Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Proyecto terminal:

Simulador de los movimientos y acciones del ratón a través de la captura de los movimientos de la mano en tiempo real.

Presenta:

Vega Vázquez Agustín
206301147

Asesores: Dra. González Brambila Silvia Beatriz
M. en C. Figueroa González Josué

Diciembre 2011

Agradecimientos

Contenido

Índice de Figuras

Resumen

En esta sección se presenta una breve descripción del diseño y desarrollo del proyecto.

Resumen

El desarrollo de este proyecto se basa en la necesidad por hacer más cómoda la vida del ser humano, y hacer que la interacción del ser humano y la computadora, sea cada vez más fácil y más amena para el usuario.

El proyecto trata de simular las acciones del ratón, por medio de capturas de video de una mano, de dónde se obtienen distintas posiciones y movimientos de la mano, las cuales son asignadas a cada acción del ratón, con lo cual se pretende interactuar con la computadora dejando a un lado el ratón convencional.

Durante el desarrollo, se logró construir lo que solo eran ideas en un principio, por lo cual después de la ardua tarea de la investigación y la documentación, se procedió a programar, para lo cual se utiliza como base el lenguaje de programación C++, y la ayuda de las bibliotecas de OpenCV, con la cuales se trabajó en el entorno de desarrollo DevC++ 4.9.

El primer paso fue desarrollar la interfaz gráfica, pues era fundamental poder desplegar las pruebas y los resultados de estas, posteriormente, se procedió a crear un modelo por medio de funciones matemáticas, que permitiera distinguir la forma de una mano, que posteriormente, se convertiría en el patrón base. Las primeras pruebas se realizan solo con el seguimiento (*tracking*) de la mano, teniendo algunos problemas con el tono de piel, es por eso que se decidió crear el modelo de la mano, para no tener que seguir un color sino una forma. Pero para poder lograr lo antes mencionado se realizaron pruebas, de donde surgió la implementación de un histograma de colores, que arrojaran qué colores se asemejaban más al color de la piel, y así poder evitar distractores en el video. De igual manera se implementó la vista en retroproyección para la captura del video, esto después de que se creó el modelo de la mano, y así poder hacer pruebas, y estar seguros de que lo que se reconocía y seguí era la mano, y no el color de la mano.

Posterior a las pruebas antes mencionadas, se trabajó con el cambio de posiciones y movimientos de la mano para poder hacer su reconocimiento y asignación a cada acción del ratón. En este paso se tuvieron algunas dificultades técnicas con algunos planteamientos de la propuesta, pues como no se habían tomado en cuenta factores como la resolución de la cámara con la cual se trabajó, algunos movimientos eran casi imperceptibles para esta; por lo cual se decidió utilizar otras posiciones más marcadas para que la aplicación trabaje de manera adecuada.

En general el proyecto avanzó paulatinamente encontrando errores y corrigiéndolos, se pudo salir avante de las dificultades; pero la parte más importante, es que se cumplieron con los objetivos de la propuesta, con lo cual se

puede dar por concluido satisfactoriamente el presente proyecto.

Se pretendería que se trabajara en un futuro y se continuara con la integración de este proyecto a un sistema operativo, o quizás trabajar con sensores dejando de lado las cámaras y así hacer más fácil la vida del hombre que es la justificación que se buscó para este proyecto.

Descripción

En esta sección se presenta una breve descripción de lo qué es el proyecto desarrollado, incluyendo los objetivos, justificación, antecedentes y descripción técnica.

Introducción

El presente trabajo pretende mostrar las fases de diseño e implementación de una aplicación que simula los movimientos u operaciones que se pueden realizar con el ratón, esto se logrará a través de la detección de los movimientos de la mano los cuáles serán procesados en tiempo real a través de video.

El trabajo está realizado con el lenguaje de programación C++, las interfaces se realizaron con ayuda del entorno de desarrollo Dev C++ 4.9. Y se utilizaron las bibliotecas de visión artificial de OpenCV[1] para el desarrollo de dicha aplicación.

Para conseguir los resultados que se esperaban, se recurrió al uso de funciones de OpenCv, y al uso de modelos matemáticos de algebra lineal, para poder realizar la retroproyección, y la creación de un modelo de la mano, por medio de proyecciones ortogonales de los pixeles de las imágenes captadas en tiempo real por medio de la cámara.

Se hace hincapié en que el proyecto, solo sienta las bases para la simulación de los eventos del ratón de una computadora, para poder realizar trabajos futuros, y poder integrarse a un sistema operativo.

Se realizaron pequeños cambios respecto a la propuesta original, debido a las dificultades que se presentaron conforme avanzaba el desarrollo de este proyecto, y de algunos aspectos que no se habían tomado en cuenta, como la resolución y velocidad de la cámara, o el desperdicio de memoria, con la creación y despliegue de ventanas paralelas en nuestra aplicación.

En general, con el trabajo realizado, a lo largo de tres trimestres, desde que se inició con el planteamiento de la propuesta, podemos decir que se cumplieron todos los objetivos marcados, con lo cual podemos decir que el proyecto cumple con todo lo que se planteó desde un principio, dejando una base sustentable para poder realizar trabajos futuros en él.

[1] Bibliotecas de funciones creadas por Intel, para trabajar con visión artificial

Objetivos.

OBJETIVO GENERAL

Diseñar e implementar una aplicación que permita simular las operaciones del ratón a través de la detección de los movimientos de la mano capturados en video.

OBJETIVOS ESPECÍFICOS

- Diseñar e implementar un módulo para la captura de video de la mano
- Diseñar e implementar un módulo para la detección de los movimientos de la mano
- Diseñar e implementar un módulo para determinar el tipo de movimiento realizado
- Diseñar e implementar una aplicación para probar los movimientos de la mano y su relación con las acciones del ratón
- Realizar pruebas a los módulos realizados

Antecedentes

En esta sección se presentan diversos proyectos que están relacionados con la detección y clasificación de movimientos de la mano.

Referencias Internas en la Universidad Autónoma Metropolitana Azcapotzalco.

Un proyecto similar en cuanto a la detección de movimientos de la mano, es "Sistema de aprendizaje del alfabeto dactilógico mediante procesamiento de imágenes utilizando software libre", [1] sin embargo este sólo reconoce y procesa ciertas posiciones de la mano para poder generar un alfabeto útil para personas sordomudas. Cabe mencionar que este proyecto también se realizó con ayuda de OpenCv y se encuentra concluido.

No existe dentro de la UAM Azcapotzalco un proyecto de procesamiento de imágenes en tiempo real que permita simular las acciones del ratón en una computadora.

Referencias Externas

Finger Mouse [2]: es una aplicación creada para utilizar el dedo de una mano como cursor de una computadora, apoyándose de una cámara. Este proyecto tiene cierto tipo de restricciones como la iluminación y el fondo. Está escrito en Visual C++.

Al igual que otros algoritmos de visión por ordenador, esta aplicación funciona con restricciones (iluminación, fondo, etc.).

