video = false;

video.open(0);

break;

case _CAMARA1_:

camara_video = false;

video.open(1);

break;

case _DEMO1_:

camara_video = true;

video.open("C:/Users/andres/Documents/Mis
carpetas/ESPE/Tesis/Practico/conteo/res/480p.mp4");

break;
```

```
case _DEMO2_:

camara_video = true;

video.open("C:/Users/andres/Documents/Mis
carpetas/ESPE/Tesis/Practico/conteo/res/240p.mp4");

break;

}

contFrames=0;

contador1=0, contador2=0;

this->label_VC1->Text = contador1.ToString();

this->label_VC2->Text = contador2.ToString();

this->progressBar_Estado->Value = 0;

if(!video.isOpened()){

this->finalizarVideo();

this->label_Estado1->Text = "Estado 1 : Error!!! no se puede abrir el
video";

return;

}

this->button_Stop->Enabled = true;

this->label_Estado1->Text = "Estado 1 : Video abierto...";

this->label_Estado2->Text = "Estado 2 : Reproduciendo video...";
```

```
if(camara_video) numFrames =
video.get(CV_CAP_PROP_FRAME_COUNT)-1;

else this->progressBar_Estado->Value = 100;

video.set(CV_CAP_PROP_FPS, 60.0);

frameWidth  = video.get(CV_CAP_PROP_FRAME_WIDTH);

frameHeight = video.get(CV_CAP_PROP_FRAME_HEIGHT);

lim1Width   = frameWidth*2/5;

lim2Width   = frameWidth*3/5;

lim1Height  = frameHeight*2/5;

lim2Height  = frameHeight*3/5;

areaMin     = frameWidth*frameHeight/400;

areaMax     = frameWidth*frameHeight/4;

this->iniciarTimer1(1000/60);

int i;

for(i=0; i<200; i++){

_estado[i] = 0;

estado_track[i] = 0;

}

}

private: System::Void pausarVideo(){
```

```
this->button_Play->BackColor = System::Drawing::Color::Green;

this->button_Play->Text = L"PLAY";

play_pause = 2;

this->label_Estado2->Text = "Estado 2 : Video pausado...";

temporizador->Enabled = false;

}

private: System::Void reanudarVideo(){

this->button_Play->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast
<System::Byte>(192)),
static_cast<System::Int32>(static_cast<System::Byte>(64)),

static_cast<System::Int32>(static_cast<System::Byte>(0)));

this->button_Play->Text = L"PAUSE";

play_pause = 1;

this->label_Estado2->Text = "Estado 2 : Reproduciendo video...";

this->iniciarTimer1(1000/60);

}

private: System::Void finalizarVideo(){

this->button_Play->BackColor = System::Drawing::Color::Green;

this->button_Play->Text = L"PLAY";

play_pause = 0;
```

```
this->item_CamaraPC->Enabled = true;

this->item_CamaraExterna->Enabled = true;

this->item_Demo1->Enabled = true;

this->item_Demo2->Enabled = true;

this->item_ConteoVertical->Enabled = true;

this->item_ConteoHorizontal->Enabled = true;

this->button_Stop->Enabled = false;

cvReleaseTracks(tracks);

cvReleaseBlobs(blobs);

temporizador->Enabled = false;

this->label_Estado1->Text = "Estado 1 : Cerrando video...";

video.release();

this->label_Estado1->Text = "Estado 1 : Video cerrado...";

this->label_Estado2->Text = "Estado 2 : Video sin reproducir...";

}
```

2.4.6.Programación del procesamiento de imágenes

Como se puede apreciar en la programación de la función *reproducirVideo()* se invoca a la función *iniciarTimer1()* esto se debe a que el programa utiliza un temporizador para procesar los fotogramas de video a intervalos de tiempo constante. Visual Studio 2010 permite crear eventos

de tipo temporizador, en otras palabras, realiza la llamada a una función cada vez que se cumpla con el intervalo de tiempo fijado en el temporizador, de esta forma se libera la programación del procesamiento de imágenes en la función del botón PLAY.

Se escogió la utilización de un evento tipo temporizador para el procesamiento de imágenes para quitar la ejecución del procesamiento del hilo principal del programa y de esta forma evitar que el programa no responda a otros eventos como el botón STOP, recordemos que al usar un lazo repetitivo en una función se impide la respuesta a eventos de las herramientas de visualización. Existe otra alternativa para realizar el procesamiento de imágenes de modo continuo y es creando un hilo paralelo de ejecución (thread) pero lastimosamente la versión del compilador que se escogió no soporta esta herramienta de codificación.

La configuración del temporizador se la realiza únicamente en la etapa de carga de la aplicación, es decir, el intervalo de tiempo desde que se abre el programa hasta que el usuario divisa la interfaz. En las siguientes líneas se aprecia la inicialización del temporizador en la función *Form1_Load()* y la ejecución del procesamiento de fotogramas en la función *procesarVideo()* que es el método de respuesta asignado al evento del temporizador.

