ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

MOUSE PARA PERSONAS CON DISCAPACIDAD MOTRIZ

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y CONTROL

DINA PATRICIA JARRIN ZAMBRANO

DIRECTOR: Dr. ROBIN GERARDO ÁLVAREZ RUEDA

Quito, agosto 2009

DECLARACIÓN

Yo, Dina Patricia Jarrín Zambrano, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Dina Patricia Jarrín Zambrano

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Dina Patricia Jarrín Zambrano, bajo mi supervisión.

Dr. Robin Álvarez
DIRECTOR DE PROYECTO

Agradecimientos

Agradezco a Dios por bendecirme y darme sabiduría para alcanzar mi triunfo, a mi madre Merceditas que me supo apoyar desde que me tuvo en su ser, y formó mi personalidad con definición y seguridad.

A mi esposo Geoffrey que con su amor incondicional me acompaño en la culminación de mi carrera. A mi familia, Josué mi Padre, mis hermanas Consuelito, Dianita, Josesita, mis sobrinas Franshesquita y Malenita y mi cuñado Oswaldo, por su apoyo moral.

Quiero agradecer a mi universidad donde conocí personas inolvidables como mis amigos y profesores.

Dedicatoria

Dedico mi trabajo realizado a Dios y a todas las personas importantes en mi vida.
Y también a mi bebé que viene en camino.

Contenido

Índice de Figuras

RESUMEN

Planteamiento del problema: Por diferentes causas ya sea enfermedades o accidentes, las personas pierden o disminuyen sus capacidades motrices, lo que dificultará el manejo de ciertos dispositivos, como por ejemplo en informática, el tradicional mouse, abriéndose la necesidad de utilizar la tecnología adaptativa.

Posibles Soluciones:

- **Control a través de la voz.**
- **Control a través del movimiento de la cabeza.**
  - **Por medio de sensores:** El problema que se manifiesta es que es un sistema *invasivo*, es decir necesita la conexión de cables.
  - **Por medio de visión artificial:** ya que no se requiere de ningún tipo de conexión que involucre cables, por lo tanto, es un sistema *no invasivo.*

En esta ocasión, se ha decidido investigar este último caso: Controlar el movimiento del mouse por medio del movimiento de la cabeza, utilizando visión artificial.

Material y Métodos: Se utiliza una webcam y con el objetivo de evitar problemas de niveles de luz bajos se utiliza un objeto reflejante circular que facilite su detección mediante algoritmos desarrollados en Matlab.

Los algoritmos trataran el seguimiento del mouse y alternativas para la emulación de clic.

Resultados: Finalmente con el análisis de las diferentes alternativas de mouse se determinará si este método o no es apropiado para que una persona con discapacidad motriz pueda utilizarlo.

Con las pruebas que se realicen en deducción de los algoritmos se podrá definir las afinidades a nuestros objetivos, y con argumentos respectivos se expondrá los aciertos y errores de los mismos.

Posteriormente se aportará con conclusiones acorde al trabajo realizado y también recomendaciones para trabajos futuros.

PRESENTACION

El desarrollo del proyecto se realiza de la siguiente manera:

Capítulo 1 "***Introducción y estado del arte***", se presenta un resumen de lo que implica este proyecto, así como proyectos similares realizados de mouse para personas con discapacidad motriz. Se analizará posibles soluciones tales como faciales o con la ayuda de un objeto o dispositivo (objeto reflector) para controlar el puntero del mouse.

Capítulo 2 "***Desarrollo de algoritmos de reconocimiento del objeto emulador del puntero del mouse y algoritmos para emular el movimiento del puntero del mouse***", se implementan varios algoritmos que van desde la adquisición de imágenes hasta la selección del objeto reflector.

También se implementa un método de control del puntero del mouse para su posicionamiento en la pantalla del computador.

Capítulo 3 "***Implementación de los algoritmos***" se implementan los algoritmos y métodos mencionados en el capítulo anterior, que se realizan en tiempo diferido.

Capítulo 4 "**Pruebas y Resultados en tiempo diferido y en tiempo real**" una vez concluidos las respectivas pruebas en tiempo diferido, se implementan en tiempo real para la respectiva evaluación del proyecto.

Capítulo 5 "***Conclusiones y recomendaciones",*** *s*e resaltan los resultados más importantes que se obtuvieron.

- *Referencias Bibliográficas.*
- *Anexos.*

CAPITULO I

INTRODUCCION Y ESTADO DEL ARTE

1. INTRODUCCION

A través del tiempo se ha visto el aislamiento que han sido sometidas las personas discapacitadas, sin permitir que sean partícipes en desarrollar sus capacidades intelectuales en beneficio de la sociedad, debido a sus limitantes, por esta razón es que varias instituciones privadas o públicas han contribuido con los avances tecnológicos que se han dado y se siguen dando, con el propósito de mejorar el estilo de vida de estas personas.

Tomando en cuenta todas las barreras que tienen que vencer las personas con discapacidad motriz, se desarrollará este proyecto con tecnología adaptativa, como es la manipulación del puntero del mouse, con el fin de que tengan una interacción con los sistemas computacionales y además su uso sea óptimo y cómodo.

1.1 PLANTEAMIENTO DEL PROBLEMA

Debido a la imposibilidad de manejar la tecnología para personas con algún tipo de discapacidad motriz, es necesaria la utilización de la tecnología adaptativa.

Especialmente estos estudios han sido dedicados a jóvenes que presentan algún tipo de discapacidad, ya que las personas de tercera edad usualmente se refugian con sus familiares y el desarrollo social que ellos muestran es mínimo, comparado con las personas jóvenes, pero por la poca facilidad de comunicación tecnológica su desarrollo social también se ha visto reducido.

En los estudios del doctor Pedro Jurado de los Santos, del grupo CIFO miembro del departamento de Pedagogía Aplicada de la Universidad Autónoma de Barcelona (UAB), indica lo siguiente:

Para que, en lugar de ser una carga social, pase a formar parte de la fuerza de trabajo, y así la persona discapacitada solo requiera de tecnología adaptativa,

que le ayude a desarrollar o adquirir habilidades que le den oportunidades para efectuar su labor personal y/o profesional de mejor calidad a la esperada.

Para poder ayudar a sobresalir al discapacitado en este proceso se necesitan de varios aspectos globales como son los aspectos psicológicos, entre los cuales están: motivación constante, recuperación de la confianza en sí mismo, etc., y también aspectos tecnológicos tales como dispositivos que faciliten cualquier labor que el discapacitado desee desempeñar, educación necesaria con la cual adquiera suficiente conocimiento y manipulación de la herramienta a utilizar, etc. y es allí donde el desarrollo de tecnología para personas con discapacidades pasa a ser un gran aporte para una alternativa de vida de calidad y un desempeño eficaz.

Es por ello que muchas áreas científicas como son: Electrónica, Robótica, Biónica, Bioingeniería, Informática, Mecánica, y similares, se han asociado con el fin de desarrollar tecnologías adaptadas para facilitar la vida del discapacitado en especial del discapacitado joven, como lo anuncia la Agencia Europea para el desarrollo de la Educación Especial en el 2002, en el Informe “Transición de la escuela al empleo”.

Para establecer el ámbito en este trabajo se aportará al desarrollo de una tecnología adaptativa para el control del mouse de un computador.

Se analizará el estado del arte de los métodos publicados para la evaluación del mouse, analizando las respectivas características para afirmar el método elegido para la realización de este proyecto.

1.2 Conceptos asociados a este proyecto.

1.2.1 Esclerosis múltiple (EM).- es una enfermedad del sistema nervioso que afecta al cerebro y la médula espinal. Lesiona la vaina de mielina, el material que rodea y protege las células nerviosas. La lesión hace más lentos o bloquea los mensajes entre el cerebro y el cuerpo, conduciendo a los síntomas de la EM (Figura1.2.1).

Figura1.2. 1 La esclerosis múltiple (Foto Prensa Libre)

1.2.2 Atrofia Muscular.- La atrofia muscular (Figura1.2.2) es la disminución de masa muscular y el desgaste de los tejidos musculares. Los músculos que pierden inervación pueden atrofiarse o simplemente dañarse.

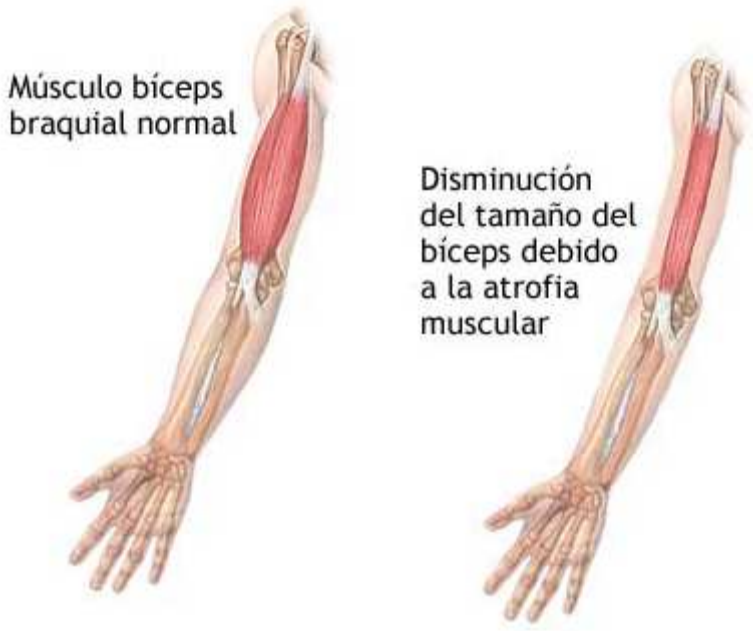


Figura1.2. 2 Atrofia Muscular

1.2.3 Sistema de Reconocimiento Facial.- es mediante la comparación de determinados rasgos faciales de la imagen facial y una base de datos, mediante la comparación de determinadas características faciales a partir de una imagen digital o un fotograma de una fuente de vídeo.

1.2.4 Método por luz Infrarroja

Esta tecnología está basada en rayos luminosos que se mueven en el espectro infrarrojo. Se encuentra en muchos ordenadores portátiles, y en diferentes tipos de teléfonos celulares, aunque en la actualidad esta tecnología ha sido reemplazada por la de bluetooth.

Desarrollos en los sistemas de visión artificial para la identificación y seguridad, realiza un análisis en factores como:

- Calidad de los objetos a inspeccionar y de su entorno.
- Condiciones de iluminación exterior y posibles interferencias puntuales.
- Condiciones ambientales.
- Nivel de preparación técnica para la instalación y configuración.
- Nivel de conocimientos en tecnologías de información del usuario/explotador.
- Capacidad del sistema de inspección para la integración en entornos de comunicación, redes y bases de datos.
- Integración en sistemas de control de accesos ya existentes.
  Absorción de luz infrarroja

Una de las desventajas de este sistema es que tiene alto número de falsas alarmas.

1.3 Estado del Arte

Desde el año 1994 se han realizado estudios sobre este tema, y se han desarrollado equipos equivalentes al Mouse que han servido de gran ayuda para discapacitados que aún tengan cierta movilidad motriz, desafortunadamente debido a su valor económico han sido para algunas personas inalcanzable.

En junio del 2007 iniciaron prototipos de ratones faciales que han ayudado a este tipo de discapacidad. A continuación resumimos los trabajos similares realizados:

1.3.1 Camera Mouse

Este Mouse se dirige a discapacitados sin movilidad motriz, por ello se utilizará o aprovechará los movimientos de la cabeza. Este trabajo fue realizado por la Universidad “Boston College”, su proyecto ha sido dedicado a personas con control muscular voluntario muy limitado o que tengan solo el manejo del movimiento de la cabeza.

Utilizan una cámara web con conexión USB que está ubicada en la parte superior o inferior del computador, el cual enfoca el rostro del usuario haciendo de puntero la nariz del usuario, o el borde de las cejas de la imagen captada, el control del mouse se dará por el movimiento de la cabeza. Para la ejecución del clic, el usuario debe mantener la posición de la cabeza por cierto tiempo y así podrá efectuar el clic derecho o el izquierdo. Esta aplicación trabaja con cualquier Sistema Operativo como Windows Vista o XP OS y una cámara web de alta resolución.

Un servicio extra que ofrece esta institución es entregar licencias a head mouse vendidos, así el usuario cuenta con el respaldo y garantía que da la prestación de este servicio. En la Figura1.1 se puede ver un ejemplo de aplicación de este sistema. **[1]**

Figura1.3. 1 Zubair, estudiante del colegio “Holly Bank School” usando el sistema de la cámara web. [2]

Una de las ventajas más representativas es que este sistema de emulación de mouse es óptimo debido a la cámara web de alta resolución que este método ofrece.

Este sistema se ve gravemente afectado por las condiciones de luz, siendo su rendimiento muy bajo en situación de baja iluminación.

1.3.2 Emulador de Mouse

Emulador de Mouse para personas con alteraciones motoras en sus manos realizado en la escuela ORT de Argentina con el apoyo de Centro de Investigación y Desarrollo de Asistencias Tecnológicas para la Discapacidad del Instituto Nacional de Tecnología Industrial (INTI).

Utilizan una cámara web ubicada en la cabeza del usuario, sujetada con una bincha, gorro o casco, para que los movimientos de la cabeza sean reconocidos por un emisor infrarrojo ubicado sobre el monitor, realizando el movimiento brusco horizontal como negación y movimiento brusco vertical como afirmación. El clic izquierdo, derecho y doble clic se configuran en el software y con el mismo dispositivo.

A continuación se muestra en la Figura1.2 esta aplicación. **[3]**

Figura1.3. 2 Mouse para personas con alteraciones motoras en las manos.

La ventaja de este sistema es que por los dispositivos utilizados es más económico y el sistema de emulación de mouse se hace más aceptable, hablando económicamente.

La fuente que muestra este proyecto no entrega mucha información por lo que es difícil denotar desventajas del mismo.

1.3.3 HeadMouse1

Realizado por Grupo de Robótica - Universitat de Lleida. Utilizan una cámara web ubicada en la parte superior del computador, y por medio del movimiento de la cabeza conserva el control del puntero visto en la Figura1.3.