Este es el proyecto se hizo en la Universidad de Columbia. Fue hecho bajo la asesoría del profesor John Kender. La aplicación rastrea el dedo (con una webcam) y el movimiento de los dedos para el movimiento del cursor del ratón en la pantalla.

Camera mouse [3]: Es una aplicación que permite controlar el puntero del ratón sobre la pantalla de la computadora con Windows con sólo mover la cabeza. El programa fue desarrollado para ayudar a las personas con discapacidad a utilizar la computadora. El público principal de este programa son las personas que no tienen un control fiable de la mano, pero que pueden mover la cabeza.

Camera mouse funciona mejor con los programas de aplicación que requieren

sólo un ratón y un clic a la izquierda y que no tienen objetos pequeños. Es más fácil de usar la cámara del ratón con los programas de aplicación que no requieren una precisión extrema.

Para utilizar Camera mouse se necesita una cuenta de Windows 7, Windows Vista o Windows XP y una webcam.

Justificación

Desde la aparición de las computadoras con un entorno gráfico, se ha tenido la necesidad de diseñar nuevos dispositivos periféricos, que nos puedan ayudar al manejo eficaz y cómodo de estos equipos.

El ratón ha sido desde su aparición uno de los dispositivos, más usados en el manejo de las computadoras, pues con los sistemas operativos modernos, que innovan en cuestiones gráficas día tras día, es necesario un mayor manejo de este dispositivo, para sacar provecho a estos sistemas, y no solo estos, sino también existen muchas aplicaciones y herramientas, que son especialmente manejadas por medio del cursor de la computadora; por lo cual el ratón es parte esencial de casi todos los sistemas de hoy en día.

Aunque de la misma manera, con los avances antes mencionados, también existen innovaciones tecnológicas en el hardware, como la aparición de las pantallas táctiles (*touch*), que desechan un poco las funciones del ratón, pero para la utilización de esta tecnología sigue siendo necesario el contacto físico entre la computadora y el usuario.

Con el presente proyecto se pretende dejar de lado el contacto antes mencionado, bastando con sustituirlo por los movimientos naturales del cuerpo humano, en este caso utilizando la mano como referencia.

Este proyecto tiene distintas aplicaciones; entre las cuales remarcaría, la comodidad para poder hacer uso de una computadora en áreas específicas, por ejemplo en la medicina, donde cada centímetro es vital para que los médicos realicen sus labores, sin que nada les pueda estorbar, bastaría únicamente con mover la mano, para dirigir una cámara, un bisturí, etc. Haciendo más fácil la labor del médico, y ayudándolo a tener un desempeño optimo en su trabajo. Otro ejemplo seria la docencia, pues al hacer uso de presentaciones electrónicas, no sería indispensable, estar pegado a la computadora; únicamente bastaría con apuntar la mano en dirección a una cámara y desde ahí gestionar el desarrollo de la presentación.

Este proyecto se basa en la búsqueda de la innovación tecnológica, la interacción máquina hombre y la comodidad del usuario enfocado a la no

dependencia de contacto directo, para poder tener una interacción.

Especificaciones técnicas

El procesamiento del video y detección de movimientos de la mano se realizaran utilizando las bibliotecas de OpenCV.

La posición de la mano para poder ser detectada será la mostrada en la *Figura 1.* Es necesario adoptar esta posición, ya que si los dedos se separan mucho se corre el riesgo que la aplicación, no detecte la mano para su posterior seguimiento. Y por otro lado si se juntan mucho, la aplicación captaría la posición para la simulación del clic izquierdo, pero en esta primera fase lo que se desea es que se realice un seguimiento de la mano para su desplazamiento.

Figura 1: Posición de la mano para captura del video.

Los movimientos del ratón a simular con el movimiento de la mano serán:

- Desplazamiento: Desplazamiento de la mano.
- Clic izquierdo: Flexionar y extender el dedo índice.
- Clic derecho: Contraer los dedos de la mano.

Cabe mencionar, que estos movimientos y posiciones, varían un poco de la propuesta original, ya que el hecho de trabajar con una cámara web convencional, dificulta la captación de posiciones similares, por la resolución tan baja en el dispositivo, lo que provoca que algunos pixeles se distorsionen, esto aunado al hecho de que la velocidad en *frames* (cuadros) por segundo de la cámara hace más lenta la transición en las imágenes haciendo casi imperceptible el movimiento que se vincula a una acción del ratón.

Productos de trabajo.

Los productos a entregar que acompañarán al presente reporte final en el CD son:

- Código fuente documentado.
- Archivos necesarios para la ejecución.
- Manual de usuario.

Recursos.

Para la realización del proyecto se contó con una computadora personal con un procesador AMD Athlon™ Processor TF-20 a 1.60 GHz y 2.00 GB de memoria RAM, y con una cámara web; la cual para el proyecto contaba con objetivo de enfoque automático, vídeo de alta definición (captura hasta 1600 x 1200), imágenes de hasta 8 megapíxeles y vídeo de hasta 30 cuadros por segundo, con la cual se realizó la captura del video de la mano.

Se tuvieron a la mano todas las herramientas de software necesarias para la realización, como lo fue la plataforma Windows 7, el entorno de desarrollo Dev C++ 4.9 y las bibliotecas de OpenCV.

No se requirió ningún recurso adicional a los disponibles para realizar el proyecto.

Diseño

En esta sección se presenta el diseño del sistema que incluye comunicación entre bloques, y diagramas de flujo.

Estructura General del Proyecto.

La estructura general del proyecto se basa en cuatro acciones principales: Capturar Video, Seguimiento de la mano y dedos, Reconocimiento de las posiciones y movimientos, y Vincular a las acciones del ratón. A continuación se describen cada una de ellas.

Capturar Video

Esta acción se refiere a obtener un video de la mano con ayuda de una cámara web, para posteriormente trabajar con el video capturado, y realizar pruebas y algoritmos de seguimiento y reconocimiento. Ver *Figura 2.*

Figura 2: Imagen Ilustrativa Capturar Video.

Seguimiento de la Mano y Dedos.

Seguimiento, se refiere, a que nuestra aplicación se enfoque en las manos, y que se pueda realizar un acompañamiento de la misma, pues en si esta parte es la parte principal con la que nuestra aplicación va a trabajar, y en la que se van a basar las acciones posteriores a la captura del video. Ver *Figura 3.*

Figura 3: Imagen Ilustrativa Seguimiento de la Mano y Dedos.

Reconocimiento de las Posiciones y Movimientos.

Esta acción se considera la más importante, ya que si no podemos lograr un reconocimiento y diferenciación entre algunas posiciones de la mano o incluso movimientos de esta, el proyecto no podría llevarse a cabo. El reconocimiento de la mano o la detección de la misma, nos ayuda a encontrar una posición diferente para cada acción que pueda asociarse al ratón. Un ejemplo de una posición que puede tomar la mano se observa en la *Figura 4.*

Figura 4: Imagen Ilustrativa Reconocimiento de las Posiciones y Movimientos.