```
static System::Windows::Forms::Timer^ temporizador = gcnew
System::Windows::Forms::Timer;



private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {

fuenteVideo = _CAMARA0_;

fuenteImagen = _PROCESADO_;
```

```
play_pause = 0;

tipoConteo = _CONTEO_V_;

grafico = this->picture_Imagen->CreateGraphics();

temporizador->Enabled = false;

temporizador->Tick += gcnew System::EventHandler(this,
&Form1::procesarVideo);

}


void procesarVideo(System::Object ^ sender, System::EventArgs ^ e){

Mat frameOriginal, frameGris, framePreprocesado, frameMascara,
frameSegmentacion, frameProcesado;

video.read(frameOriginal);

if(!frameOriginal.empty()){

cvtColor(frameOriginal, frameGris, CV_RGB2GRAY);

GaussianBlur(frameGris, framePreprocesado, cv::Size(3, 3), 0, 0 );

bs(framePreprocesado, frameMascara, -1);

contFrames++;

if(contFrames>30){

threshold(frameMascara, frameMascara, 0, 255,
THRESH_BINARY+THRESH_OTSU);

dilate(frameMascara, frameMascara, Mat(), cv::Point(-1,-1), 1);
```

```
erode(frameMascara, frameMascara, Mat(), cv::Point(-1,-1), 1);

findContours(frameMascara, contornos, jerarquia,
CV_RETR_EXTERNAL,CV_CHAIN_APPROX_SIMPLE);

frameMascara &= 0;

for(size_t i=0; i < contornos.size(); ++i)

if((contourArea(contornos[i]) > areaMin)&&(contourArea(contornos[i]) <
areaMax))

drawContours(frameMascara, contornos, i, Scalar(255), CV_FILLED, 8);

bitwise_and(frameGris, frameMascara, frameSegmentacion);

IplImage *IPLmask = cvCloneImage(&(IplImage)frameMascara);

IplImage *IPLblobs = cvCreateImage(cvSize(frameMascara.cols,
frameMascara.rows), IPL_DEPTH_LABEL, 1);

unsigned int resultados = cvLabel(IPLmask, IPLblobs, blobs);

IplImage *frameTracking = cvCloneImage(&(IplImage)frameOriginal);

cvUpdateTracks(blobs, tracks, 2.0, 30);

switch(fuenteImagen){

case _PREPROCESADO_:

framePreprocesado.copyTo(frameProcesado);

break;

case _SEGMENTACION_:

frameSegmentacion.copyTo(frameProcesado);
```

```
break;

case _TRACKING_:

cvRenderBlobs(IPLblobs, blobs, frameTracking, frameTracking,
CV_BLOB_RENDER_BOUNDING_BOX|CV_BLOB_RENDER_COLOR);

cvRenderBlobs(IPLblobs, blobs, frameTracking, frameTracking,
CV_BLOB_RENDER_BOUNDING_BOX);

cvRenderTracks(tracks, frameTracking, frameTracking,
CV_TRACK_RENDER_ID);

frameProcesado = cvarrToMat(frameTracking);

break;

case _PROCESADO_:

frameOriginal.copyTo(frameProcesado);

break;

}

if(frameProcesado.type() == CV_8UC1) cvtColor(frameProcesado,
frameProcesado, CV_GRAY2RGB);

if(tipoConteo == _CONTEO_V_){

cv::line(frameProcesado, cv::Point(0, (int)lim1Height),
cv::Point((int)frameWidth, (int)lim1Height), Scalar(0,255,255), 1, 8);

cv::line(frameProcesado, cv::Point(0, (int)lim2Height),
cv::Point((int)frameWidth, (int)lim2Height), Scalar(0,255,255), 1, 8);