Figura1.3. 3Mouse con movimientos de la cabeza.

Las diferentes opciones de clic lo realiza mediante los ojos (pestañeando), con el borde de las cejas (alzándolas), la boca (abriéndola) y finalmente con un temporizador de 1.5 segundos para el auto-clic, percibido en la Figura1.4.

Ojos **Cejas** **Boca** **Auto-clic**

Figura1.3. 4Opciones para realizar el clic.

A continuación se mostrará en la Figura1.5 un menú de diferentes decisiones para un óptimo control del puntero del mouse, el cual ofrece variación de decisión para el puntero del headmouse. **[4]**

Figura1.3. 5Menú de opciones de aplicaciones.

Este sistema tiene una gran ventaja con respecto a los mostrados anteriormente, y es que a este sistema de Headmouse1 se lo puede encontrar gratuito en el internet. **[4]**

Aunque la ventaja que se ha descrito es muy aceptable, aún prevalece la desventaja de que este sistema tiene alta dependencia de la luz, por que el método utilizado es por detección facial.

Para una mejor experimento de este proyecto, se procedió a instalar y a ver su respectivo funcionamiento, y se concluyó que otra desventaja fue un mal seguimiento del puntero del mouse, ya que al subir la cabeza subía el puntero del mouse y al hacer el siguiente movimiento de subir la cabeza, la reacción del emulador del mouse fue bajar y volver a subir, entonces se concluye que su desplazamiento es pobre.

1.3.4 Hands-Free Cursor Control

Este trabajo fue realizado por SmartNav Natural de Point Company Sites. Consta de una cámara web infrarroja (IR) (visualizada en la Figura 1.3.7), con un valor de 500 dólares, la cual se coloca en la parte superior del monitor o computadora portátil o en cualquier lugar que se logre la reflexión con un accesorio creado por SmartNav, como puede ser la utilización de Gafas, Microphone Boom o una gorra ($20) con una luz especial como se ve en la Figura1.3.6 **[5]**.

Figura1.3. 6SmartNav Hat [6]

En la Figura1.3.7 muestra la cámara web infrarroja a utilizar, la cual envía instrucciones de control al puntero del mouse, con una alta resolución de 1280 x 480 lo que permite sentir un mejor control del puntero del mouse.

Figura1.3. 7Cámara Infrarroja IR [7]

La base del control del puntero del mouse se da mediante los movimientos de la cabeza del usuario, con algún dispositivo ya nombrado, así la luz Infrarroja refleja en el producto de SmartNav y este envía instrucciones al computador para realizar los movimientos del mouse.

Para la ejecución del clic derecho y del clic izquierdo SmartNav ofrece varias opciones, tales como:

Hotkey: mapa de su teclado y los asigna para emular la Izquierda, el Derecho y botones Medianos de ratón.

La ventaja de este proyecto es el funcionamiento óptimo que tiene este sistema ante el desarrollo normal, ya que la detección se la hace por medio de una luz infrarroja, es decir, su dependencia de la luz no se nota, y se podría decir que con respecto a funcionamiento ante los demás proyecto se podría decir que es el más avanzado.

Sin el uso de los dispositivos extras que ofrece la misma compañía, se estaría hablando de un seguimiento facial, y por ende alta dependencia de la luz.

1.4 Nuestra Propuesta:

MOUSE PARA PERSONAS CON DISCAPACIDAD MOTRIZ BASADO EN OBJETO REFLECTOR

Haciendo un breve análisis de los problemas de iluminación que se dan en otros proyectos similares, como el de detección facial, se determinó que el uso del objeto reflector servirá para minimizar este inconveniente.

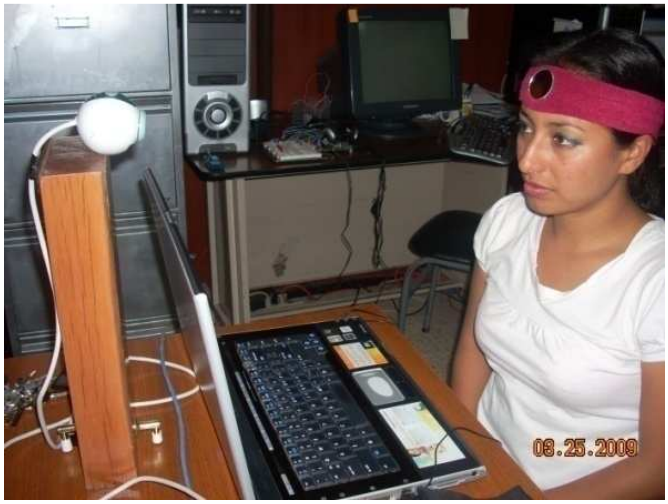


Figura1.4. 1Mouse para personas con discapacidad motriz

Los sistemas que se manejan por detección o reconocimiento facial tienen un alto nivel de dependencia de la luz. Por lo anterior se ha seleccionado el uso del objeto reflector circular (figura 1.4.2) que facilitará su detección aún en ambientes de reducido nivel de luz.

Figura1.4. 2Objeto Reflector circular

El objeto reflector está dado por un pedazo de espejo de forma circular, sujeto a un cintillo para facilitar la fijación del elemento en la frente del usuario (figura1.4.2). Para la detección de dicho objeto se utiliza una cámara web de bajo costo (figura 1.4.3). El hecho de que sea un material reflector y redondo facilitará su detección por medio de los respectivos algoritmos de procesamiento de imágenes, por medio de las herramientas que ofrece Matlab.

Figura1.4. 3 Cámara web Logitech, Resolución 352X288

1.5 ELEMENTOS DEL SISTEMA:

Hardware:

- Cámara Web (de características básicas).
- Un objeto reflector.
- Un computador.

Software:

- Software desarrollado.
- Sistema Operativo XP o Windows Vista.

1.5 Selección de hardware

1.5.1 Selección de cámara web

A continuación se hará un listado de cámaras web con el fin de seleccionar el dispositivo que más se acerque a los requerimientos de este proyecto.

Se procede a mostrar una reseña de características de diferentes cámaras web (figura 1.5.1).

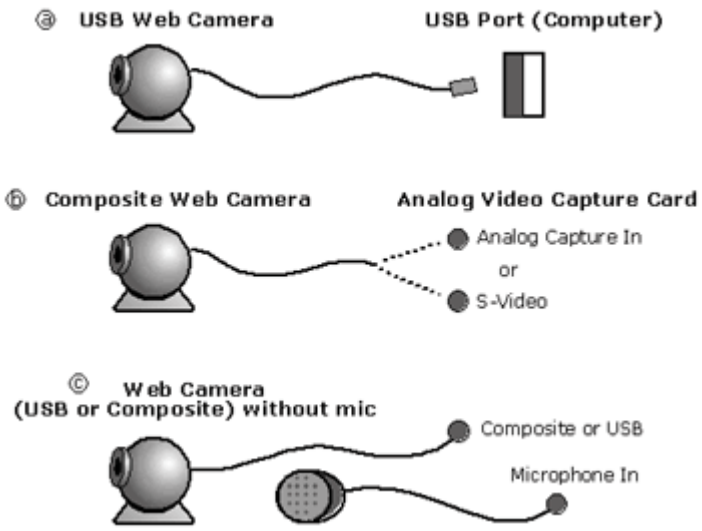


Figura 1.5. 1 Cámaras web con diferentes conectores al computador.

1.5.1.1 Cámara web (Quick cam) Logitech

Figura 1.5. 2Cámara web (Quick cam) Logitech

Características:

- Resolución: VGA (640 x 480).
- Tecnología RightLight™: Produce automáticamente imágenes más nítidas incluso en condiciones de iluminación escasa.
- Tecnología RightSound™: Garantiza conversaciones con sonido nítido sin molestos ruidos de fondo.
- Eficacia

- Videoconferencias Plug and Play: Permite realizar videoconferencias sin tener que instalar software. Basta conectar la cámara Web para empezar a usarla.
- Clip para monitor universal: Se acopla firmemente a todo tipo de monitores, ya sean de computadoras portátiles o de sobremesa, y puede colocarse sobre superficies horizontales.
- Compatibilidad: Funciona con la mayoría de los programas de mensajería instantánea.

1.5.1.2 Microsoft LifeCam VX6000

Figura 1.5. 3Microsoft LifeCam VX6000

Características:

- Tipo del dispositivo: Web camera – fixed
- Dimensiones: (WxDxH): 5.5 cm x 5.3 cm x 7.4 cm
- Peso: 95 g
- Tipo de sensor Óptico: CMOS - 1,300.000 pixels (1280 x 1024)
- Ajuste de foco: Manual
- Interface: Hi-Speed USB
- Soporte de audio: incluye micrófono.
- Tamaño de cable: 1 x USB cable - integrado - 1.8 m
- Requerimientos del Sistema: Microsoft Windows XP Home Edition, Microsoft Windows XP Professional, Microsoft Windows XP Media Center Edition, Microsoft Windows XP Professional x64 Edition, Microsoft Windows Vista.
- Garantía: 3 años
- Precio: $ 77

1.5.1.3 Logitech QuickCam Sphere

Figura 1.5. 4 Logitech QuickCam Sphere

Características:

- Tipo de dispositivo: Web camera - pan / tilt
- Cámara: Colour
- Sensor óptico: - 2,000,000 pixels
- Lente Focus: 3.7 mm
- Ajuste de focus: Automático.
- Iris del lente : F/2.0
- Rango de cobertura horizontal en grados: 189
- Rango de cobertura vertical en grados : 102
- Soporte de Audio: incluye micrófono.
- Interfaz: Alta velocidad USB.
- Tamaño de cable: 1 x USB cable - integrado - 1.8 m
- Requerimientos del Sistema: Microsoft Windows Vista / XP.
- Incluye soporte
- Garantía: 2 años
- Precio 134

La selección de la cámara se dio en bases económicas, como es la Cámara web (Quick cam) Logitech que cumple con las especificaciones básicas descritas anteriormente.

1.5.2 Selección del computador

El proyecto se ejecutará en tiempo real, y para esto, se necesita una computadora que cumpla las siguientes características mínimas que necesita un computador:

- Procesador: Intel Pentium 4.
- Memoria RAM: 512 MB.

1.5.3 Selección del Software

MATLAB

MATLAB (Laboratorio de Matrices), es un lenguaje computacional por medio de código de manera sencilla, que por sus herramientas (Toolboxes) y funciones tiene una alta gama de aplicaciones en la industria, ya que MATLAB es un software amigable en comparación a otros lenguajes de programación.

Las áreas en donde los toolboxes están disponibles incluyen adquisición de datos, procesamiento de señales, diseño de sistemas de control, la simulación de sistemas dinámicos, la identificación de sistemas, redes neuronales, etc.

MATLAB es la fundación numérica y gráfica para todos los productos de The MathWorks. MATLAB combina computación numérica, gráficos 2D y 3D y capacidades de lenguaje en un único ambiente fácil de usar. MATLAB en el entorno de desarrollo nos permite resolver problemas de cálculo complejo y es así como en el cálculo matricial y vectorial se puede hacer buen uso de MATLAB.

1.5.3.1 Características

- Vectores y Matrices 2D.
- Arreglos Multidimensionales.
- Estructuras.
- Objetos definidos por el usuario (sobrecarga, herencia).
- Datos de tipo múltiple.
- Texto y escalares.
- Enteros de 8, 16 y 32 bits (compatible con la lectura, escritura y presentación de imágenes).

1.5.3.2 Ventajas

- Amplio soporte matemático
- Alta precisión
- Amplio soporte de funciones ya desarrolladas
- Rápido prototipado.
- Integración con dispositivos hardware
- Una comunidad muy extendida
- Magnífica ayuda
- Comercial

1.5.3.3 Desventajas

- Gestión “oscura” de la memoria
- Problemas eventuales de velocidad
- Comercial
- Distribución de ejecutables

1.6 Funcionamiento:

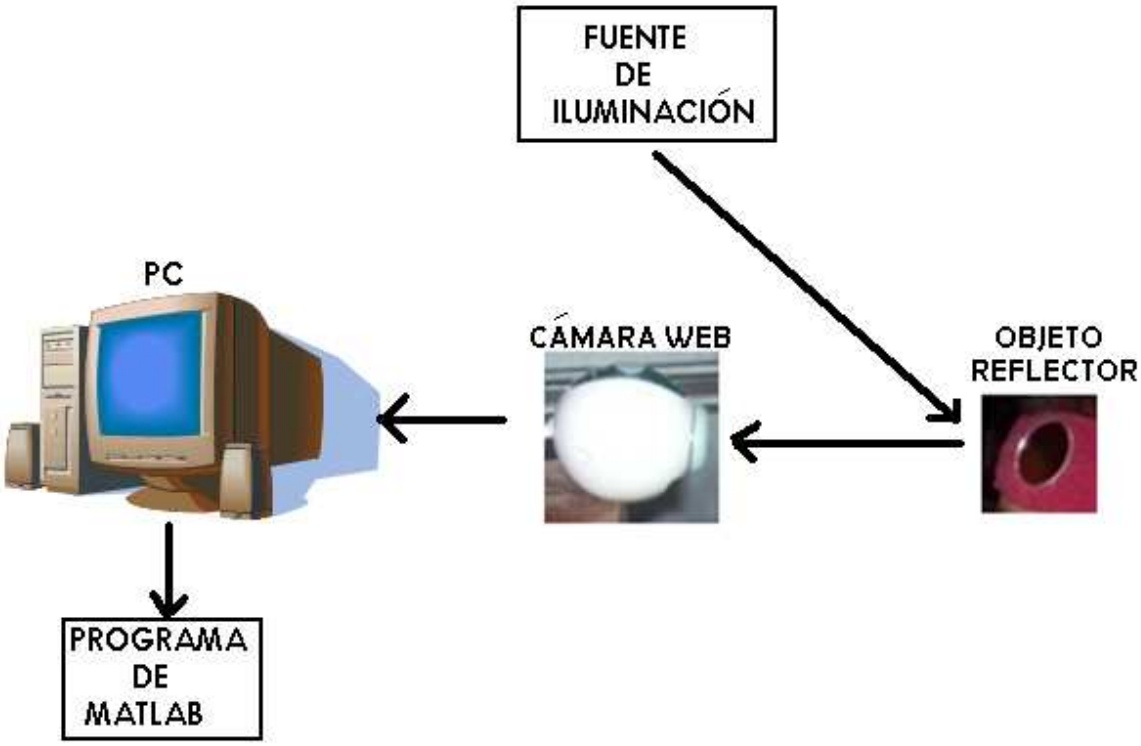


Figura1.6. 1 Descripción gráfica del funcionamiento del mouse para personas con discapacidad motriz.