Vincular las Acciones del Ratón.

Esta acción del proyecto, se basa en proporcionar a cada una de las posiciones y movimientos de la mano reconocidos, una función propia del ratón. Esto quiere decir que para cada clic, izquierdo o derecho, y el desplazamiento del ratón, se va a tener una posición distinta de la mano. Ver *Figura 5.*

Figura 5: Imagen Ilustrativa Vincular las Acciones del Ratón.

Secuencia y Continuidad de la Estructura General de la Aplicación.

Las cuatro acciones principales en las que se basa la aplicación, son continuas, es decir, una vez que se inicia la captura de video, las acciones posteriores se realizan continuamente, en un ciclo, que sólo el usuario puede detener. Lo anterior se ilustra en la *Figura 6.*

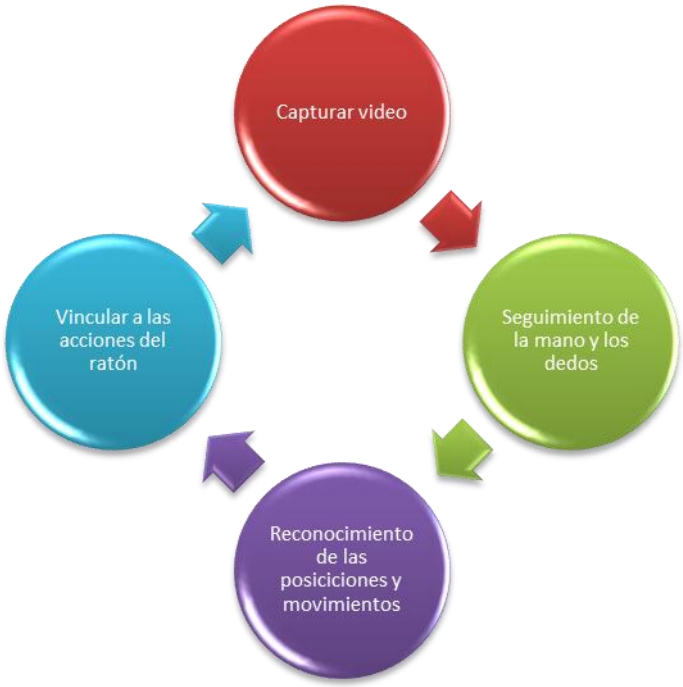

Figura 6: Secuencia y continuidad de la estructura general de la aplicación.

Descripción de los Sub-Bloques.

En esta sección de presentan y describen cada uno de los sub-bloques particulares del proyecto, así como también la comunicación que existe entre ellos.

Capturar video de la mano.

Su función es recibir el video que está siendo capturado por la cámara de los movimientos de la mano. Tal como se ilustra en el la *Figura 7.*

Figura 7: Primer módulo de la aplicación.

El primer paso es, lograr captar el video de la mano, con la ayuda de las bibliotecas de OpenCV, y colocar dicho video en una ventana especialmente creada para trabajar sobre ella.

Detectar los movimientos de la mano.

Su función es determinar las acciones de la mano, separando los eventos de los dedos con los cuales se determinarán los movimientos que el usuario realice con el ratón. Tal como se ilustra en la *Figura 8.*

Figura 8: Segundo módulo de la aplicación.

Determinar el tipo de movimiento de la mano.

Permite saber cómo se está moviendo la mano; si de izquierda a derecha, de arriba hacia abajo o viceversa, de igual modo saber si se flexionan o extienden los dedos, ya que con este movimiento se encuentran representados los clics del ratón. Tal y como se ilustra en la *Figura 9.*

Figura 9: Tercer módulo de la aplicación.

Desplegar resultados.

Muestra en una interfaz gráfica los tipos de movimientos de la mano y el evento del ratón asociado a éste.
Se desplegarán indicadores dependiendo del evento que se esté simulando; por ejemplo, dibujar una línea de un color para movimiento normal, dibujar un círculo para el alrededor de la mano y que este permanezca con la mano a lo largo de su trayecto y avisos de cuándo se presione un botón. Tal como se ilustra en la *Figura 10.*

Figura 10: Cuarto módulo de la aplicación.

Comunicación entre sub-bloques.

El módulo o sub-bloque de captura de video de la mano, haciendo uso de las bibliotecas de OpenCV, procesa el video y lo envía al sub-bloque de detección de los movimientos de la mano. Éste separa los movimientos y elementos de la mano necesarios para identificar los eventos realizados por el ratón y enviar la información al sub-bloque de determinación de los tipos de movimientos de la mano; el cuál, en base al procesamiento de la información indicará al módulo desplegar resultados, que presente en la interfaz el evento ocurrido.

El flujo que siguen las funciones del proyecto se presentan en el diagrama mostrado en la *Figura 11.*

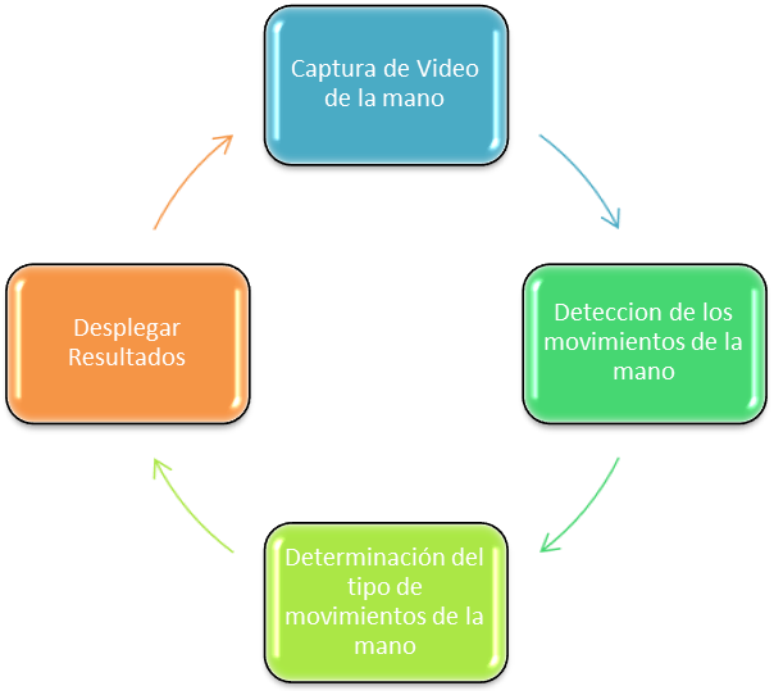

Figura 11: Interacción y flujo entre sub-bloques y funcionalidades del proyecto.

Implementación

En esta sección se describe el proceso de implementación del sistema.

La sección de implementación no contiene el código fuente, este se puede revisar en el directorio correspondiente del CD en el que se incluye este reporte. Las capturas de pantalla se muestran a lo largo del presente documento.

Consideraciones iniciales.