}
```

```
else{

cv::line(frameProcesado, cv::Point((int)lim1Width, 0),
cv::Point((int)lim1Width, (int)frameHeight), Scalar(0,255,255), 1, 8);

cv::line(frameProcesado, cv::Point((int)lim2Width, 0),
cv::Point((int)lim2Width, (int)frameHeight), Scalar(0,255,255), 1, 8);

}

for(std::map<CvID,CvTrack*>::iterator it = tracks.begin(); it != tracks.end();
it++){

CvID id = (*it).first;

CvTrack* track = (*it).second;

CvPoint2D64f centroid = track->centroid;

char estado = estado_track[id];

if(tipoConteo == _CONTEO_V_){

if(centroid.y < lim1Height)

estado_track[id] = 1;

else if(centroid.y < lim2Height)

estado_track[id] = 2;

else

estado_track[id] = 3;

if(_estado[id] == 0 && estado == 1 && estado_track[id] == 2)

_estado[id] = 1;
```

```
if(_estado[id] == 1 && estado == 2 && estado_track[id] == 3){

cv::line(frameProcesado, cv::Point(0, (int)lim2Height),
cv::Point((int)frameWidth, (int)lim2Height), Scalar(0,255,0), 2, 8);

contador1++;

this->label_VC1->Text = contador1.ToString();

_estado[id] = 0;

estado_track[id] = 0;

}

if(_estado[id] == 0 && estado == 3 && estado_track[id] == 2)

_estado[id] = 2;

if(_estado[id] == 2 && estado == 2 && estado_track[id] == 1){

cv::line(frameProcesado, cv::Point(0, (int)lim1Height),
cv::Point((int)frameWidth, (int)lim1Height), Scalar(0,255,0), 2, 8);

contador2++;

this->label_VC2->Text = contador2.ToString();

_estado[id] = 0;

estado_track[id] = 0;

}

}

else{

if(centroid.x < lim1Width)
```

```
estado_track[id] = 1;

else if(centroid.x < lim2Width)

estado_track[id] = 2;

else

estado_track[id] = 3;

if(_estado[id] == 0 && estado == 1 && estado_track[id] == 2)

_estado[id] = 1;

if(_estado[id] == 1 && estado == 2 && estado_track[id] == 3){

cv::line(frameProcesado, cv::Point((int)lim2Width, 0),
cv::Point((int)lim2Width, (int)frameHeight), Scalar(0,255,0), 2, 8);

contador1++;

this->label_VC1->Text = contador1.ToString();

_estado[id] = 0;

estado_track[id] = 0;

}

if(_estado[id] == 0 && estado == 3 && estado_track[id] == 2)

_estado[id] = 2;

if(_estado[id] == 2 && estado == 2 && estado_track[id] == 1){

cv::line(frameProcesado, cv::Point((int)lim1Width, 0),
cv::Point((int)lim1Width, (int)frameHeight), Scalar(0,255,0), 2, 8);

contador2++;
```

```
this->label_VC2->Text = contador2.ToString();

_estado[id] = 0;

estado_track[id] = 0;

}

}

}

System::Drawing::Bitmap^ bitmapFrame = gcnew
System::Drawing::Bitmap(frameProcesado.cols, frameProcesado.rows,
frameProcesado.step,
System::Drawing::Imaging::PixelFormat::Format24bppRgb,
(IntPtr)(frameProcesado.ptr()));

System::Drawing::RectangleF rect(0, 0, (float)this->picture_Imagen-
>Width, (float)this->picture_Imagen->Height);

grafico->DrawImage(bitmapFrame, rect);

delete bitmapFrame;

cvReleaseImage(&IPLmask);

cvReleaseImage(&IPLblobs);

cvReleaseImage(&frameTracking);

}

if(camara_video){

this->progressBar_Estado->Value = (int)(contFrames/numFrames*100.0);

if(contFrames == numFrames) this->finalizarVideo();
```

```
}

}

frameOriginal.release();

frameGris.release();

framePreprocesado.release();

frameMascara.release();

frameSegmentacion.release();

frameProcesado.release();