La cámara web detecta el haz de luz que refleja el pedazo de espejo de forma circular, y así esta información es transmitida por datos de imagen los cuales por medio de un algoritmo se procede a interpretarlos y hacer coincidir el movimiento de la cabeza del mouse con el puntero del mouse propio del computador.

1.7 Objetivos del Estudio

1.7.1 Objetivos Generales:

- Diseñar software en tiempo real que contenga algoritmos para poder "**Crear un mouse que de accesibilidad al uso del computador a personas con discapacidad motriz"** y así aportar a que el discapacitado tenga un acercamiento de la tecnología.

1.7.2 Objetivos Específicos:

- Analizar el Estado del Arte de los tipos de control del puntero del mouse que se encuentran en el mercado y en investigación.
- Lograr que con el uso del software Matlab, una cámara web y un objeto reflector se pueda emular la misma operación de un mouse de uso común.
- Desarrollo de los algoritmos en Matlab.
- Prueba de los algoritmos realizados.
- Prueba de funcionamiento en diferentes ambientes, para encontrar diferentes limitaciones.

CAPITULO II

DESARROLLO DE ALGORITMOS DE RECONOCIMIENTO DEL OBJETO EMULADOR DEL PUNTERO DEL MOUSE Y ALGORITMOS PARA EMULAR EL MOVIMIENTO DEL PUNTERO DEL MOUSE

En este capítulo se hará revisión de los términos y conceptos que se emplean en el trabajo relacionados con adquisición de imágenes, segmentación.

Se precederá al desarrollo **"Desarrollo De Algoritmos De Reconocimiento Del Objeto Emulador Del Puntero Del Mouse Y Algoritmos Para Emular El Movimiento Del Puntero Del Mouse".**

2.1 MATLAB desde el punto de vista de procesamiento digital de imágenes.

El toolbox de procesamiento de imágenes contiene un conjunto de funciones de los algoritmos más conocidos para trabajar con imágenes binarias, trasformaciones geométricas, morfología y manipulación de color que junto con las funciones ya integradas en matlab permite realizar análisis y trasformaciones de imágenes en el dominio de la frecuencia (trasformada de Fourier y Wavlets).

El Toolbox de adquisición de imágenes es una colección de funciones que sirven, entre otras cosas, para adquirir imágenes de diversos dispositivos (USB webcams), para visualizar videos en vivo, para la adquisición de imágenes mediante triggers, para importar datos hacia el entorno de trabajo de MATLAB y a su vez nos permitirá efectuar la aplicación en tiempo real.

El Toolbox de Procesamiento de Imágenes proporciona a MATLAB un conjunto de funciones que amplía las capacidades del producto para realizar desarrollo de aplicaciones y de nuevos algoritmos en el campo del proceso y análisis de imágenes. El entorno matemático y de creación de MATLAB es ideal para el procesado de imágenes, ya que estas imágenes son, al fin y al cabo, matrices. Este toolbox incorpora funciones para:

- Diseño de filtros.
- Funciones para la extracción de bordes.
- Imágenes binarias y segmentación por umbral.
- Operaciones morfológicas.
- Operaciones basadas en objetos.
- Selección de objetos.
- Medición de características.
- Funciones para la conversión de imágenes y formatos de color
- Mejora y retocado de imágenes.
- Análisis y estadística de imágenes.

2.2 DESARROLLO DE ALGORITMOS DE RECONOCIMIENTO DEL OBJETO EMULADOR DEL PUNTERO DEL MOUSE.

Ya instalado todos nuestros elementos para poder comenzar con el desarrollo de este proyecto se procede a abrir el MATLAB con la dirección vista hacia la carpeta work propia de Matlab, para evitar conflictos de direccionamiento. Se activa la cámara web para tomar muestras y así comenzar con la programa en el software ya mencionado.

Esta muestra tomada (figura 2.2.1) tiene la misma resolución en pixeles que la cámara web provee (mxn = 352X288), donde m representa el largo y n representa el ancho.

La imagen tomada nos servirá de patrón, ya que servirá para obtener las características y hacer diferentes pruebas. Este proceso se hace de forma supervisada, es decir, se encarga de controlar como se produce la captura de las imágenes y si cumple con los requisitos para la evolución del sistema. Además, se aprovecha esta etapa para aclarar todas las dudas que se pudiera tener en el trayecto de este sistema.

La imagen será a color RGB (R=rojo, G=verde y B=azul), esta imagen se puede guardar con la extensión jpg, jpeg o bmp, con ello la imagen contenida en el archivo imagen con la extensión quedará contenida en la variable en matlab y así poderla utilizarla con las funciones para procesar la imagen.

Figura 2.2. 1 Adquisición de imagen

En matlab una imagen a escala de grises es representada por medio de una matriz bidimensional de m x n igual que la imagen RGB. Entonces la imagen en colores RGB se convierte a escala de grises para facilitar la aplicación de los algoritmos posteriores (figura2.2.2).

Figura 2.2. 2 Imagen en escala de Grises

La imagen de muestra RGB, puede ser de precisión uint8, uint16, solo, o doble, que son datos que pueden variar de 0 a 255, sin tolerar decimales o valores que salgan del rango ya mencionado. Al ser convertida a imagen en escala de grises,

esta es de la misma clase como la imagen RGB. Es por ello que para asegurar mejores resultados, convertimos a la imagen a precisión doble, como se puede ver en la figura2.2.3.

Figura 2.2. 3Imagen doble

El filtraje espacial (figura 2.2.4) es una de las operaciones comunes en procesamiento de imágenes ya sea para realizar efectos de eliminación de ruido o bien detección de bordes. En ambos casos la determinación de los píxeles de la nueva imagen depende del píxel de la imagen original y sus vecinos. De esta forma es necesario configurar una matriz (mascara o ventana) que considere cuales vecinos y en qué forma influirán en la determinación de el nuevo píxel.

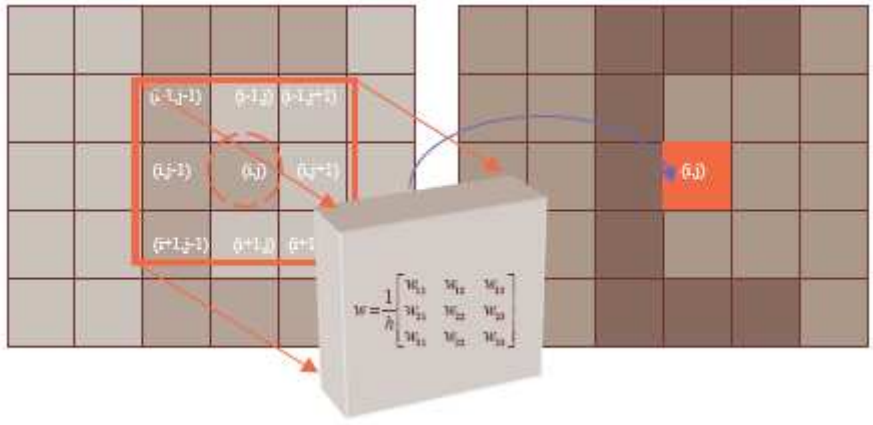


Figura 2.2. 4 Filtrado espacial con una máscara de 3x3.

Para el procesamiento de imágenes es necesario hacer reconocimiento de objetos o segmentación, extraer los bordes de objetos (que en teoría delimitan sus tamaños y regiones).

Es por ello que sobre esta imagen aplicamos un algoritmo de detección de bordes, mediante el algoritmo de Prewitt (figura 2.2.5), el cual se detallará con más claridad en capitulo 3.

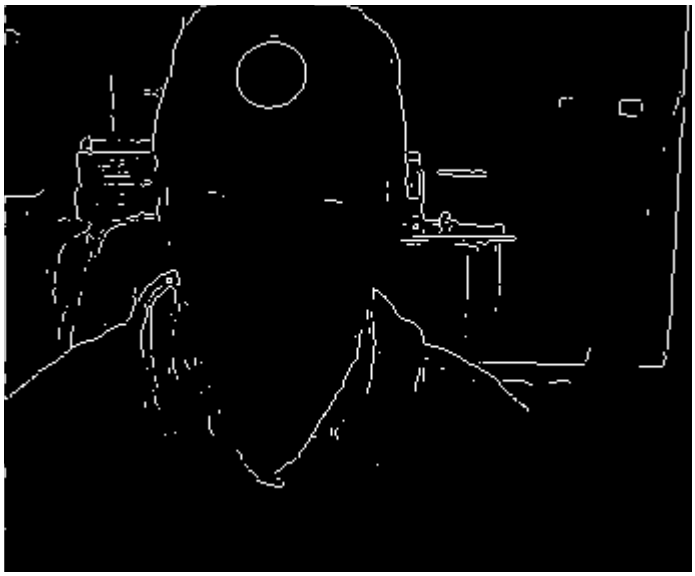


Figura 2.2. 5 Detección de bordes mediante algoritmo de PREWIT

A continuación se procede a filtrar la imagen y a escoger el mejor método con el cual se logre un buen reconocimiento de nuestro objeto. De la misma manera se explicará estos diferentes métodos de filtración en el capítulo 3. En nuestro caso a la imagen resultante se le aplica una herramienta de eliminación de objetos inferiores a 80 pixeles (figura 2.2.6).

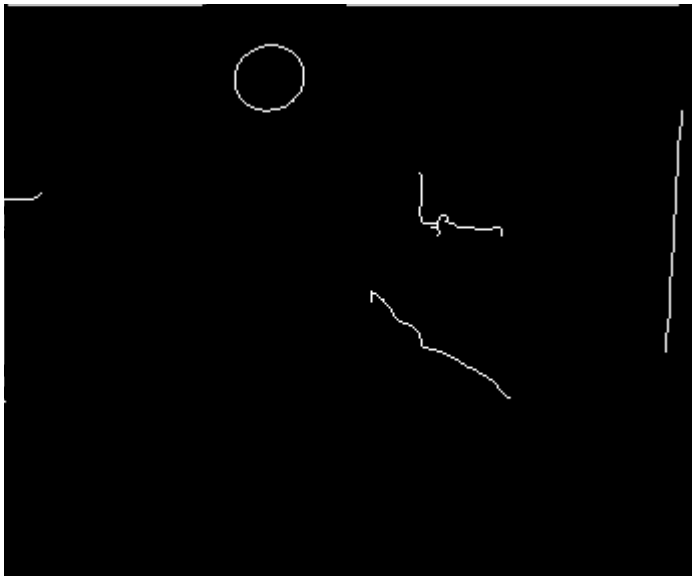


Figura 2.2. 6 Imagen filtrada.

Las operaciones morfológicas son operaciones realizadas sobre imágenes binarias basadas en formas. Estas operaciones toman como entrada una imagen binaria regresando como resultado una imagen también binaria. Se analiza por objetos los cuales serán tomados en cuenta para la determinación del píxel resultado. Cada objeto es un arreglo cuadricular que contiene unos y ceros, en los lugares que contiene unos serán los vecinos de la imagen original, los cuales serán tomados en consideración para determinar el píxel de la imagen resultado, mientras que los lugares que tengan ceros no serán tomados en cuenta.

Mediante una operación morfológica cerrada (figura 2.2.7), se rellena los objetos cerrados de la imagen anterior, cuyas dichas herramientas se explicará en capítulo 3.

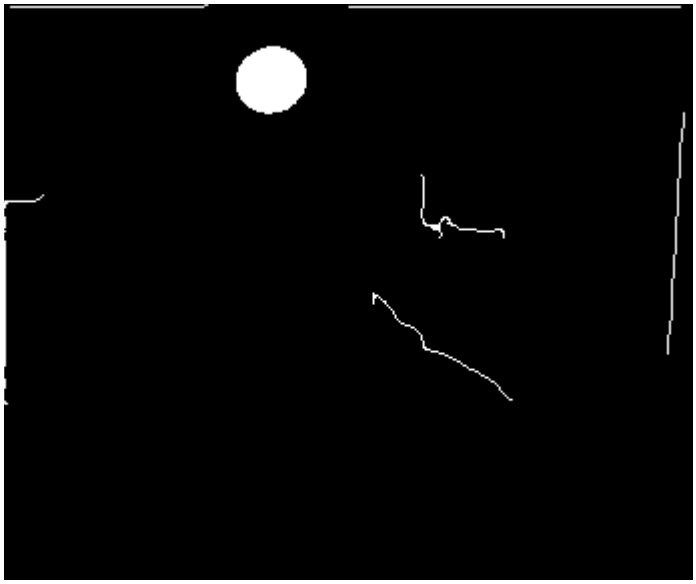


Figura 2.2. 7 Imagen con operación morfológica cerrada.

A partir de este resultado, procedemos a realizar segmentación para poder reconocer objetos y cuestionar:
¿Cuántos objetos hay?
¿Cuál de ellos es redondeado?

Para responder a las preguntas anteriores, se extrae las filas y columnas de cada objeto, ubicamos su centroide (figura2.2.8) y respecto de este, se compara la longitud del eje vertical con la del eje horizontal aceptando una cierta tolerancia con lo cual estaremos verificando si fue o no un objeto redondeado (figura2.2.9).