El primer paso para poder realizar la implementación, es asegurarnos de que tenemos instalas y configuradas las bibliotecas de OpenCV, así como el entorno de desarrollo, que en este caso es Dev C++ 4.9.

Posteriormente a la configuración se verifica, que la cámara este bien instalada, esto quiere decir que este activa, y que tenga todos sus controladores instalados y actualizados.

Y como último paso antes de iniciar con la programación, es necesario que calibremos la cámara, para poder enfocar claramente el objetivo; cabe señalar que este proceso se encuentra descrito más adelante, en el presente documento.

Teniendo en cuenta las consideraciones anteriores, entonces si pasamos a la programación, y a la utilización de las bibliotecas de OpenCV, a continuación descritas.

Bibliotecas Utilizadas.

El proyecto está basado en una programación con lenguaje C++, pero para poder hacer uso de la visión artificial que es en la cual se basa, es necesario utilizar las bibliotecas de OpenCV. Se decidió utilizar estas bibliotecas por su versatilidad, ya que son de libre acceso, también es importante señalar que OpenCv es multiplataforma. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

Las bibliotecas de OpenCV están divididas en cinco grandes grupos como se observa en la figura *Figura 12*:

a) CXCORE: donde se encuentran las estructuras y algoritmos básicos que usan las demás funciones.

b) CV: donde están implementadas las funciones principales de procesamiento de imágenes.

c) HighGUI: todo lo relacionado a la interfaz gráfica de OpenCV y las funciones que permiten importar imágenes y video (actualmente *ffmpeg*, *Cvcam*, etc.).

d) ML: que cuenta con algoritmos de aprendizaje, clasificadores y demás.

e) CvAux: con funciones experimentales.

Figura 12: Bibliotecas de OpenCV

Modelos matemáticos usados.

Laplace (*src* , *dst* , *apertureSize* = *3*) → Ninguno

Calcula el Laplaciano de una imagen.

Parámetros:
src (CvArr) - Fuente de la imagen
dst (CvArr) - imagen de destino
apertureSize (int) - tamaño de la apertura (que tiene el mismo significado que *Sobel*)

La función calcula el Laplaciano de la imagen original por la suma de los x segundos y derivados y, se calculan utilizando el operador de Sobel[2]:

$$\texttt{dst}(x,y) = \frac{d^2\texttt{src}}{dx^2} + \frac{d^2\texttt{src}}{dy^2}$$

Establecen *apertureSize* = 1 da el mejor variante que es igual a la convolución [3]de la imagen con el kernel:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Al igual que la función de *Sobel*, no se lleva a cabo la ampliación y las mismas combinaciones de formatos de entrada y de salida son compatibles.

[2] Utilizado en procesamiento de imágenes, especialmente en algoritmos de detección de bordes.
[3] Operador matemático que transforma dos funciones en una tercera.

Sobel (src , dst , xorder , yorder , apertureSize = 3) → Ninguno

Calcula la primera, segunda derivada, la imagen de terceros o mixtos con un largo operador de Sobel.

Parámetros:
- src (CvArr) - Fuente de la imagen de tipo CvArr *
- dst (CvArr) - imagen de destino
- xorder (int) - Orden de la derivada x
- yorder (int) - Orden de la derivada y
- apertureSize (int) - Tamaño de la extensión del kernel de Sobel, debe ser de 1, 3, 5 o 7

$$\begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}$$

para el x-derivado o adaptado para el y derivados.

La función calcula la derivada de imágenes mediante la convolución de la imagen con el núcleo adecuado:

$$\texttt{dst}(x,y) = \frac{d^{xorder+yorder}\texttt{src}}{dx^{xorder} \cdot dy^{yorder}}$$

Los operadores de Sobel combinar suavizado de Gauss y la diferenciación de lo que el resultado es más o menos resistente al ruido. Muy a menudo, la función se llama con (xorder = 1, yorder = 0, apertureSize = 3) o (xorder = 0, yorder = 1, apertureSize = 3) para calcular el x-o primera derivada y la imagen. El primer caso corresponde a un núcleo de:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

y el segundo corresponde a un núcleo de:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

o un grano de:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{bmatrix}$$

en función del origen de la imagen.

Calibración de la Cámara

Las librerías OpenCV dan la posibilidad de utilizar las funciones que tienen para la Calibración de cámaras. Las funciones de calibración de cámaras son usadas para calcular los parámetros intrínsecos e extrínsecos de la cámara. Los parámetros de la cámara son una serie de números que describen una configuración particular de la cámara.

Los parámetros intrínsecos son aquellos que especifican una característica propia de la cámara incluye:

- distancia focal, que es, la distancia entre la lente de la cámara y el plano sensor,
- la localización en la imagen de su centro en píxeles,
- el tamaño efectivo del píxel, y
- los coeficientes de distorsión radial de la lente.

Los parámetros extrínsecos de la cámara describen la relación espacial entre la cámara y el mundo, incluye:

- la matriz de rotación y
- el vector de translación.

Lo que especifican es la transformación entre la cámara y la referencia de imágenes del mundo.

Para poder utilizar estas librerías son necesarias también las librerías Imagine procesing Library[4], también de Intel. Lo interesante de dichas librerías es el poder utilizarlo tanto en Linux como en Windows. Una posibilidad que brindan bajo el sistema operativo Windows son las DirectX. Ya que con ellas es posible hacer módulos independientes, es decir filtros[5].

[4] Plataforma independiente de manipulación de imágenes de C/C++.
[5] la salida es una combinación lineal de los píxeles en una vecindad para dar suavizado, bordes, perfilado, etc.

Los pasos siguientes son usados para realizar la calibración de la cámara:

1. Encontrar la homografía de todos los puntos de una serie de imágenes.
2. Inicializar los parámetros intrínsecos; la distorsión es puesta a cero.
3. Encontrar los parámetros extrínsecos para el patrón de cada imagen.
4. Hacer una optimización general minimizando el error de la proyección de puntos con todos los parámetros.

Todos los pasos anteriores los realizamos con la ayuda de la función:

cvCalibrateCamera: Calibra la cámara con una precisión normal. La función no devuelve nada. Sus argumentos son: Número de imágenes, arreglo con el número de puntos de cada imagen, tamaño de la imagen, puntero a las imágenes, puntero al patrón, arreglo con los cuatros coeficientes de distorsión encontrados, la matriz de la cámara encontrada, arreglo de vectores de translación por cada posicionamiento del patrón en la imagen, arreglo de las matrices de rotación para cada posición del patrón en la imagen y la suposición de parámetros intrínsecos, si es igual a uno, se necesita suponer unos parámetros para el cálculo.

Desarrollo de los bloques.