}
```

En la figura 29 se puede apreciar el funcionamiento de la aplicación.

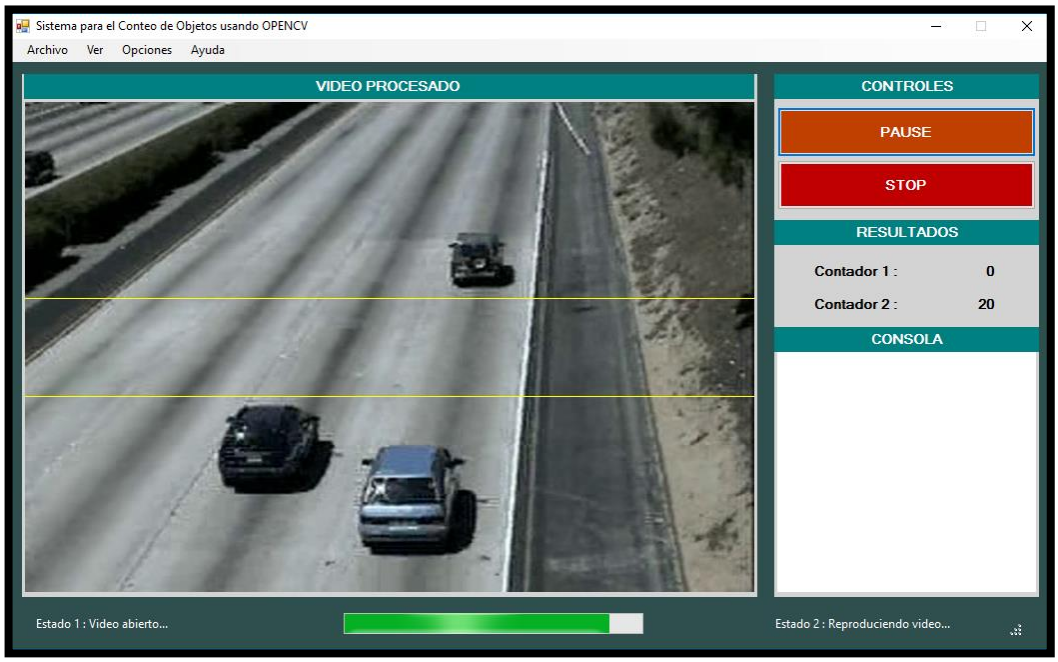

Figura 29: Funcionamiento de la aplicación

CAPÍTULO III

PRUEBAS Y ANÁLISIS DE RESULTADOS

3.1.Introducción

El presente capítulo corresponde al análisis del funcionamiento del sistema que se ha elaborado para lo cual se ha dividido el presente capitulo en tres secciones, en primer lugar, se analizará el costo computacional que produce el programa al someter el procesamiento con fotogramas de diferente resolución.

Posteriormente se muestra la tabulación del error obtenido al verificar el sistema en diferentes pruebas como por ejemplo la posición de la cámara, la orientación del conteo, el tipo de iluminación, entre otros. Finalmente se analiza la información obtenida para encontrar las condiciones donde el sistema presentará el mejor rendimiento.

3.2.Análisis del consumo computacional del sistema

El computador que se emplea para el procesamiento posee un procesador Intel CORE i7 y memoria RAM de 8GB, al estar corriendo la aplicación desarrollada se capturó la imagen del administrador de tareas como se puede observar en la figura 32, en la misma que se puede ver como el sistema utiliza un 41% de la capacidad de procesamiento del CPU y 42 MB aproximadamente en memoria RAM.

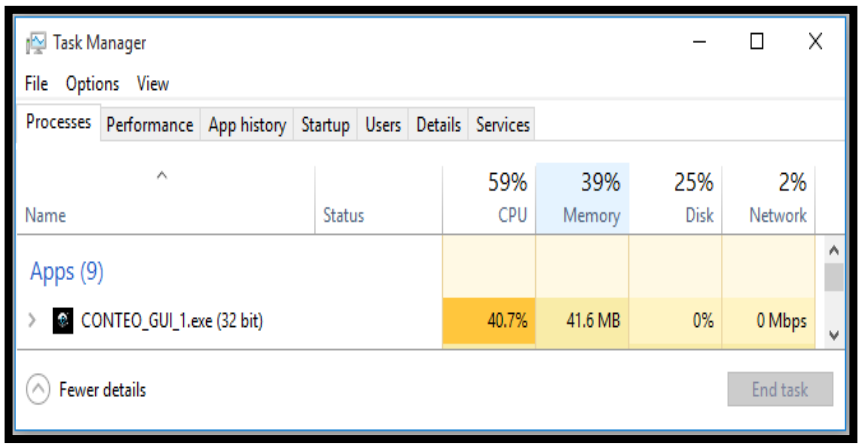

Figura 30: Captura de la ventana del administrador de tareas mientras se ejecuta la aplicación

En lo que se refiere a capacidad de procesamiento se tiene inconvenientes ya que la ejecución de la aplicación conlleva un elevado costo computacional pero la utilización de memoria RAM es satisfactoria debido al bajo porcentaje de trabajo de la misma con lo cual se comprueba la optimización de código de la biblioteca OpenCV. En la tabla 2 se observa los valores de capacidad de procesamiento y consumo de memoria para videos con fotogramas de diferente resolución.

Tabla 2

Consumo computacional del sistema según la resolución del video

RESOLUCIÓN	CPU	MEMORIA
144p	13%	20 MB
240p	21%	27 MB
360p	33%	44 MB
480p	44%	59 MB

Como se puede apreciar los valores de capacidad de procesamiento y consumo de memoria RAM son directamente dependientes de la resolución

de los fotogramas del video lo cual es lógico puesto que a mayor resolución se necesitará mayor espacio en la memoria RAM para almacenar las matrices del procesamiento de cada fotograma y por ende se empleará mayor capacidad del CPU.

3.3.Pruebas de funcionamiento de la aplicación

A continuación, se presentan los resultados obtenidos en diferentes entornos de prueba de la aplicación:

3.3.1.Posición de la Cámara

Para las pruebas de funcionamiento en base a la ubicación de la cámara se plantearon 2 escenarios en los cuales se obtuvieron diferentes resultados como se presenta en las tablas 3 y 4.

3.3.1.1.Posición Colateral

Tabla 3

Resultados al ubicar la cámara en posición colateral a la transición de los elementos contabilizados

TIEMPO DE VIDEO	CONTEO REAL	CONTEO SISTEMA	ERROR
1 MIN	52	47	9,6%
3 MIN	114	102	10,5%
5 MIN	253	219	13,4%
10 MIN	541	438	19,0%

Como se puede apreciar en la tabla anterior existe un error significante al realizar el conteo de objetos con la cámara ubicada colateralmente al movimiento de los mismos, el error es producido principalmente cuando dos objetos se mueven en dirección contraria y se intersecan en la zona de detección del conteo, en este punto el sistema solo ve al objeto más cercano.

3.3.1.2.Posición Superior

Tabla 4

Resultados al ubicar la cámara en la parte superior a la circulación de los elementos contabilizados