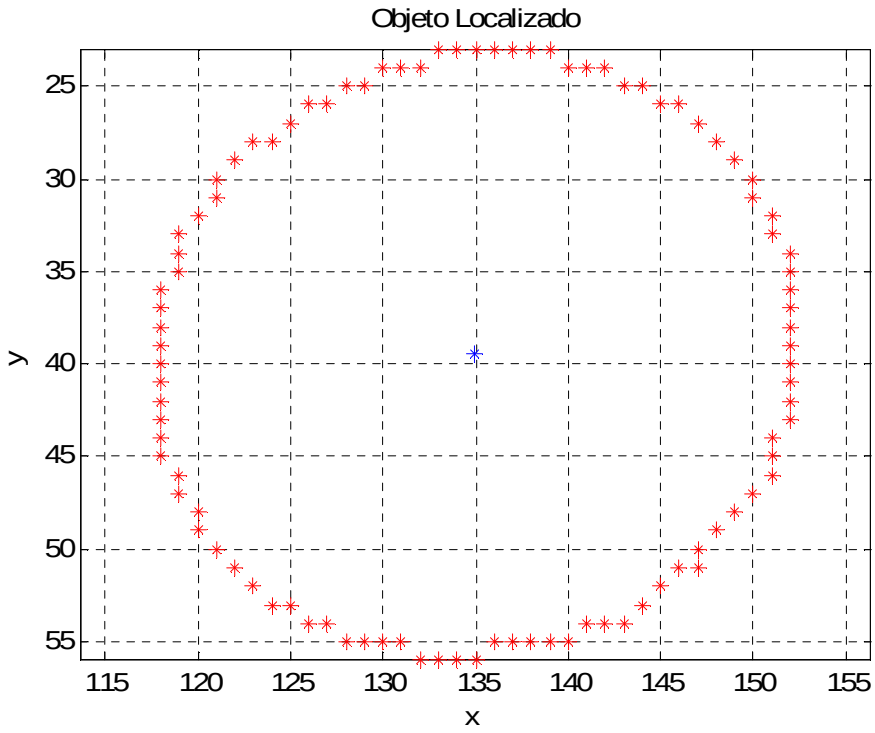


Figura 2.2. 8 Obtención del centroide del objeto seleccionado.

Se basa en identificar el objeto reflector dentro de todos los objetos que ya se encuentran en el sistema. Por lo tanto se comparan las características extraídas de los objetos con la del objeto reflector. El resultado de la comparación entre la información de las característica base con las características halladas puede tener altas probabilidades de acierto con un algoritmo bien desarrollado de identificación de objetos.

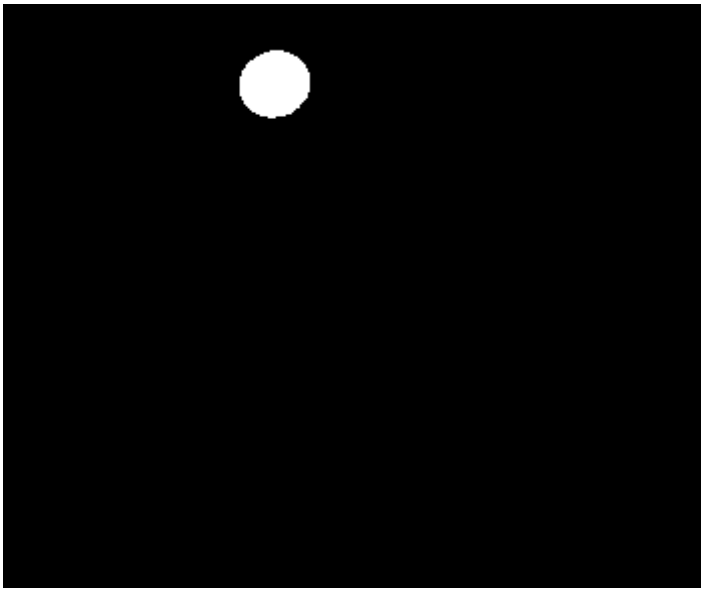


Figura 2.2. 9 Resultado de aplicar el algoritmo de detección de objetos redondeados.

Para facilitar la visualización de filas y columnas del objeto, y aplicar un algoritmo que ayude a determinar la posición (con el movimiento de la cabeza), se rellena la figura de pixeles (figura 2.2.10).

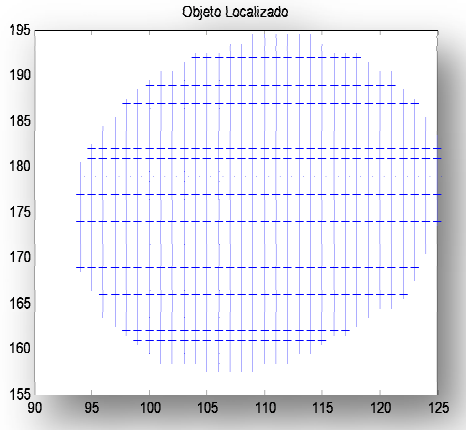


Figura 2.2. 10 Visualización de la ubicación del objeto en pixeles

Finalmente, tomando en cuenta la información del centroide, se verá la dis diferentes posiciones que sirva para el seguimiento del puntero del mouse (figura 2.2.11).

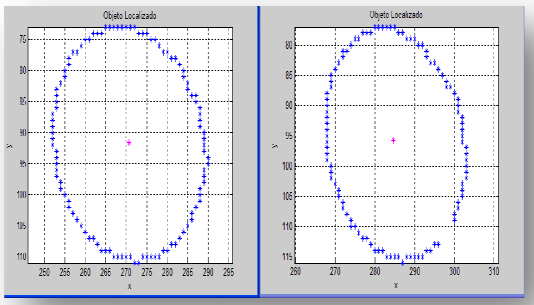


Figura 2.2. 11 Imágenes en posiciones diferentes.

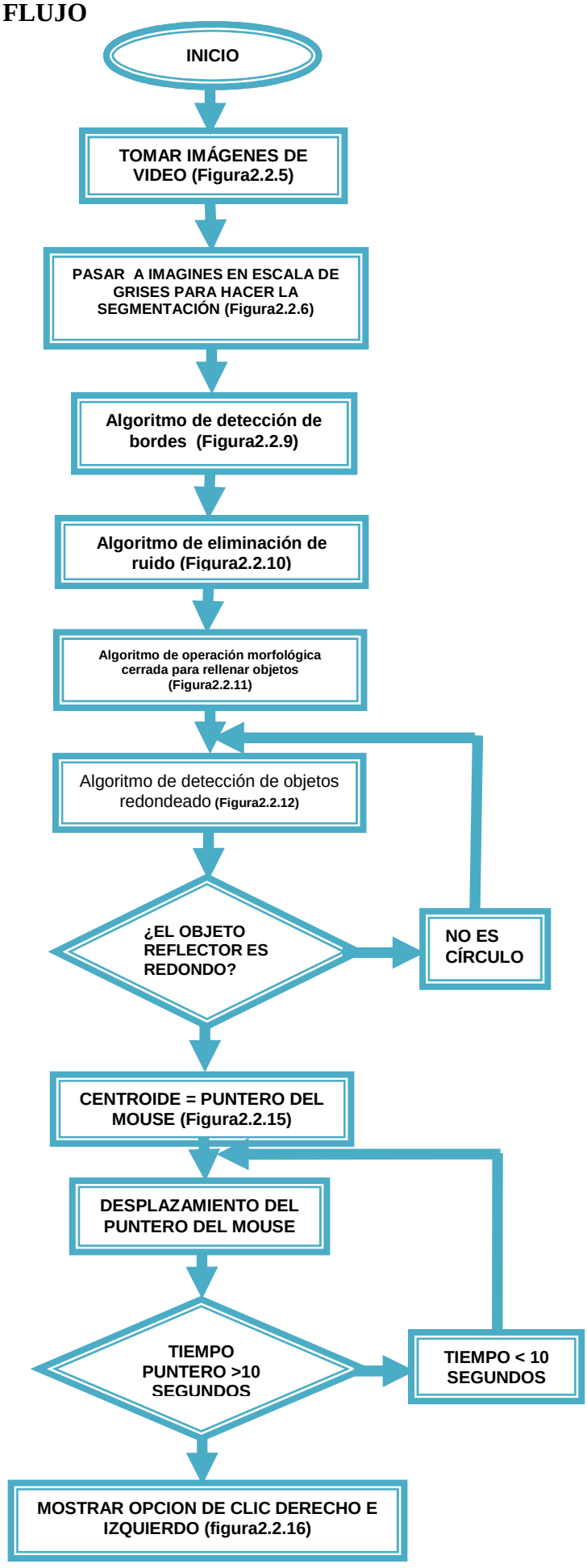
2.3 Método de detección de clic

Se procede a leer el tiempo que el usuario mantiene inmóvil su cabeza (el puntero del mouse) en un lugar determinado y si el tiempo es mayor a lo indicado por el programa (por ejemplo t > 10s), se ejecutará el cuadro de selección de clic derecho y clic izquierdo (figura 2.2.12).

Para la utilización de los comandos del puntero del mouse se utiliza las librerías de Java y Robot, con el fin de hacer coincidir el puntero del mouse con el centroide del objeto calculado.

Figura 2.2. 12 Opción de clic derecho y clic izquierdo.

2.4 DIAGRAMA DE FLUJO



CAPÍTULO III

IMPLEMENTACIÓN DE LOS ALGORITMOS

En este capítulo se implementará los algoritmos y métodos mencionados en el capítulo anterior, además se implementará el algoritmo para detección del objeto reflector, que se realizan en tiempo diferido, y una vez concluidos las respectivas pruebas, se implementarán en tiempo real para la respectiva evaluación de su desempeño. También se analizará las reacciones bajo las diferentes condiciones de luz.

3.1 DESARROLLO DE ALGORITMOS DE TOMA DE IMÁGENES EN TIEMPO DIFERIDO.

Se procede a leer imágenes a través de la captura de imágenes desde la cámara web. Se almacenará cada imagen en un archivo con las diferentes características tanto de resolución como de iluminación, las cuales ayudará a clasificar y ver probabilidades de acierto y desacierto.

3.1.1 Lectura de imágenes

Para trabajar con las imágenes almacenadas en MATLAB se procederá a leerlas con la función **imread** de la siguiente manera:

```
I=imread('foto8.JPG');
```

La variable I se encuentra almacenada la imagen (Figura 2.2.5 Adquisición de imagen) que muestra al usuario con el dispositivo (objeto reflector) en su frente, el cual ayudará a emular el mouse, ya explicado en el anterior capítulo.

Procedemos a transformarla a escala de grises como se explicó en el capítulo II. Esto ayudará a realizar un mejor análisis de imágenes. Para este fin se utilizará la función **rgb2gray**, (figura2.2.6. Imagen en escala de Grises). A continuación para asegurar mejores resultados se pasará a la imagen que está en escala de grises a imagen doble (figura2.2.7), con la función **im2double**. Y para mostrar estas imágenes se utilizó la función **imshow** (figura 3.1) la cual muestra en un cuadro incluido menú que ofrece MATLAB como se muestra en esta ilustración:

```
I2=rgb2gray(I);
I3 = im2double(I2);
imshow (I2)
```

Figura 3. 1 Ejemplo de la función imshow

Luego de utilizar las funciones antes mencionadas, desplegaremos la imagen recortada utilizando la función **imshow.**

3.1.2 Pre-procesado de imágenes.

El pre-procesamiento de la imagen ayudará a mejorar las características de la misma antes de ser analizadas.

3.1.2.1 Detección de Bordes

La detección de borde ayudará a resaltar los bordes de los objetos que se encuentran en la imagen y eliminará la parte rellena de los mismos. Esto se realizará con la función **edge**, este efecto devuelve una matriz binaria compuesta de “unos” los cuales indican los bordes y de “ceros” que indican la parte negra o hueca de la imagen; **edge** permite definir los bordes como:

- Lugares donde la primera derivada de la intensidad es mayor en magnitud, que un umbral.
- Lugares donde la segunda derivada de la intensidad tiene un cruce por cero.

Estas derivadas forman parte importantísima para cumplir con la detección de bordes y con la ayuda de las siguientes técnicas:

“sobel, prewit, roberts canny, zerocross, log”.

A continuación se realizará las pruebas con las diferentes técnicas para elegir la que mejor se adapte a los objetivos mencionados. Para comenzar con la detección de bordes a partir de la imagen doble que se muestra con la figura 3.1.

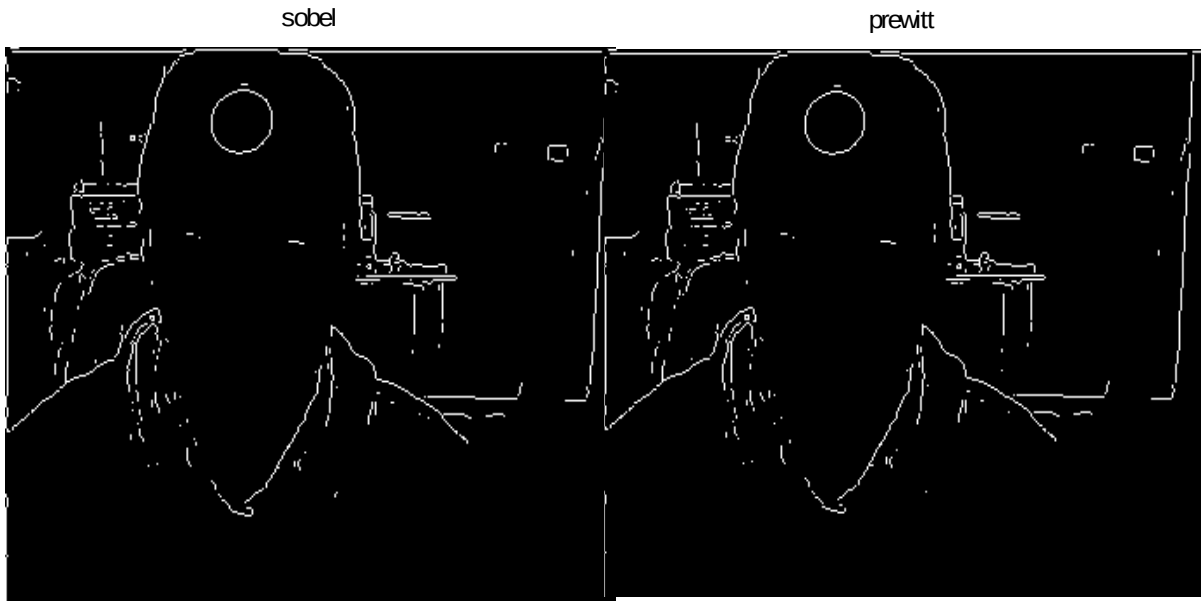


Figura 3. 2Pruebas con diferentes tipos de detección de bordes

La técnica que da los mejores resultados para nuestros propósitos es el de **Prewitt**, como se puede apreciar los detalles de objetos vecinos que no interesan son mínimos y se mantiene claro los detalles del objeto reflector, el cual es nuestro principal objetivo.