Para la programación utilizamos el lenguaje de programación C++, y las bibliotecas de OpenCV de las cuales a su vez utilizamos algunas funciones para el desarrollo de cada uno de los bloques y la interfaz gráfica, entre las más importantes están:

Crear ventana para el despliegue del video: Esta función sirve para crear una ventana, ya sea para desplegar video o desplegar imágenes, en este caso será un video captado en tiempo real.

```
int  cvNamedWindow ( const char * ventana ) ;
void  cvMoveWindow ( const char * ventana , 60 , 80 ) ;
```

Visualizar una imagen: Sirve para visualizar imágenes desde la computadora odesde algún servidor remoto.

```
void cvShowImage ( const char * name , constCvAr r * im ) ;
```

Esperar un clic: Se utiliza para esperar eventos de nuestro ratón.

```
int cvWaitKey ( int delay =0);
```

Visualizar un parámetro y asociar una función de devolución de llamada a sus cambios: Con esta función podemos ver algunos parámetros de la misma desplegados en la pantalla, y a su vez asociar el parámetro a otra función para realizar cambios.

```
int cv Create Trackbar ( const char * trackbar name ,
const char * window name ,
int *  value , int count ,
Cv Trackbar Callback on change )
```

Sistema de macros incluidos en OpenCV para hacer funciones manejando errores: Esta función trabaja análogamente como las excepciones en JAVA, solo que esta función es exclusiva para trabajarla con OpenCV**.**

```
/* Raises an error within the current context */
#define CV ERROR( Code , Msg )
{
cvError ( ( Code ) , cvFuncName , Msg , FILE , LINE ) ;
EXIT ;}
```

Cargar una imagen: Sirve para cargar una imagen desde el disco duro de la computadora, o utilizando como parámetro una dirección URL.

```
IplImage * cvLoadImage ( const char * filename ,
Int f lags=CVLOADIMAGECOLOR ) ;
```

Salvar una imagen: Con ella podemos guardar una imagen en el algún medio de almacenamiento masivo.

```
int cvSaveImage ( const char * f ilename , constCvAr r * image ) ;
```

Formatos soportados: JPEG, PNG, PNM, BMP, TIFF.

Cargar imágenes a partir de un archivo video: Sirve para desplegar en una ventana un video, guardado en la computadora, o utilizando como parámetro en una dirección URL.

```
CvCapture * cvCreate File Capture ( const char * f ilename ) ;
int cvGrabFrame ( CvCapture * capture ) ;
```

```
I p l I m a g e ∗ c v R e t r i e v e F r a me ( CvCapture ∗ capture ) ;
```

Ejemplo:

```
CvCaptur e ∗ capture = cvCaptureFromAVI ( ” infile.avi ” ) ;
I p l I m a g e ∗ img = 0 ;
i f ( ! cvGrabFrame ( capture ) ) { // Captura
printf ( ” No se puede calcular un frame \n” ) ;
exit ( 0 ) ;
}
img=cv Retrieve Frame ( capture ) ; // Recuperar el frame
```

Cargar imágenes a partir de una cámara: Con esta función logramos cargar imágenes en pantalla captadas desde una cámara web asociada al sistema.

```
CvCapture∗ cvCreate CameraCapture ( int index ) ;
int cvGrabFrame ( CvCapture ∗capture ) ;
Ipl Image ∗ cvRetrieveFrame ( CvCaptur e ∗ capture ) ;
```

Ejemplo:

```
CvCapture∗ capture = cvCaptureFromCAM ( 0 ) ;
Ipl Image∗ img = 0 ;
i f ( ! cvGrabFrame ( c a p t u r e ) ) { // Captura
p r i n t f ( ” So se puede capturer un frame \n” ) ;
e x i t ( 0 ) ;
}
img=c v R e t r i e v e F r a me ( c a p t u r e ) ; // Recuperar el  frame
```

Dibujar círculos: Con ella podemos dibujar círculos ya sea en una ventana en blanco, sobre imágenes o incluso sobre videos.

```
cvCircle(img, cvPoint(100,100), 20, cvScalar(0,255,0), 1);
```

Dibujar Rectángulos: Con ella podemos dibujar rectángulos ya sea en una ventana en blanco, o sobre imágenes o incluso sobre videos.

```
cvRectangle(img,cvPoint(100,100),cvPoint(200,200),cvScalar(255,0,0), 1)
```

Desplegar texto: Esta función sirve para desplegar texto en pantalla, ya sea sobre imágenes o sobre videos.

```
CvFont font;
double hScale=1.0;
double vScale=1.0;
int    lineWidth=1;
cvInitFont(&font,CV_FONT_HERSHEY_SIMPLEX|CV_FONT_ITALIC,
hScale,vScale,0,lineWidth);

cvPutText (img,"Mi texto en pantalla",cvPoint(200,400), &font, cvScalar(255,255,0));
```

Histograma de colores.

El histograma está hecho, para poder verificar que los tonos de piel sean similares en cada usuario, pues es necesario para este proyecto, que se siga una mano, entonces así nos podemos dar cuenta de la mezcla de colores que cada objeto en este caso la mano tienen. Este parte del programa sólo se utiliza en la parte de las pruebas, pues una vez que se puede hacer el reconocimiento y seguimiento de la mano, no por el color, sino por su forma, ya no es necesario, simplemente es un extra de la aplicación, para poder observar la mezcla de colores y tonos que cada usuario tiene en su piel.

Operaciones con histogramas:

- En OpenCV se define el tipo CvHistogram y las operaciones para manejarlo: cvCreateHist, cvReleaseHist, cvCalcHist, cvQueryHistValue, cvGetHistValue, cvNormalizeHist, cvThreshHist, cvGetMinMaxHistValue.
- Tenemos también una operación de ecualización del histograma: cvEqualizeHist.

Interfaz Gráfica

La interfaz gráfica se hizo con ayuda de DevC++ 4.9 y la biblioteca de OpenCV: ***HighGUI*.** La cual permite la creación de ventanas, la interacción de imágenes y video, y el despliegue de figuras y texto utilizando funciones que tienen esta función.
Dicha interfaz consta de una ventana donde se proyecta la captura de video en tiempo real, de la mano, y en la cual se marca la simulación de los movimientos del ratón, por medio del dibujo de figuras geométricas como círculos y cuadrados de distintos colores y también el despliegue de texto para saber de qué tipo de acción se trata. Ver *Figura 13.*

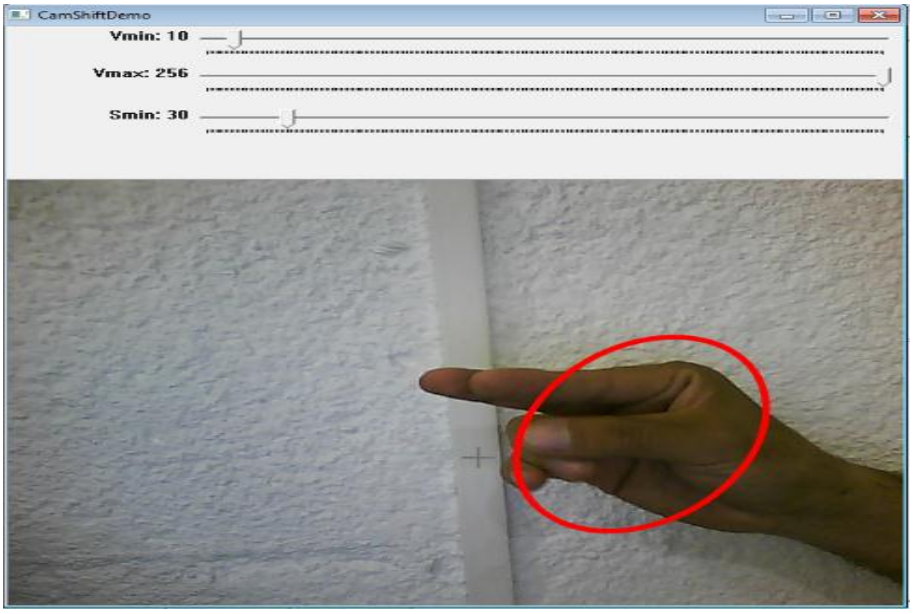

Figura 13: Interfaz Gráfica del proyecto.