TIEMPO DE VIDEO	CONTEO REAL	CONTEO SISTEMA	ERROR
1 MIN	68	68	0,0%
3 MIN	124	125	0,8%
5 MIN	295	290	1,7%
10 MIN	645	632	2,0%

Al ubicar la cámara en la parte superior del escenario el error se reduce notablemente puesto que ahora el sistema puede ver a la mayoría de objetos.

3.3.2.Resolución del video

La siguiente prueba consiste en verificar el error al cambiar la resolución del video para determinar la mejor opción de trabajo del sistema. A continuación, se muestra una tabla con los resultados obtenidos:

Tabla 5

Resultados al someter el Sistema a diferentes resoluciones de video

RESOLUCIÓN	CONTEO REAL	CONTEO SISTEMA	ERROR
144p	36	34	5,6%
240p	36	36	0,0%
360p	36	35	2,8%
480p	36	35	2,8%

A la hora de decidir la resolución de los fotogramas del video a procesar es importante ver el costo computacional que se requiere y el error que se obtiene y como se observa en la tabla 5 el menor error aparece al procesar fotogramas de 240p de resolución, puesto que al procesar fotogramas de menor resolución se afecta la calidad de la información aumentando los conteos erróneos. Por otra parte, al manipular fotogramas de resolución mayor a 240p se necesita mayor tiempo de procesamiento y por ende habrá fotogramas que no alcancen a ser procesados perdiendo información.

3.3.3.Orientación del conteo

El resultado de la contabilización de objetos respecto a la orientación del conteo se muestra en las tablas 6 y 7, se dividieron en dos escenarios, horizontal y vertical, los mismos que presentan los resultados que se analizarán posteriormente.

3.3.3.1.Conteo Horizontal

Tabla 6

Resultados al someter el sistema de conteo a una circulación de objetos en dirección Izquierda – Derecha o viceversa

TIEMPO DE VIDEO	CONTEO REAL	CONTEO SISTEMA	ERROR
1 MIN	68	67	1,5%
3 MIN	124	121	2,4%
5 MIN	295	286	3,1%
10 MIN	645	611	5,3%

3.3.3.2.Conteo Vertical

Tabla 7

Resultados al someter el sistema de conteo a una circulación de objetos en dirección Sur – Norte o viceversa

TIEMPO DE VIDEO	CONTEO REAL	CONTEO SISTEMA	ERROR
1 MIN	68	68	0,0%
3 MIN	124	122	1,6%
5 MIN	295	289	2,0%
10 MIN	645	633	1,9%

Para realizar estas pruebas se utilizó un mismo video en los dos escenarios, en el primer escenario se procesó el video normalmente pero en el segundo escenario se rotó cada fotograma del video noventa grados con la ayuda de la función getRotationMatrix2D(). Fue necesaria la rotación del video para determinar bajo las mismas circunstancias que tipo de

conteo es el más adecuado pero los resultados obtenidos indican que el sistema responde de forma similar en ambos casos.

3.3.4.Tipo de Iluminación

En ocasiones la iluminación es una cualidad del procesamiento de imágenes que se debe tomar muy en cuenta debido a que la misma puede intervenir de manera significativa en el desarrollo normal de la aplicación, los resultados obtenidos se observan en las tablas 8 y 9.

3.3.4.1.Iluminación Natural

Tabla 8

Resultados al someter el sistema de conteo en un ambiente de luz natural

TIEMPO DE VIDEO	CONTEO REAL	CONTEO SISTEMA	ERROR
1 MIN	19	20	5,3%
3 MIN	34	37	8,8%
5 MIN	66	72	9,1%
10 MIN	159	176	10,7%

Los resultados obtenidos en la tabla 8 muestran la vulnerabilidad de la sustracción de fondo en escenarios donde la iluminación cambia rápidamente, los errores más importantes que se pudieron notar en esta prueba son los falsos conteos provocados por la sombra de las nubes que al moverse modifican el fondo del escenario.

3.3.4.2.Iluminación Artificial

Tabla 9

Resultados al someter el sistema de conteo en un ambiente de luz artificial