La función se detallará a continuación:

```
BW1=edge(I3,'prewitt');
```

```
figure, imshow(BW1)
title('técnica de prewitt')
```

3.1.2.2 Eliminación de Ruido

Se sabe que el ruido es la perturbación o señal que altera los resultados ideales de un proceso. Y es el mismo problema que se presenta en la adquisición de imágenes y finalmente en el reconocimiento del objeto. Por lo que el objetivo ahora es idear una manera de eliminar este ruido y esto se obtendrá removiendo los objetos no deseados.

Una de las maneras que puede facilitar la eliminación de ruido es con las herramientas de filtración de ruido que MATLAB ofrece, pero ya hechas las pruebas se determinó que la mejor opción fue la función **bwareaopen** ya que permite remover los píxeles que componen los objetos por numeración, como se mostrará a continuación.

```
BW2 = bwareaopen(BW1,P);
```

Donde **P** indica el número de píxeles que se desean remover de la imagen binaria que se va a proceder, la imagen se encuentra almacenada en la variable **BW1** y la resultante es otra imagen binaria almacenada en una nueva variable llamada **BW2**.

A continuación se mostrará varios ejemplos con diferentes números de pixeles a fin de escoger el valor de P más adecuado.

imagen sin ruido P=20

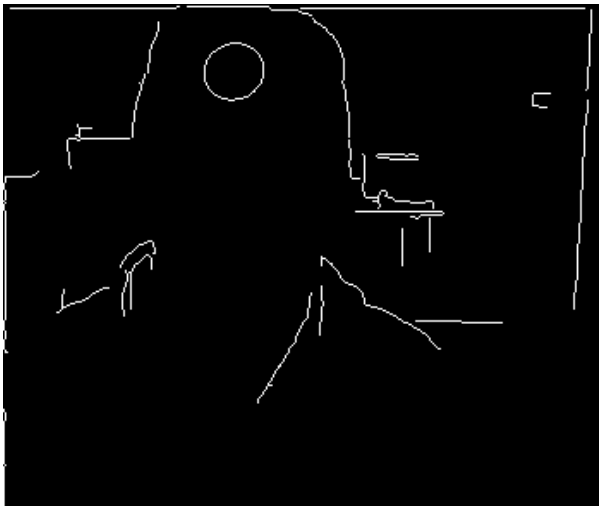
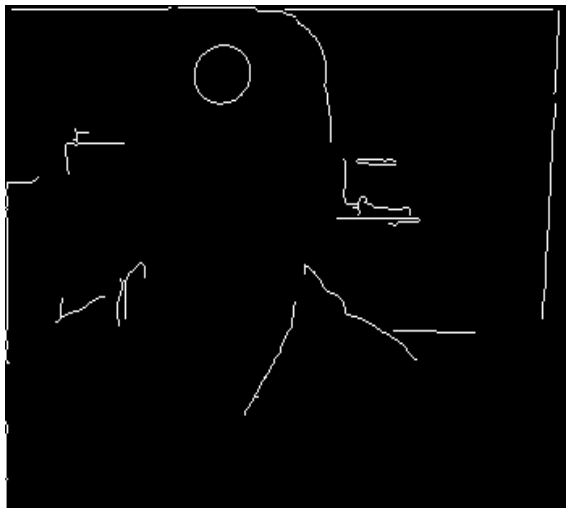


imagen sin ruido P=40



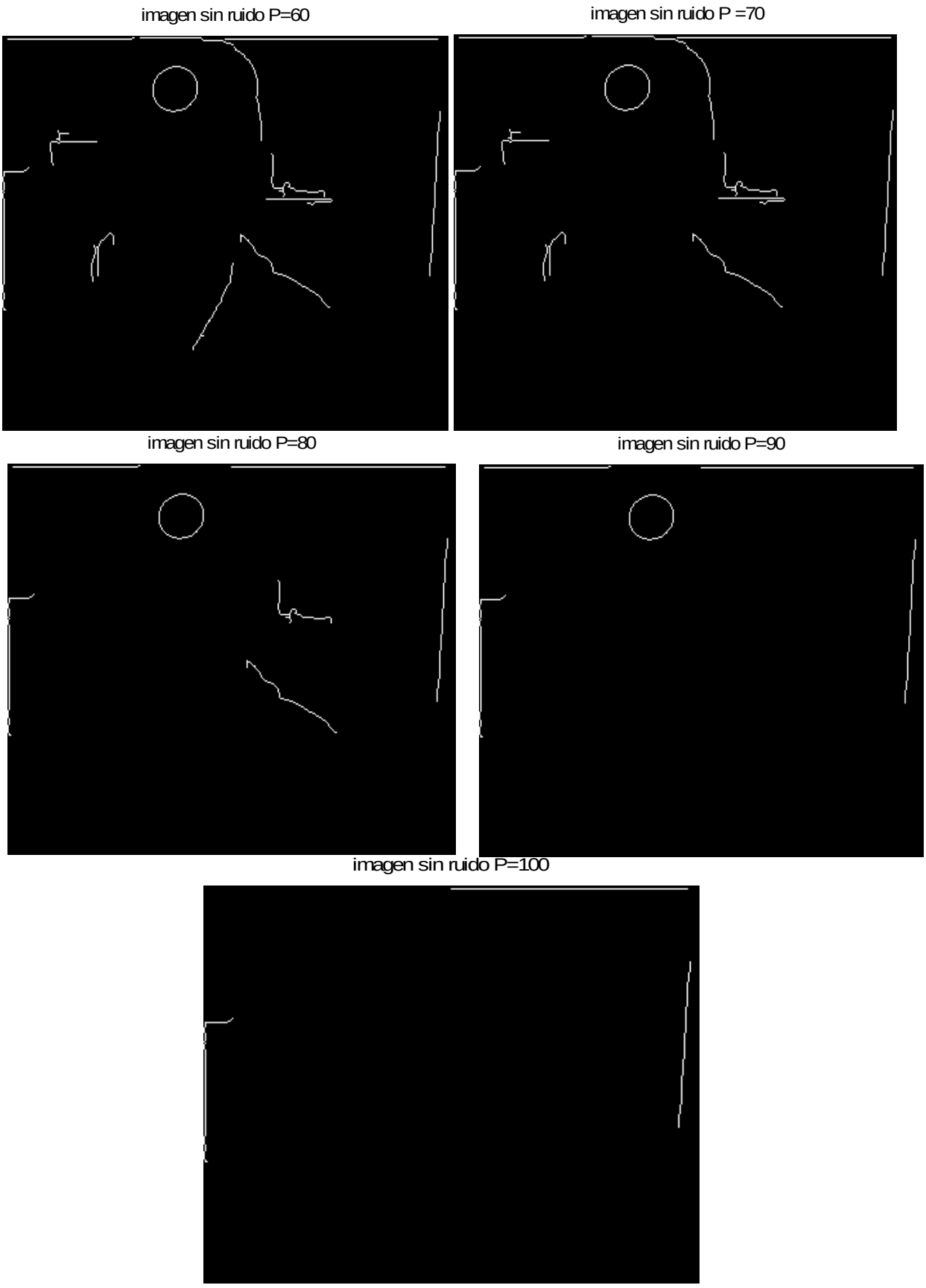


Figura 3. 3 Figuras de Eliminación de ruido.

Como se puede ver en los ejemplos con los diferentes valores de P se determinó que la prueba se dio con **P = 80** ya que con valores menores a este aún existen muchos objetos alrededor que no se necesitan y con valores mayores a este se pierde nuestro objeto principal (objeto reflector). La idea principal es mantener la información principal para poder procesar la imagen.

La función **strel** permite crear una estructura morfológica de una determinada forma y anchura de **N** píxeles y con la función **imclose** la cual permite que la operación morfológica cierre a la imagen almacenada en **SE**, como sigue de la siguiente manera.

```
se = strel('disk',1);
BW11 = imclose(BW2,se);
```

También se utiliza la función **imfill** la cual permite rellenar los agujeros de una imagen binaria. (figura 3.2 Imagen de objetos rellenos).

```
BW12 = imfill(BW11,'holes');
    figure, imshow(BW12)
title('relleno de objetos cerrados')
```

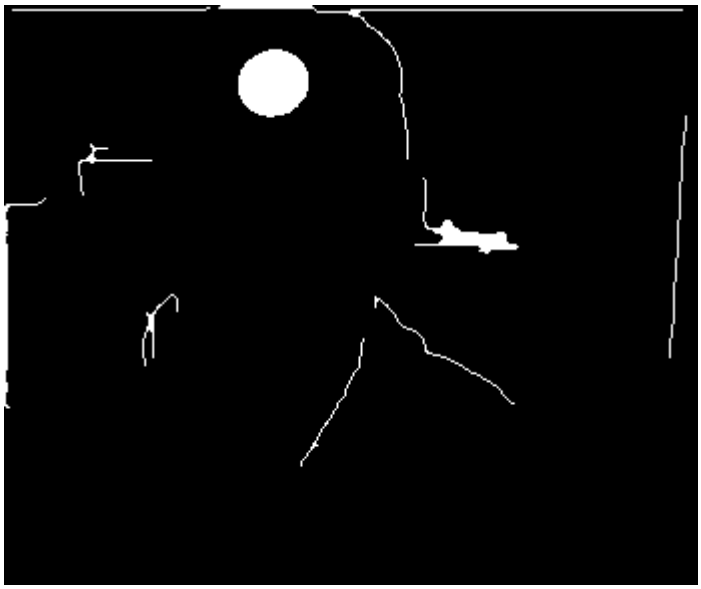
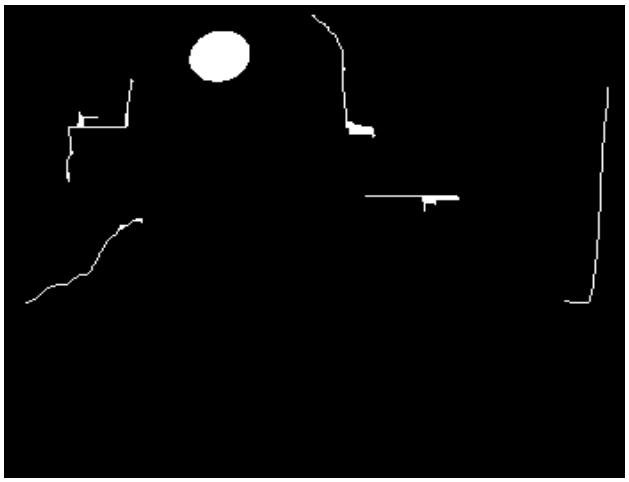


Figura 3. 4Imagen de objetos rellenos

Esta aplicación se encuentra almacenada en **BW12** y su respectiva aplicación corresponde a la figura 3.4.

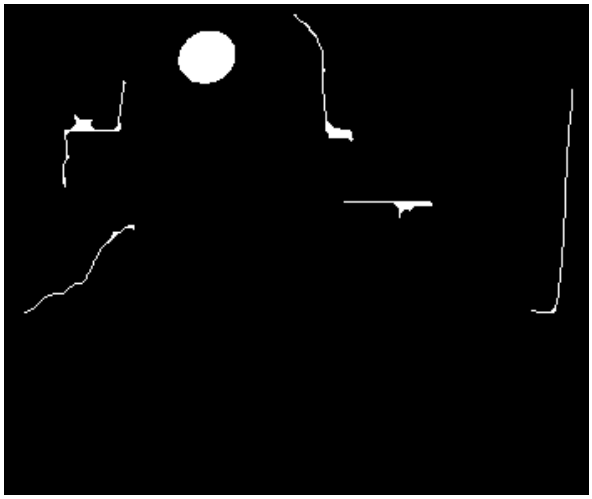
La función ***strel*** e ***imclose*** se utiliza para determinar cuál es el mejor radio dado por la variable N, para seleccionar el N más indicado se realizaron las siguientes pruebas.



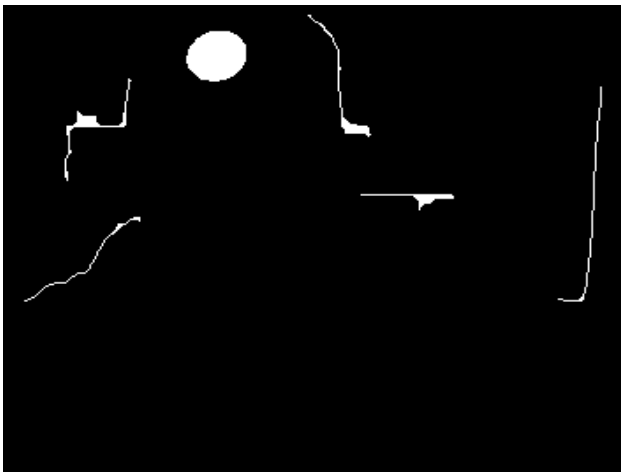
Para N = 1

Para N = 2

Para N = 3



Para N = 4



Para N = 5

Para N = 6

Figura 3. 5 Imágenes de la función Strel e imclose

Se concluye que debido a que hay aumento del grosor en los bordes según aumenta el valor ***N,*** el valor más adecuado para aplicar a la función ***strel*** es N = 1 y lo más importante es que el relleno de la imagen del objeto reflector no se deforma.