Pruebas

En esta sección se describen las pruebas que se realizaron al sistema para verificar su funcionalidad.

Pruebas y resultados

Prueba	Resultado
Capturar Video	Se desplegara una ventana con el video de la mano. *Ver Figura 14.*

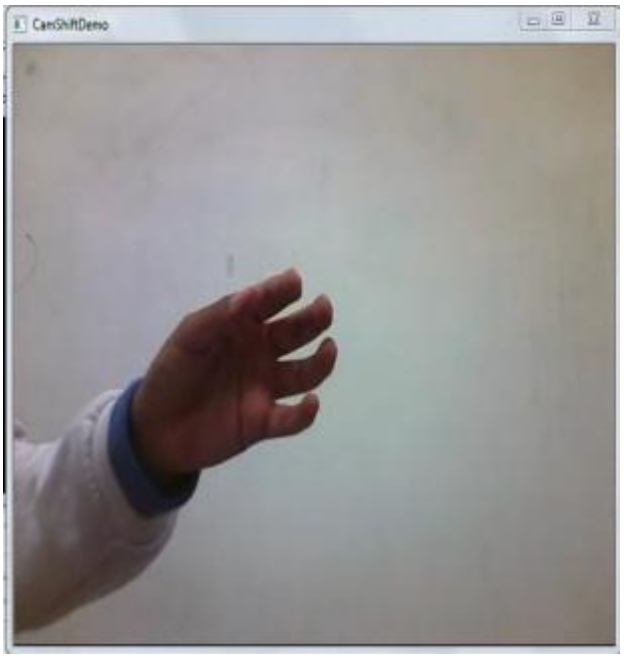

Figura 14: Prueba capturar video.

Prueba	Resultado
Desplegar el histograma de colores de la imagen	Se desplegara una ventana con un histograma con los colores de la imagen para poder diferenciar distintos tonos de piel. *Ver Figura 15*

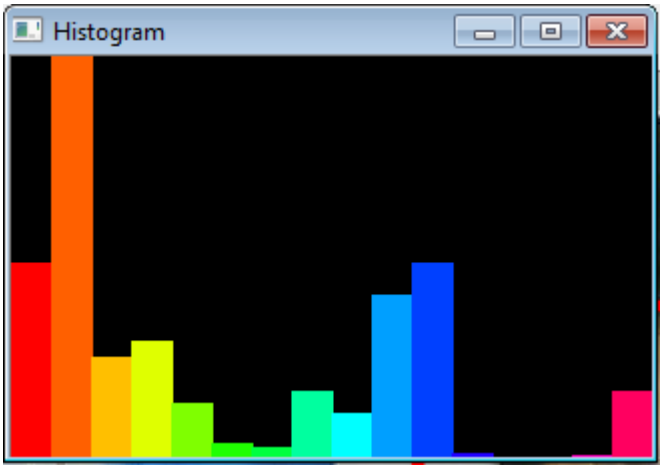

Figura 15: Prueba Histograma de colores.

Prueba	Resultado
Realizar la retroproyección de la imagen.	Se desplegara una ventana con el video en retroproyección para verificar que se trabaje con un modelo de una mano y no solo de los colores. *Ver Figura 16*

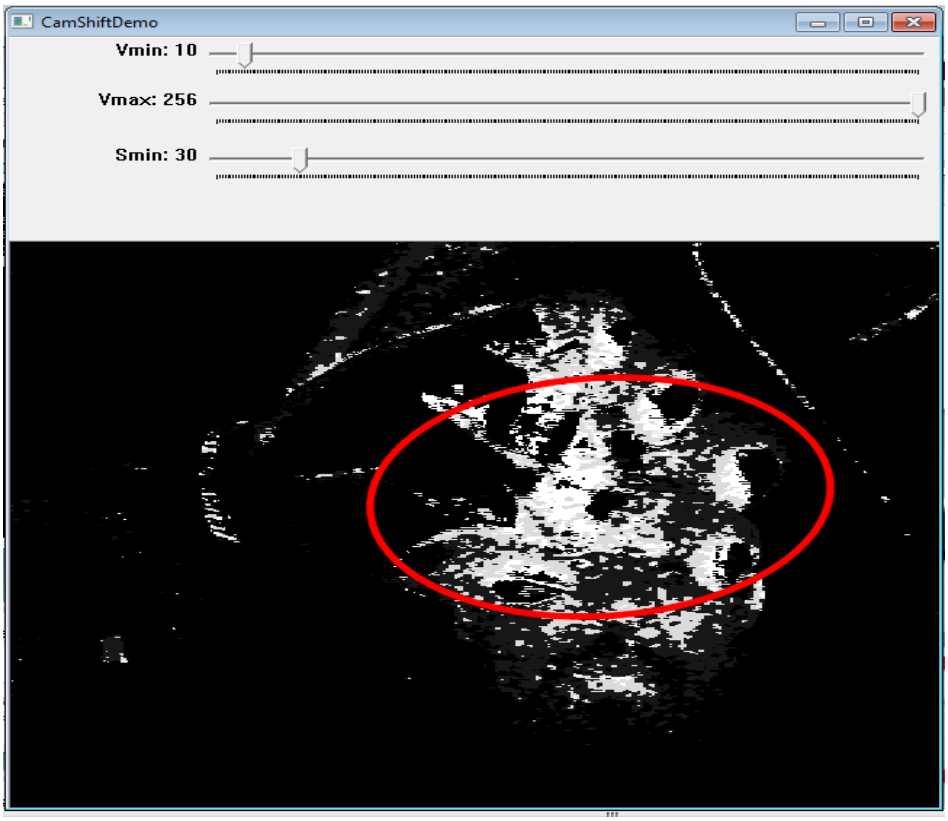

Figura 16: Prueba de retroproyección.