TIEMPO DE VIDEO	CONTEO REAL	CONTEO SISTEMA	ERROR
1 MIN	27	27	0,0%
3 MIN	54	53	1,9%
5 MIN	107	104	2,8%
10 MIN	183	189	3,3%

Los resultados obtenidos en las dos pruebas son coherentes puesto que en un escenario con iluminación controlada (ver Tabla 9) la sustracción de fondo es más eficiente que en un escenario donde la iluminación cambia con el transcurso del tiempo (ver Tabla 8).

3.3.5.Condiciones Ambiental

Las condiciones ambientales no son las mismas en el transcurso de todo el tiempo, debido a estas circunstancias se consideró analizar 2 escenarios comúnmente presentes, los mismos presentaron los resultados que se muestran en las tablas 10 y 11.

3.3.5.1.Ambiente soleado

Tabla 10

Resultados al someter el sistema de conteo en un día soleado

TIEMPO DE VIDEO	CONTEO REAL	CONTEO SISTEMA	ERROR
1 MIN	47	49	4,3%
3 MIN	132	139	5,3%
5 MIN	268	285	6,3%
10 MIN	498	533	7,0%

En la prueba anterior se determinó que existe una mayor probabilidad de error en escenarios donde la iluminación varía y como se puede ver en la tabla anterior los resultados ratifican dicho análisis en un ambiente soleado.

3.3.5.2.Ambiente lluvioso

Tabla 11

Resultados al someter el sistema de conteo en un día con presencia de lluvia

TIEMPO DE VIDEO	CONTEO REAL	CONTEO SISTEMA	ERROR
1 MIN	52	45	13,5%
3 MIN	133	114	14,3%
5 MIN	282	237	16,0%
10 MIN	462	381	17,5%

Un ambiente lluvioso perjudica aún más al sistema puesto que el fondo del escenario está variando constantemente y esto genera errores significativos en la etapa de segmentación.

3.4.Análisis de resultados obtenidos

Se realizaron varios escenarios de prueba en los que el sistema presentó diversos resultados actuando bajo las condiciones de cada uno de los mismos, la posición ideal de la ubicación del dispositivo óptico respecto al escenario en la que se está captando los fotogramas es la superior ya que así se eliminan falsos conteos por superposición de objetos como ocurre cuando la cámara está en posición colateral al escenario, podemos observar en la tabla 3 la que corresponde a dicha ubicación en la que el error es bastante significante y con el trascurso del tiempo de video este aumenta considerablemente, lo que determina que esta ubicación es inapropiada para colocar la cámara, en la tabla 4 se aprecia que el error presentado al ubicar la cámara en posición superior (arriba) de los objetos en movimiento a contabilizar, es insignificante respecto a la cantidad de elementos en el conteo real, con lo que se llega a determinar que ésta última es la posición adecuada.

El siguiente caso al que se sometió el sistema, es a la variación de la resolución del video, se realizaron pruebas bajo 4 tipos de resolución, al procesar fotogramas de 240p el sistema presenta la menor probabilidad de generar falsos conteos como se observa en la tabla 5 en donde se presentan los resultados, cabe recalcar que para el funcionamiento adecuado del sistema no es prioridad que la resolución de la imagen sea elevada, sino que el procesamiento ocurra de la manera más rápida posible debido a que se está realizando en tiempo real lo que proporciona, de igual manera, un consumo computacional moderado (21%).

Otro de los escenarios al que se puso a prueba el sistema es al sentido de conteo, es decir la orientación que circulan los objetos, en donde el sistema presenta excelentes resultados como se visualiza en las tablas 6 y 7 que corresponde al sentido norte – sur e izquierda – derecha respectivamente. Sin tomar en cuenta la forma en que se realice la orientación del conteo se demuestra la confiabilidad de los algoritmos de seguimiento que se encuentra en la librería cvblob.