Para concluir con el pre-procesado de imágenes se mostrará a continuación un resumen de las líneas de programa a utilizar:

```
I=imread('foto8.JPG');
figure, imshow(I)
title('visualizacion del retrato')

%escala de grises con delineacion doble
       I2=rgb2gray(I);
       I3 = im2double(I2); % CAMBIO A DOUBLE
       figure, imshow(I3)
       title('imagen doble')

 %Detección de bordes:
       BW1=edge(I3,'prewitt');
       figure, imshow(BW1)
       title('técnica de prewitt')
        % eliminación de objetos
       BW2 = bwareaopen(BW1,80);
```

```
        figure, imshow(BW2)
        title('imagen sin ruido'

% Operaciones morfológicas con N=1
       se = strel('disk',1);
       BW11 = imclose(BW2,se);
       BW12 = imfill(BW11,'holes');
       figure, imshow(BW12)
       title('relleno de objetos cerrados')
```

3.1.3 Propiedades de Objetos

Con las propiedades básicas (área, volumen, perímetro, centroide, etc.) de los objetos de diferentes figuras como círculos, rectángulos, cuadrados, triángulos, etc., se desea plantear una solución para lograr el reconocimiento del objeto reflector y esta forma se lo puede lograr con las diferentes funciones que ofrece MATLAB como se procederá de la siguiente forma:

```
[M,NUM]=bwlabel(BW2,8);
stats = regionprops(M,'all');
```

Donde M indica la matriz de BW2 que contiene la última imagen pre-procesada (Figura2.2.11 Imagen con operación morfológica cerrada), donde los ceros indican el fondo de la imagen y diferentes conjuntos de números indican los distintos objetos que esta contiene.

NUM indica los números totales de los objetos que se encuentran en la figura, en nuestro ejemplo se encuentran siete objetos. El numero 8 o 4 indica la conexión cercana de 8 o 4 objetos, o a su vez ninguno el cual indica por default el numero 8.

La función regionprops se puede aplicar para diferentes aplicaciones como puede ser área, perímetro, centroide, etc., y para notar esto a continuación se muestra un ejemplo de la área y perímetro del objeto reflector, como se podría denotar de otros objetos pertenecientes a la imagen.

```
a = stats(k).Area;      %area = 35
 p=stats(k).Perimeter; %perim = 246.6274
```

A continuación definiciones de algunas propiedades.

Área: Calcula el área en píxeles cuadrados de la región.
Centroide: Posición del centroide de la región de algún objeto en mención.
Perímetro: Calcula el perímetro en pixeles lineales de la región.

3.1.4 Algoritmo de reconocimiento de objeto reflector.

Una vez que por medio de las funciones de MATLAB se han obtenido las propiedades del objeto se procede a plantear un algoritmo que ayudará a definir al objeto reflector como objeto principal y por medio de la propiedad centroide ubicar la posición del mouse.

Con el análisis de estas propiedades se trato analizar un posible algoritmo de comparación de áreas, perímetros o centroide, etc., pero debido a que la reflexión del objeto por medio de la webcam el objeto pierde características de círculo perfecto y no se podría aplicar áreas o perímetros de círculo como propiedad.

Es debido a ello, que la función-propiedad que ofrece MATLAB escogida es el centroide como se mostrará posteriormente que a pesar de que el círculo pierde sus propiedades aún sigue siendo un círculo redondeado.

Entonces la solución que se desea plantear es que por medio de un algoritmo se obtenga el objeto redondeado y se compare el valor del centroide con el fin de determinar si el objeto identificado es realmente el objeto reflector y este algoritmo se procede de la siguiente manera:

Primero se muestra como se obtiene el valor del centroide por medio de las funciones que ofrece MATLAB.

```
%PROPIEDADES DE LA IMAGEN
      stats = regionprops(M,'all'); %todas las propiedades
% CENTROIDE
      c=stats(k).Centroid;  % 134.9792   39.4792
      x = c(:, 1);  %centroide componente x
      y = c(:, 2);    %centroide componente y
```

Antes de explicar el algoritmo realizaré una breve explicación de la determinación del uso de bordear diferentes objetos (figura 3.6) y aplicar las propiedades mencionadas anteriormente.

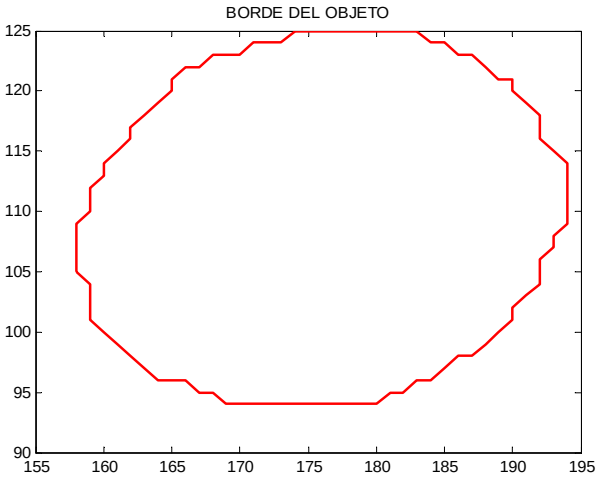


Figura 3. 6 Borde del objeto redondeado

La función **bwboundaries** permite extraer los bordes de cada uno de los objetos de una imagen binaria y a su vez etiquetarlos, si así se lo deseara. Para mostrar las con la aplicación de esta función se utiliza la función ***label2rgb*** la cual permitirá convertir una matriz etiquetada en una imagen RGB, donde **V** es la matriz a utilizar tomada de la foto original (figura 3.5) y se la convierte en imagen **RGB** (figura 3.7)**,** como se muestra a continuación:

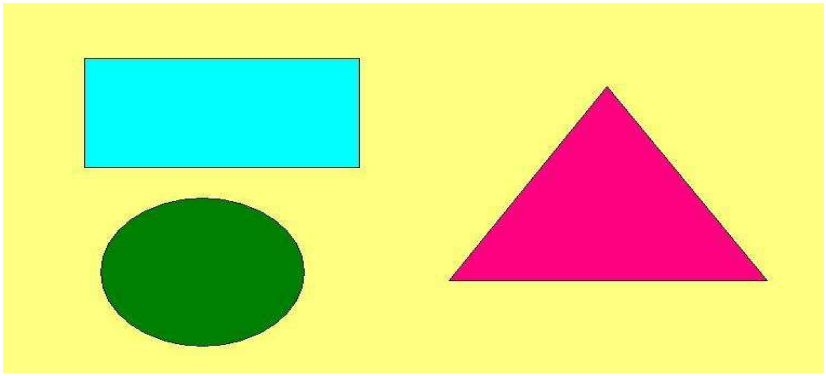


Figura 3. 7 Foto original para aplicación de bordes y diferentes propiedades

```
[F,C]=bwboundaries(V,'noholes');

%Desplega cada objeto en RGB
figure
imshow(label2rgb(C, @jet, [.5 .5 .5]))
title('resultado de las figuras en RGB')
hold on
```

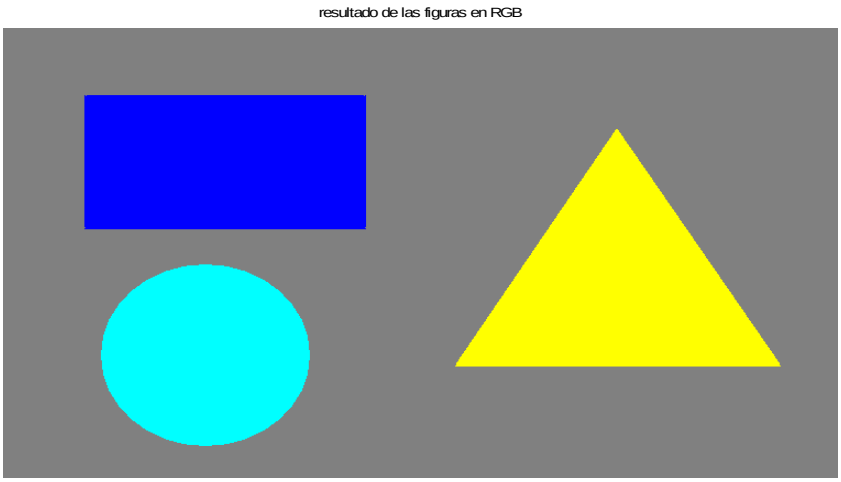


Figura 3. 8 Figuras convertidas en RGB

Ahora se trata de leer todos los bordes de cada objeto con la función **boundary** y a su vez leer cada propiedad de los objetos como área, perímetro y centroide (figura 3.9) como se muestra a continuación:

```
for k =1:length(F)
    boundary = F{k};
    plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
```

```
        end

stats=regionprops(V,'all')

%objeto 1, rectángulo
 a1=stats(1).Area      %area = 33276
 p1=stats(1).Perimeter %perim = 771.3137
 c1=stats(1).Centroid;
 x1 = c1(:, 1);
 y1 = c1(:, 2);

%objeto 2, círculo
a2=stats(2).Area      %area = 26152
p2=stats(2).Perimeter %perim = 603.8133
c2=stats(2).Centroid;
x2 = c2(:, 1);
y2 = c2(:, 2);

%objeto 3, triángulo
 a3=stats(3).Area      %area = 34416
 p3=stats(3).Perimeter %perim = 875.7788
 c3=stats(3).Centroid;
 x3 = c3(:, 1);
 y3 = c3(:, 2);

plot(x1, y1, 'm*');
plot(x2, y2, 'm*');
plot(x3, y3, 'm*');

axis ij;
axis equal;
hold on;
xlabel('x');
ylabel('y');
title('centroide de los objetos');
```

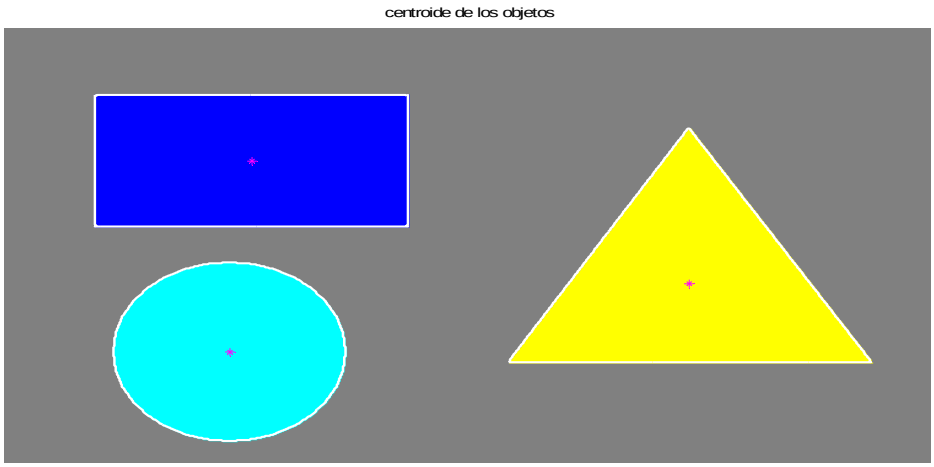


Figura 3. 9 Resultado del centroide de cada figura resaltada por un *.

Visto que las funciones de MATLAB cumplen correctamente nuestro fin, entonces volvemos al objetivo anterior (detección del objeto reflector), recordemos que el área de círculo es πR^2, donde R indica el radio del objeto y el perímetro del círculo es $2\pi R$ y centroide del círculo es el cruce de dos diámetros de un mismo círculo.

Como ya se explicó anteriormente no se puede aplicar ni los valores de área de ni de perímetro ya que la imagen que detecta la webcam pierde sus características (figura 3.10) y ya se deja de hablar de círculo y se menciona como objeto redondeado. Como se muestra a continuación recortes de algunos ejemplos tomados de este mismo proyecto.

Figura 3. 10 Diferentes visualizaciones de objeto reflector

Como se muestra claramente no son círculos perfectos, entonces lo que se procede a buscar un algoritmo más adecuado para la detección del objeto redondeado y para esto se utilizará la propiedad del centroide.

Ahora se mostrará el algoritmo para detectar el centroide paso a paso y comparar los datos del centroide y coordinar si efectivamente el objeto seleccionado es el objeto reflector o no.

En base a la matriz M (figura 2.2.11 Imagen con operación morfológica cerrada), la cual tiene todos los datos de los objetos que se encuentran en dicha imagen, de la cual extraemos la posición de las filas (R) y de las columnas (S), pero para esto solo necesitamos el borde de la imagen para tener el valor numérico de la longitud ya que si el objeto esta relleno son demasiados datos.

Entonces para esto se llama a cada objeto de la imagen para obtener su respectiva longitud tanto horizontal como vertical.

```
%RECONCIMIENTO DE REDONDEADO
   for  k=1:1:NUM
         [R,S] = find (M==k); %matriz de objetos
       for i=1:length(S);
           punts=S(i);
           puntr=R(i);
       end


          %LONGITUDES DE LAS FILAS
          a=1;
          for w=min(R):max(R);
          numeror=find(R==w);
          longitudr(a)=length(numeror);
          a=a+1;
          end
          longitudr;
         %LONGITUDES DE LAS COLUMNAS
          b=1;
```

```
for x=min(S):max(S);
numeros=find(S==x);
longituds(b)=length(numeros);
b=b+1;
end
longituds;
```

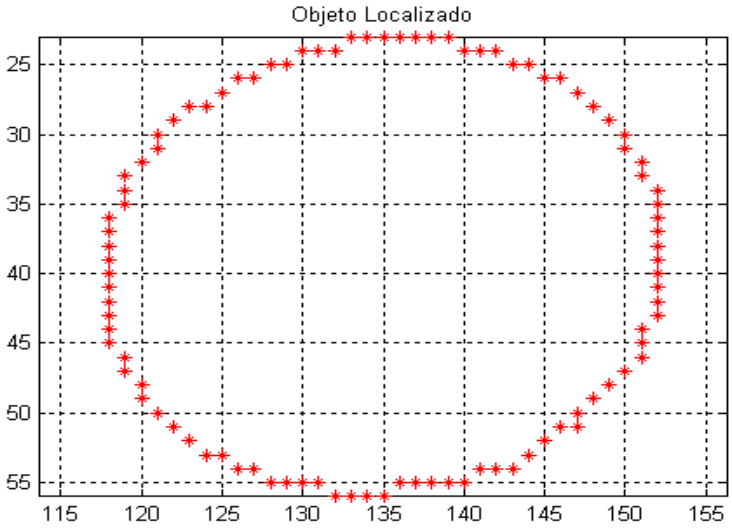


Figura 3. 11Imagen del Objeto Reflector con sus respectivas filas y columnas.