Prueba	Resultado
Simular el movimiento del ratón	En la ventana se desplegara un indicador (círculo rojo) que seguirá la mano y un letrero para así marcar el desplazamiento del ratón. *Ver Figura 17*

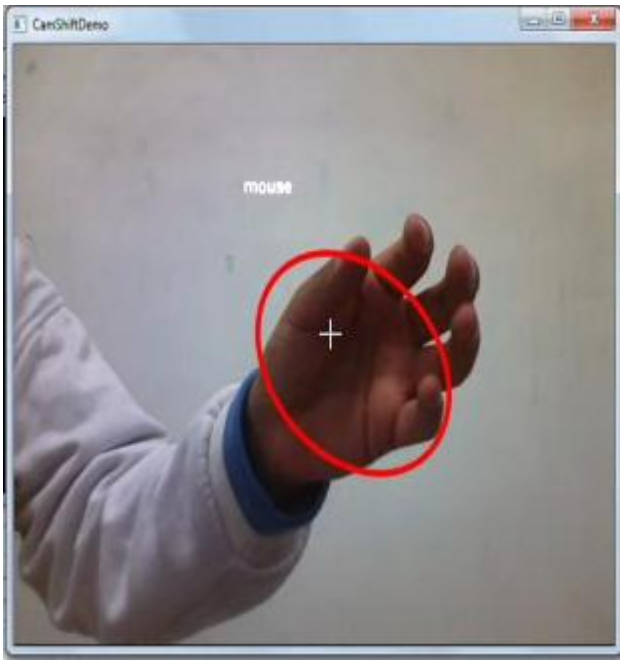

Figura 17: Prueba desplazamiento.

Prueba	Resultado
Simular el clic derecho	En la ventana se desplegara un indicador (cuadrado rosa) que distinguirá la posición de la mano y desplegara un letrero para así marcar el clic derecho del ratón. *Ver Figura 18*

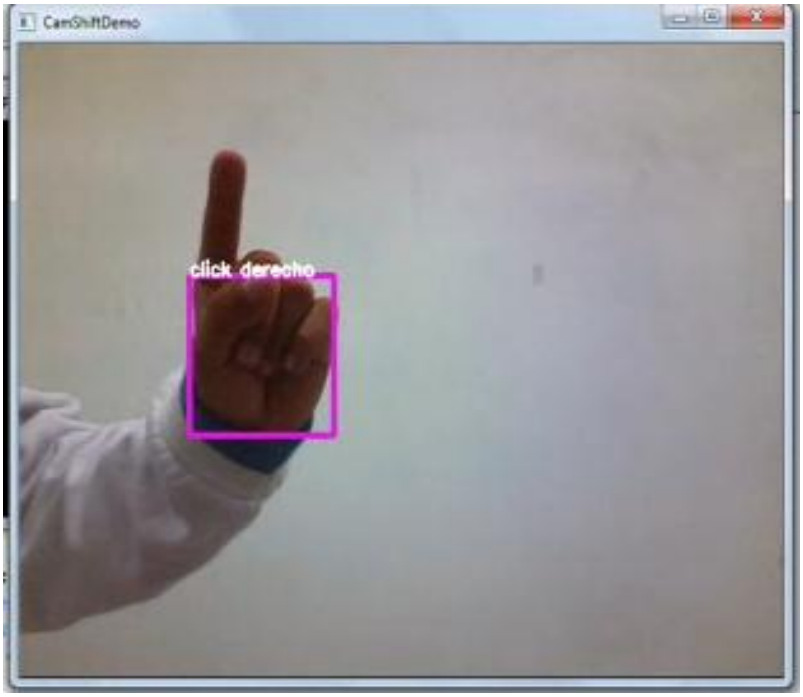

Figura 18: Prueba Clic derecho.

Prueba	Resultado
Simular el clic izquierdo	En la ventana se desplegara un indicador (círculo naranja) que distinguirá la posición de la mano y desplegara un letrero para así marcar el clic izquierdo del ratón. *Ver Figura 19*

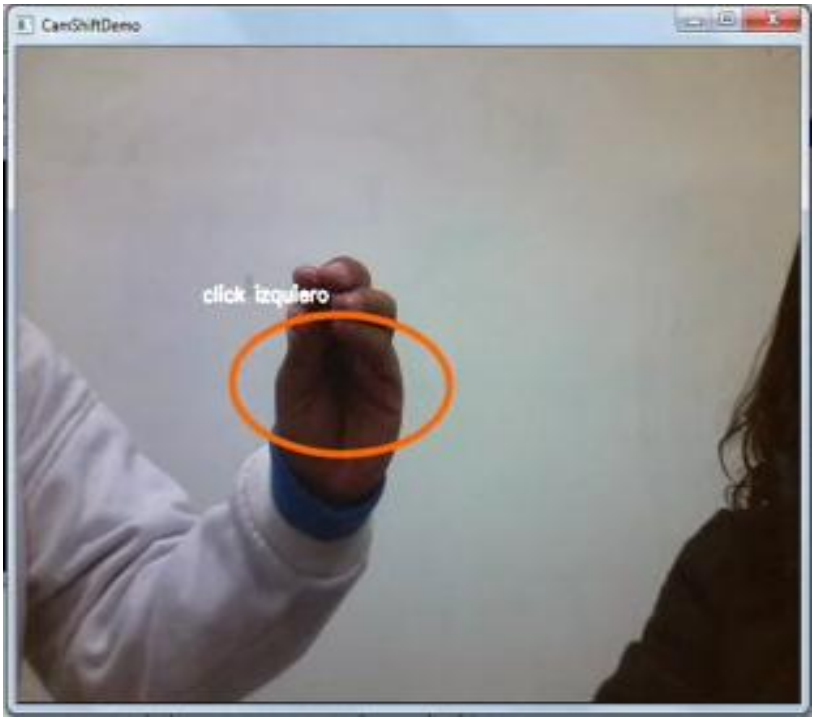

Figura 19: Prueba clic izquierdo.

En la *Figura 20* se presentan, algunos parámetros, resultados y observaciones, hechas a través de pruebas realizadas con tres personas distintas en su tono de piel; obteniendo los siguientes datos:

Persona	Tono de piel	Tiempo de detección (segundos)	Resultados	Observaciones
1	Claro	11	Se pudieron simular todas las acciones del ratón, tardando un poco más en detectar algunos movimientos.	Un tono de piel más claro, es más difícil de detectar, ya que la similitud con el fondo blanco hace muy lenta la diferenciación entre fondo y piel.
2	Moreno claro	8	Se simularon todas las acciones del ratón en un tiempo razonable.	Con un tono de piel que contrasta más con el fondo blanco es más fácil y rápido la detección de los movimientos de la mano.
3	Moreno obscuro	7	Se simularon las acciones del ratón en un tiempo razonable.	Entre más contraste exista con el fondo mejores serán los resultados.

Figura 20: Pruebas a tres tonos de piel distintos.

Conclusiones

En esta sección se presentan las conclusiones a las que se llegaron después del desarrollo del proyecto.

Conclusiones

En cuanto al diseño.

La parte del diseño fue la más fácil, pues cuando se tiene un idea en mente es fácil visualizarla y tratar de entenderla.