La iluminación es un factor crucial en sistemas con segmentación por sustracción de fondo y como se observa en los resultados de las tablas 8 y 9 donde el sistema fue sometido. Colocando la cámara a la intemperie el sistema tendrá más probabilidad de generar falsos conteos, esto se debe principalmente a las sombras generadas por el movimiento de las nubes, es por esto que es preferible ubicar la cámara en escenarios cerrados. Por último, se puede evidenciar en la tabla 12 los errores presentados en todos los entornos de prueba que se realizó al sistema el mismo que arroja como resultado que el mejor escenario para la contabilización de objetos es en una posición de cámara superior con orientación de conteo vertical y una resolución de video de 240p.

Tabla 12

Resultados de las pruebas del sistema aplicado a diferentes entornos

ENTORNO DE PRUEBA	*TIPOS*	*ERROR*
POSICIÓN DE LA CÁMARA	COLATERAL	19,0%
	SUPERIOR	2,0%
RESOLUCIÓN DE VIDEO	144p	5,6%
	240p	0,0%
	360p	2,8%
ORIENTACIÓN DE CONTEO	HORIZONTAL	5,3%
	VERTICAL	2,0%
TIPO DE ILUMINACIÓN	NATURAL	10,7%
	ARTIFICIAL	3,3%
CONDICIONES AMBIENTALES	SOLEADO	7,0%
	LLUVIOSO	17,5%

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1.Conclusiones

- La implementación de un sistema para la contabilización de objetos que circulan por el espacio captado por una cámara utilizando técnicas de visión artificial presenta excelentes resultados ya que con una adecuada ubicación del dispositivo óptico se puede obtener un error nulo a más de las diversas ventajas que ofrece este sistema frente a los tradicionales sistemas mecánicos.
- OpenCV es una biblioteca multiplataforma que permite implementar algoritmos de visión artificial en los diferentes compiladores con resultados inmejorables a pesar de ser software libre.
- La librería cvblob contiene algoritmos para realizar el etiquetado y seguimiento de objetos, los mismos que implementan algoritmos de la biblioteca OpenCV facilitando su importación al código de programa.
- Convertir un fotograma con formato RGB a escala de grises optimiza en un tercio el tiempo de procesamiento del mismo pues se reduce el número de matrices a procesar sin tener que preocuparse por la pérdida de información, es por esto que esta técnica es la primera en utilizarse al recibir una imagen del video.
- La etapa de segmentación es la más importante dentro del tratamiento de imágenes en visión artificial pues es en este punto donde se debe discriminar la información innecesaria y obtener los segmentos de información válidos para el sistema.
- Los algoritmos para sustracción de fondo de un video permiten hallar una matriz de puntos que sirve de modelo de la escena para discriminar el fondo de las imágenes procesadas.

- La sustracción de fondo aplicando texturas de Gauss es la mejor opción para segmentar objetos en movimiento puesto que permite encontrar la máscara de fondo del video continuamente, de esta forma se puede probar el sistema en la mayoría de escenarios.
- La etapa de etiquetado, permite conocer en cada fotograma la posición y la etiqueta (ID) de cada objeto segmentado, esto evita que un objeto pueda ser contado varias veces en una misma trama.
- La etapa de descripción es vital para poder clasificar los objetos segmentados y es el punto de partida para un mejoramiento del presente sistema desarrollado.
- El costo computacional del sistema de conteo es directamente proporcional a la resolución del video.