Para la localización de nuestro centroide se e mide las longitudes desde el centro reconocido por la función de MATLAB y con los valores máximos y mínimos y las comparamos, si los valores se identifican como iguales muestra un mensaje que dice ‘es círculo’, pero si estas distancias no son iguales indica ‘no es circulo’ y el algoritmo va en busca del siguiente objeto hasta encontrar el círculo (objeto reflector).

```
% CENTROIDE
  c=stats(k).Centroid;  %c=176.0104  109.4583
  x = c(:, 1);  %centroide componente x
  y = c(:, 2);    %centroide componente y
  %RECONOCIMIENTO DEL MOUSE
  x1=int32(x);  %numero entero de x
  y1=int32(y);  %numero entero de y

  %obtención del numero decimal
  x2=1000*x;
```

```
x3=1000*x1;
x4=abs(x3-x2);
x5=0;
if x4<=499 && x4>=450;
    x5=x1+1;
else x4<450 && x4>499;
    x5=x1;
end

y2=1000*y;
y3=1000*y1;
y4=abs(y3-y2);
y5=0;
if y4<=499 && y4>=450;  %num dec entre 45 y 49
    y5=y1+1;        %si es verdad sumar 1 a Y1
else y4<450 && y4>499;  %fuera del rango anterior
    y5=y1;
end
x5;
y5;

 respr=int32(max(R)+min(R))/2;
 resps=int32((max(S)+min(S))/2);

 distsmax=abs(x5-max(S));
 distsmax1=int32(distsmax);
 distsmin=abs(x5-min(S));
 distsmin1=int32(distsmin);
 distrmax=abs(y5-max(R));
 distrmax1=int32(distrmax);
 distrmin=abs(y5-min(R));
 distrmin1=int32(distrmin);

 %obtención del número decimal para distsmax
distsmax2=1000*distsmax;
distsmax3=1000*distsmax1;
distsmax4=abs(distsmax3-distsmax2);
distsmax5=0;
if distsmax4<=499 && distsmax4>=450;
```

```
    distsmax5=distsmax1+1;
else distsmax4<450 && distsmax4>499;
    distsmax5=distsmax1;
end

 %obtención del número decimal para distsmin
distsmin2=1000*distsmin;
distsmin3=1000*distsmin1;
distsmin4=abs(distsmin3-distsmin2);
distsmin5=0;
if distsmin4<=499 && distsmin4>=450;
    distsmin5=distsmin1+1;
else distsmin4<450 && distsmin4>499;
    distsmin5=distsmin1;
end

%obtención del número decimal para distrmax
distrmax2=1000*distrmax;
distrmax3=1000*distrmax1;
distrmax4=abs(distrmax3-distrmax2);
distrmax5=0;
if distrmax4<=499 && distrmax4>=450;%num dec entre 45 y 49
    distrmax5=distrmax1+1;      %si es verdad sumar 1 a Y1
          else distrmax4<450 && distrmax4>499;
      distrmax5=distrmax1;
end

%obtención del número decimal para distrmin
distrmin2=1000*distrmin;
distrmin3=1000*distrmin1;
distrmin4=abs(distrmin3-distrmin2);
distrmin5=0;
if distrmin4<=499 && distrmin4>=450;
    distrmin5=distrmin1+1;
else distrmin4<450 && distrmin4>499;
    distrmin5=distrmin1;
end

distsmax5;
```

```
            distsmin5;
            distrmax5;
            distrmin5;

            if (distrmax5==distrmin5 || distrmax5==(distrmin5+1) ||
            (distrmax5)==distrmin5-1) && (distsmin5==distsmax5 ||
            (distsmin5)==(distsmax5+1) || distsmin5==(distsmax5-1)) &&
            distrmax5>=12 && distrmin5>=12 && distsmax5>=14 &&
            distsmin5>=14

           disp('es circulo')
           k_circulo=k;
           break
           else
           disp('no es circulo')
           end
   end
```

Una vez ubicado el objeto redondeado (objeto reflector) se procede a mostrar la figura 3.12. Aclarando que el centroide se encuentra expresado con xc y yc como sigue a continuación.

```
    D = plot(S,R,'*r');
    grid on
    hold on

     % CENTROIDE
        c=stats(k_circulo).Centroid;  %c=176.0104  109.4583
        xc = c(:, 1);
        yc = c(:, 2);
        plot(x, y, '*b');
        axis ij;
        axis equal;
        hold on
        xlabel('x');
        ylabel('y');
title('Objeto Localizado');
vector = [xc , yc]
```

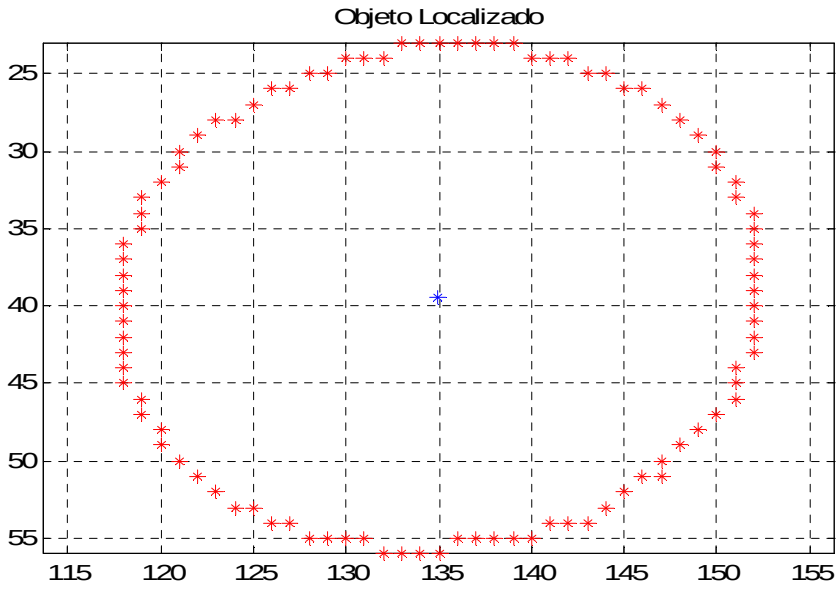


Figura 3. 12 Objeto Localizado con centroide

3.1.4 Posicionamiento del puntero del Mouse

Una vez que se ha detectado el centroide se hace coincidir con la posición del mouse y claramente al mover la cabeza se ve una actuación natural del mouse. Las funciones que se utilizan son de tiempo real que se explicará más detalladamente en el capítulo IV.

Para la utilización de los comandos del puntero del mouse se utiliza las librerías de Java y Robot, con el fin de hacer coincidir el puntero del mouse con el centroide del objeto calculado.

A continuación se mostrará las funciones a utilizar para el fin ya explicado, primero se toma el vector del centroide y se convierte a valores enteros.

```
t0 = clock;
vector = int32([xc, yc])
vector1=(vector+1);
vector1=(vector-1);
vector1=vector;
```

Aquí se guarda los datos del centroide en una nota de texto (figura 3.13) para tener la información completa del seguimiento del mouse en MATLAB.

```
dlmwrite('dato_vector_tiempo.txt', vector1, '-append')
mouse.mouseMove(xc, yc);
```

Una vez que el mouse se ha ubicado en la posición del mouse xc y yc se procede a aplicar la selección de clic.

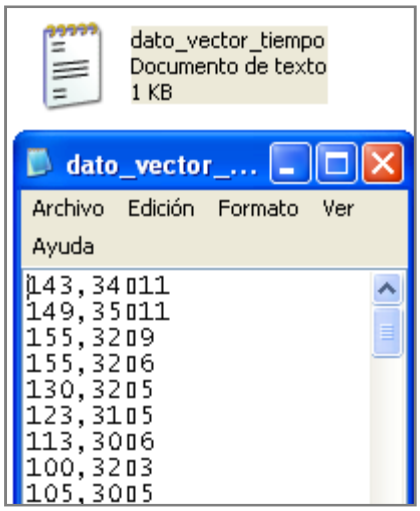


Figura 3. 13 nota de texto del vector xc y yc.

3.1.5 Método de selección de clic derecho o clic izquierdo.

Una vez que el usuario a realizado los movimientos del mouse y ha decidido realizar un clic (derecho o izquierdo), el usuario mantiene fija la cabeza hacia su objetivo y se procede a leer el tiempo que el usuario mantiene inmóvil su cabeza (el puntero del mouse) en un lugar determinado y si el tiempo es mayor a lo indicado por el programa (por ejemplo t > 10s), se ejecutará el cuadro de selección de clic derecho y clic izquierdo (figura 3.14).

Para esto se llama a las librerias de java y Robot, con el fin de hacer coincidir el puntero del mouse con el centroide calculado. Se agrega un pause de 5 segundos con el fin de que la persona que está utilizando el mouse pueda mover con libertad el puntero del mouse hasta cuando decida realizar una actividad de clic derecho o clic izquierdo.

Programa:

```
import java.awt.Robot;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

mouse = Robot;

vector = [xc, yc]
mouse.mouseMove(xc, yc);
%ayuda a ver la localizacion del mouse
mouse_loc = java.awt.MouseInfo.getPointerInfo.getLocation

cuadro=figure('Position',[vector1 260 70],...
              'Menu','None',
              'Name','HeadMouse','NumberTitle','off');
              pos=get(cuadro,'Position');

              izq=uicontrol('ButtonDownFcn',
              'pushbutton',...
              'String', 'Izquierdo',...
              'Position', [10 10 100 50],...
              'BackgroundColor',[.2 .8 0],...
              'FontWeight','bold',...
              'TooltipString','Izquierdo');
               disp('aplasto el izquierdo')

              der=uicontrol('ButtonDownFcn',
              'pushbutton',...
              'String', 'Derecho',...
              'Position', [150 10 100 50],...
              'BackgroundColor',[.2 .8 0],...
```

```
'FontWeight','bold',...
'TooltipString','Derecho');
 disp('aplasto el derecho')
         pause(2)
%clic izquierdo
        mouse.mouseMove((xc+50), (yc+50));  %yc +100
        pause(3)

 %clic derecho
        mouse.mouseMove((xc+200), (yc+50));
        pause(3)

 delete (cuadro)
```

Figura 3. 14 Cuadro de selección de clic derecho e izquierdo

CAPÍTULO IV

PRUEBAS Y RESULTADOS EN TIEMPO DIFERIDO Y EN TIEMPO REAL

En el presente capítulo se describirán los resultados obtenidos de las pruebas realizadas tanto en tiempo discreto como en tiempo real. También se discutirá los resultados para establecer los posibles errores y con los resultados obtenidos deseamos determinar el índice de efectividad del programa construido.

En la actualidad existe dos unidades principales de medidas de luz y estas son Lumen (mide la luz de una lámpara) y Lux (mide la intensidad de luz que cae en una superficie y equivale a un lumen al cuadrado).

Para calcular el nivel de Lux de una superficie se tiene en cuenta los siguientes parámetros:

- Dimensiones de la sala.
- Altura creciente de la luminaria.
- Altura del plano de trabajo (escritorio, plano de trabajo o suelo)
- Reflejos del techo, paredes y suelo.
- Cuánta suciedad tendrán las superficies de la sala y la luminaria después de varios años.
- Datos fotométricos en los accesorios que utilizaremos.
- Salida inicial de lumen de cada lámpara x el número de lámparas.
- Cuál será la salida de lumen de la lámpara en varios años.

Contemporáneamente al estudio de la medición de la luz, se han creado software donde se ingresan todos los parámetros y este luego de procesar indica el espacio requerido y el número de luminarias que se necesitan para el mismo.

A continuación se mostrará una tabla de niveles de luz de interior más utilizados en el día de hoy.

Actividad	Iluminación	
	FC	LUX
Almacenes, Casas, Teatros, Archivos	13.95	150
Trabajo sencillo de oficina, Clases	23.25	250
Trabajo Normal de Oficina, Trabajo en PC, Biblioteca de Estudio, Tiendas de Comestibles, Salones de Exposiciones, Laboratorios	46.50	500
Supermercados, Trabajos Mecánicos, Oficinas de Paisajistas	69.75	750
Trabajo de Dibujo Normal, Talleres Mecánicos Detallados, Quirófanos	93.00	1,000
Trabajo de Dibujo Detallado, Trabajos Mecánicos Muy Detallados	139.50 - 186.00	1,500 - 2,000

A continuación se mostrará una tabla de nivel de luz de exteriores como iluminación diaria, la cual es base fundamental para el desarrollo del proyecto.

Condición	Iluminación	
	FC	LUX
Luz solar	10,000	107,527
Luz del día	1,000	10,752.7
Día nublado	100	1,075.3
Día muy oscuro	10	107.53
Crepúsculo	1	10.75
Crepúsculo intenso	.1	1.08
Luna llena	.01	.108
Luna menguante	.001	.0108
Luz de las estrellas	.0001	.0011
Noche nublada	.00001	.0001

Posteriormente se mostrará equivalencias de LUX y FC.

LUX = FC (10.752)

FC = LUX / 10.752

FC = candelas-pies

En un lugar de trabajo con el computador es usualmente, oficina, aula de clases que corresponde a 23.25 FC ó 250 LUX esta iluminación es requerida por iluminación artificial es decir lámparas y la iluminación diaria corresponde a 1000 FC ó 10752.7 lo que claramente se puede ver que al trabajar con iluminación natural se obtendrá buenos resultados y en un día nublado los parámetros son menores por lo que los resultados serán poco aceptables como se demostrará posteriormente.

4.1.1 Aplicación del algoritmo según el nivel de iluminación.

Las pruebas para mostrar los diferentes niveles de iluminación se ha realizado con la misma persona y manteniendo la misma posición, para visualizar de mejor manera cualquier cambio que exista entre ellas.

Para la obtención de estos resultados se ha utilizado un luxómetro (dispositivo que mide diferentes intensidades de luz), cuenta con una pantalla LCD permanente, la cual expresa su medida en luxes.

Figura 4. 1Foto de luxómetro.