Para el diseño de esta aplicación, se buscó algo que fuese amigable y fácil de usar, algo que cumpliera con su función. En un principio se pensó en un despliegue de dos ventanas en pero finalmente se tomó la decisión de utilizar una ventana de despliegue de video, donde se pudieran ver los resultados del proyecto, debido a que la segunda ventana quitaría espacio en pantalla, y consumiría más memoria de la computadora, en general es el hecho de optimizar los recursos por lo que se decide trabajar solo sobre una ventana.

En si el diseño desde un principio, fue algo austero, de tal forma que permitiera enfocarse más profundamente en el reconocimiento de los patrones en el video para sí poder captar distintas posiciones de la mano, y vincularlas con la acciones del ratón.

En cuanto al desarrollo.

En esta parte se enfrentaron los primeros problemas, pues el hecho de trabajar con algo nuevo, como lo era la visión artificial y las bibliotecas de OpenCV, represento un reto importante pues en muy poco tiempo se tuvo que realizar labor de investigación y pruebas, para posteriormente pasar a la estructuración de nuestro código fuente que finalmente daría vida a nuestra aplicación. Realizando la tarea de investigación y documentación en libros internet y otros medios, se logró ir viendo las fortalezas y debilidades que estas herramientas tenían, pudiendo sacar provecho de las virtudes para poder realizar la aplicación a la cual este reporte hace referencia.

La primera parte de la captura del video no tuvo mayor inconveniente, pero al iniciar con la creación de los modelos geométricos de la mano, y el trabajar con pixeles por separado, no fue una tarea fácil, pues se tuvo que recurrir a modelos matemáticos, y algebra lineal para poder realizar dichos modelos y así conseguir el seguimiento (*tracking*) de la mano.

El segundo y no más fácil paso, fue poder diferenciar tres posiciones distintas del modelo de la mano, pues nuevamente se tuvo que recurrir a las matemáticas, y a funciones que en su momento fueron muy complejas de entender en su funcionamiento.

Una vez que se lograron los objetivos primordiales que eran el reconocimiento de las acciones del ratón, se decidió, que en vez de dos ventanas como se planteó en el diseño, fuera solo una, pues se pensó en que de nada serviría tener dos

ventanas, pues en una solo se desplegaría el video, entonces se decidió desplegar solo una ventana, con los resultados, y así optimizar el espacio en pantalla y la memoria de la computadora.

El uso de un entorno de desarrollo con el que antes se había trabajado como lo es DevC++ hizo un poco más fácil el desarrollo de este proyecto.

En cuanto al trabajo en general

Fue muy interesante y gratificante el realizar un proyecto en donde se aplican conocimientos adquiridos durante la carrera en diversas materias, pues se logró entender cómo es que se relacionan algunas materias que se creía que no iban de la mano, pero con la realización de este proyecto, logramos entender su relación, y cómo es que usando estos conocimientos en conjunto se pueden obtener resultados más fáciles de lograr; aunque es difícil al principio entender cómo ir uniendo todos estos conceptos y conocimientos que se manejan individualmente, pero ya en conjunto puede resultar confuso al inicio, pero una vez que se va avanzando, se va volviendo más sencillo.

Se considera que los objetivos planteados para el proyecto desde el mismo momento de la propuesta se alcanzaron durante el desarrollo, por lo que se puede asegurar que el proyecto finalizó de manera correcta.

Limitantes.

El hecho de trabajar con una cámara web convencional, causa algunos problemas, pues la resolución es muy baja lo que provoca que la realización, de modelos y el trabajo con pixeles individuales o cadenas de ellos sea muy difícil, pues cuando la aplicación tiene que distinguir manos de distintas personas, se enfrenta al problema del color de la piel, o de cicatrices, o forma de las manos, lo cual dificulta el buen desempeño de esta aplicación.

También respecto a la cámara es difícil, el uso de una webcam convencional, porque los frames por segundo que nos pueden brindar pueden variar, haciendo lenta la captura y el despliegue del video.

Trabajos futuros.

Se plantea una integración con un sistema operativo, para controlar el ratón e interactuar con las aplicaciones.

También se plantea poder trabajar con una cámara de alta definición. Para mejorar la calidad y el desempeño de la aplicación.

También se podría dejar de trabajar con cámaras y trabajar con sensores de movimiento y/u opto reflexivos que puedan distinguir las posiciones de la mano u otras partes del cuerpo.

Ventajas del proyecto respecto a otros dispositivos.

Entre las ventajas más significativas, se encuentra el hecho de no depender de ningún cable o dispositivo, más allá de una cámara web, que hoy en día casi cualquier computadora tiene una. El hecho de sentirte libre de solo usar tu cuerpo y los movimientos naturales que este tiene en las manos, hacen que sea sencilla de usar y de fácil acceso para cualquier persona, en comparación de guantes, o el mismo ratón de cualquier tipo que exista hoy en día.

Respecto al uso de OpenCv.

Usar OpenCv es fácil, una vez que se familiariza con la familia de funciones que este maneja. El hecho de tener modelos y funciones creadas, para poder trabajar con ellas es un plus que te facilita mucho el desarrollo de aplicaciones.

Todo lo anteriormente mencionado, aunado al hecho de que es software libre, y por ende cualquier persona con una computadora y acceso a internet puede hacer uso de las bibliotecas, y también el saber que se puede trabajar en cualquier plataforma, ya sea Windows, Linux, Mac OS, etcétera. Esto nos da la posibilidad de poder trabajar con una infinidad de ambientes de desarrollo pudiendo elegir el que más se acomode a las necesidades de cada usuario.

En general OpenCv, es accesible, y fácil de usar en cuestiones de seguridad, visión artificial y en proyectos de visión para robots, y me gusto trabajar con estas bibliotecas, gracias a la diversidad de funciones con las cuales se puede contar, ya sea con imágenes, o con video.

Bibliografía

[1] L. Marín. “Sistema de aprendizaje del alfabeto dactilógico mediante procesamiento de imágenes utilizando software libre” Propuesta de proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2009.

[2] *null-tarandeep* [En línea] Disponible: http://sites.google.com/site/tarandeep/fingermouse

[3] *2010, Camera Mouse*. [En línea] Disponible: http://cameramouse.org/

[4] Bradski Gary, Kaehler Adrian; **“***Learning OpenCV***”**; Publicado por O’Reilly Media, USA September 2008; ISBN: 978-0-596-51613-0

[5] Histogramas con OpenCV
http://opencv.willowgarage.com/documentation/python/imgproc_histograms.html

[6]Instalación de OpenCV
http://dis.um.es/~ginesgm/files/doc/pav/guion2.pdf

[7]Seguimiento de objetos
http://www.cimat.mx/~jbhayet/CLASES/VISIONROB/opencv1.pdf

[8]Manual de referencia en línea OpenCV
http://opencv.willowgarage.com/wiki/

[9]Interacción con el computador mediante el seguimiento de una mano
http://www.etsii.urjc.es/~jjpantrigo/PFCs/NavegacionEnMapas.pdf

[10]Introducción a la programación con OpenCv
http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html