- La ubicación del sensor óptico es importante para el éxito del sistema de conteo y se ha demostrado que en este sistema la cámara debe ser instalada en la parte superior del escenario.

4.2.Recomendaciones

- La calidad del conteo del sistema está estrechamente relacionada con la ubicación de la cámara, es por esto que se recomienda que el dispositivo óptico se localice en la parte superior del escenario y con un ángulo de vista bajo, caso contrario existirá un error considerable.
- El consumo de procesamiento de nuestro sistema es elevado es por esto que para trabajos futuros sería interesante probar los algoritmos de OpenCV con una tarjeta gráfica.
- Para que el sistema implementado tenga mayor interconectividad se aconseja utilizar una cámara IP de esta forma el conteo de los objetos se podría realizar de forma remota.
- Si se desea asegurar que el tratamiento de la información sobre la cual se está trabajando sea adecuada para las etapas posteriores, es necesario eliminar el ruido que se presenta en las imágenes.
- Se recomienda trabajar con una resolución de video de 240p pues el costo computacional es bajo al igual que la probabilidad de error.
- Para reducir el costo computacional y la probabilidad de generar falsos conteos se aconseja configurar la cámara o el archivo de video para trabajar con fotogramas con 240p de resolución.
- Es preferible implementar el sistema en ambientes cerrados donde se controle de mejor forma la iluminación.

REFERENCIAS BIBLIOGRÁFICAS

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV.* California - United States of America: O'Reilly Media.

Collazos, A. (2009). Detección de Objetos Abandonados. Recuperado el 28 de marzo del 2015, de http://objetosabandonados.blogspot.com/

Genius. (2011). FaceCam 321. Recuperado el 13 de marzo del 2015, de http://www.geniusnet.com/Genius/wSite/ct?xItem=55315&ctNode=1304&mp=3

Itseez. (2014). OPENCV (OPEN SOURCE COMPUTER VISION). Recuperado el 15 de marzo del 2015, de http://opencv.org/

Itseez. (2015). OPENCV. Recuperado el 15 de marzo del 2015, de http://itseez.com/OpenCV/

Justo, F., & Aguirre, I. (2009). Creación de una herramienta que permita mover el cursor de un computador a partir del movimiento ocular, utilizando técnicas de visión artificial. Recuperado el 16 de marzo del 2015, de http://www.laccei.org/LACCEI2009-Venezuela/p256.pdf

Maduell, E. (2010). Visión Artificial. Recuperado el 19 de marzo del 2015, de https://pdf.yt/d/6ofH1olTrxHb9jvQ/download

Martínez, L. (2004). SISTEMA DE VISIÓN PARA EL EQUIPO DE ROBOTS AUTÓNOMOS DEL ITAM. Recuperado el 23 de marzo del 2015, de http://robotica.itam.mx/documents/TESIS-LAMG-SSL.pdf

Microsoft. (2012). Introducción a Visual Studio. Recuperado el 10 de abril del 2015, de https://msdn.microsoft.com/es-es/library/fx6bk1f4(v=vs.100).aspx

Microsoft. (2015). Ventanas de flujo de trabajo de Visual Studio. Recuperado el 10 de abril del 2015, de https://msdn.microsoft.com/es-es/library/bb552457%28v=vs.90%29.aspx

Visual Studio Community. (2015). Introducción a Visual Studio. Recuperado el 11 de abril del 2015, de https://www.visualstudio.com/es-es/products/visual-studio-community-vs.aspx

ANEXOS

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por los señores: **ESTEFANÍA DAYANA MULLO LÓPEZ y CARLOS ANDRÉS MORENO MOLINA**

En la ciudad de Latacunga a los 14 días del mes de Septiembre del 2016

Ing. Eddie Galarza Zambrano
DIRECTOR DEL PROYECTO

Ing. José Bucheli Andrade
CODIRECTOR DEL PROYECTO

Aprobado por:

Ing. Franklin Silva Monteros
DIRECTOR DE CARRERA

Dr. Rodrigo Vaca Corrales
SECRETARIO ACADÉMICO