Calibres:

0-2000 Lux, Resolución: 1 Lux, Precisión ± (3%)
2000-19990 Lux, Resolución: 10 Lux, Precisión ± (3%)
20000-100000 Lux, Resolución: 100 Lux, Precisión ± (3%)
100000-200000 Lux, Resolución 1000 Lux, Precisión ± (3%)

Características:

- Luxómetro digital de 1 a 200 000 Lux
- Resolución de 1 a 1000 Lux
- Precisión: 0 a 20,000 Lux:
- Unidades de medida: LUX y FC
- LCD
- Usa batería
- Dimensión: 16 x 8 x 3,5 cm
- Peso 450g

4.2.1 Experimentación de funcionamiento

Las siguientes pruebas están basadas en función nivel de luz y en función según el usuario.

4.2.1.1 Experimentación en función de niveles de luz.

A continuación se mostrará los resultados para diferentes niveles de iluminación. Se tomará 30 imágenes de prueba para poder deducir cuántas de estas resultan válidas y cuál de estas no.

Luego se procederá a realizar un cuadro de error para los diferentes niveles de luz que se han obtenido a través del luxómetro.

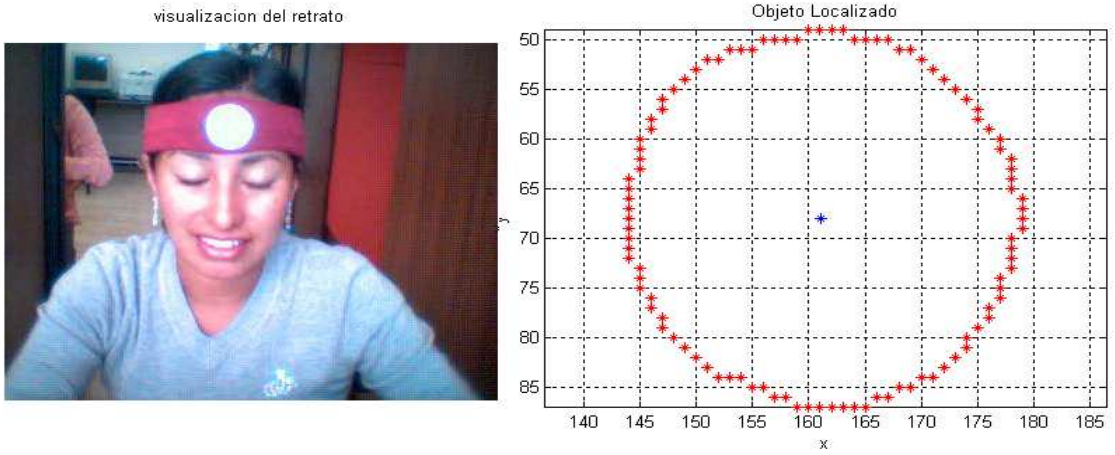
$$Er\% = \frac{V_r - V_m}{V_r} * 100$$

Er = error relativo

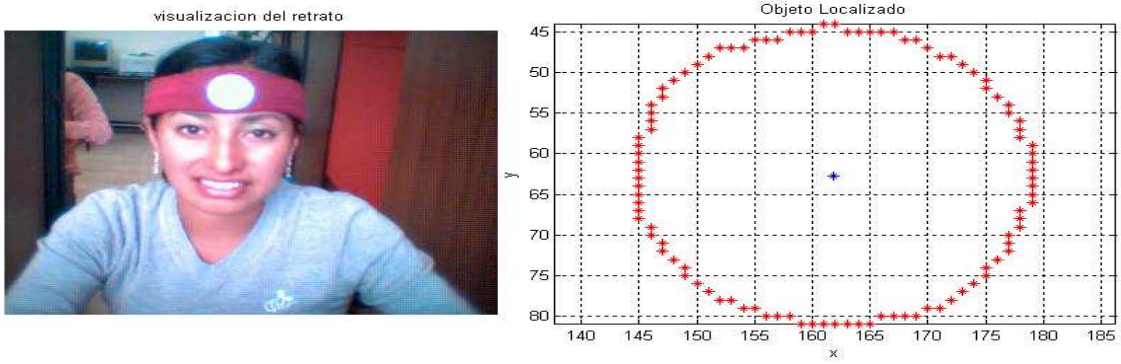
Vr = valor real

Vm = valor medido

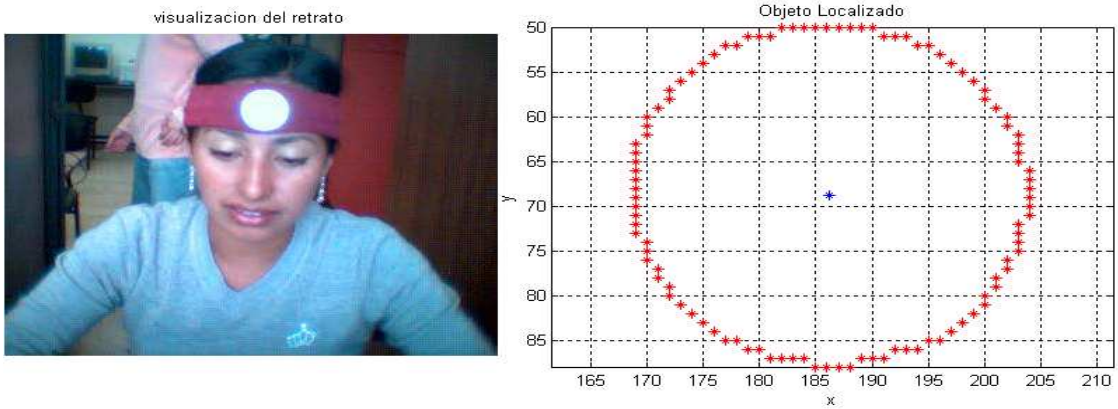
Las primeras pruebas serán para 850 luxes en diferentes posiciones (figura 4.2) para ver la evolución del proyecto a este nivel de luz.



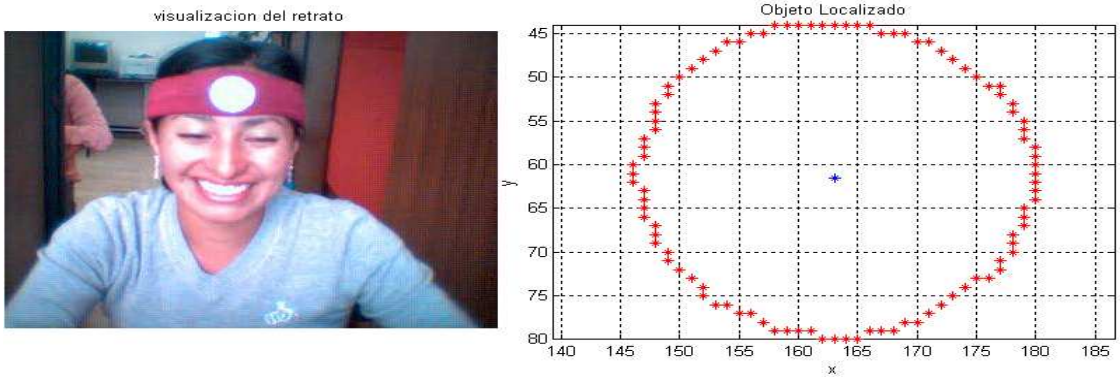
Resultado 1



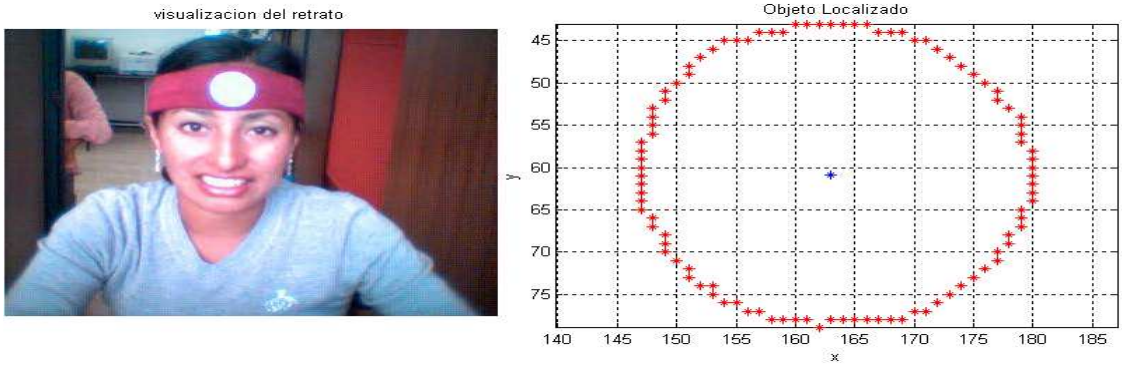
Resultado 2



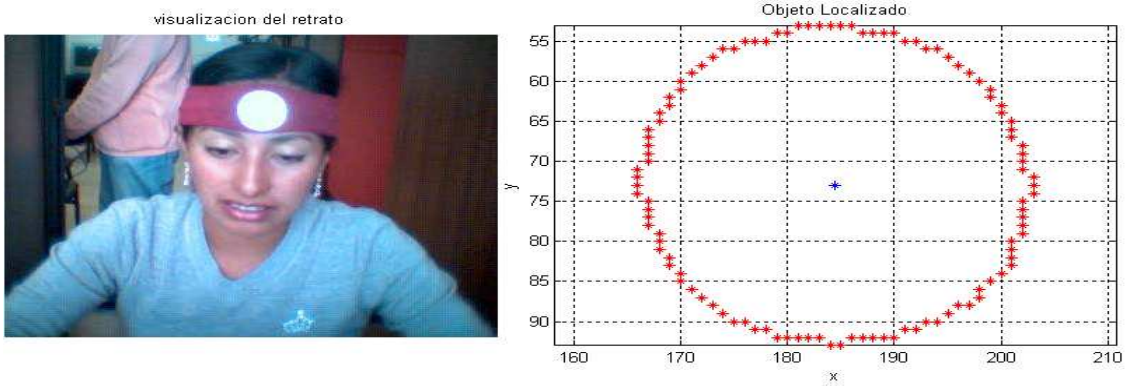
Resultado 3



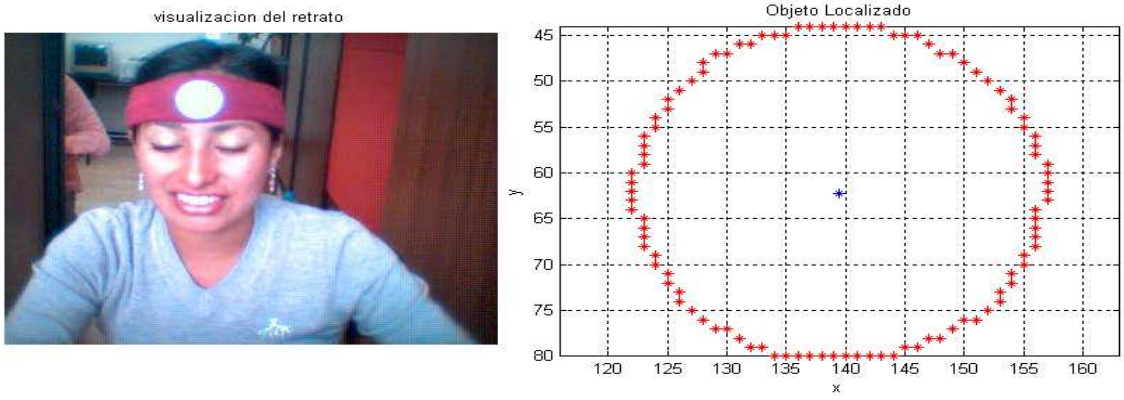
Resultado 4



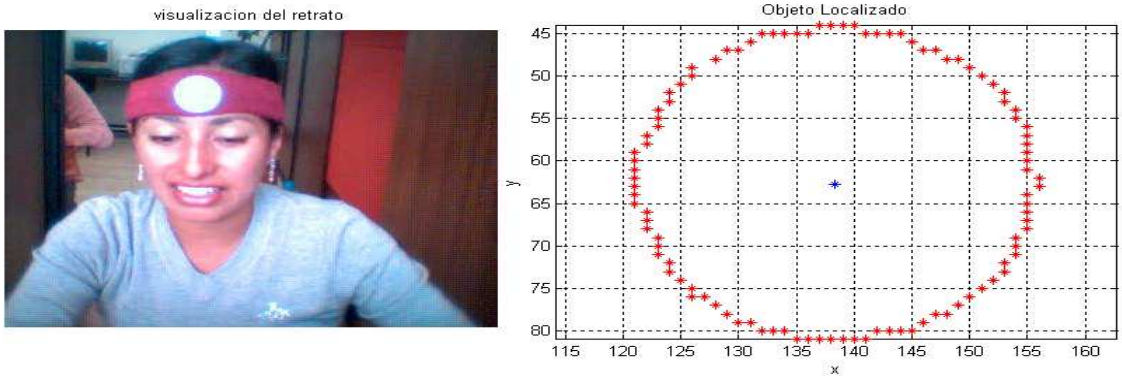
Resultado 5



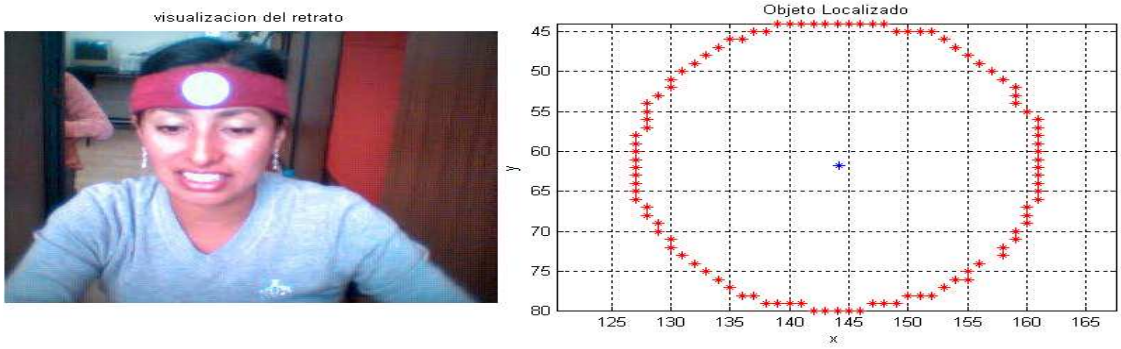
Resultado 6



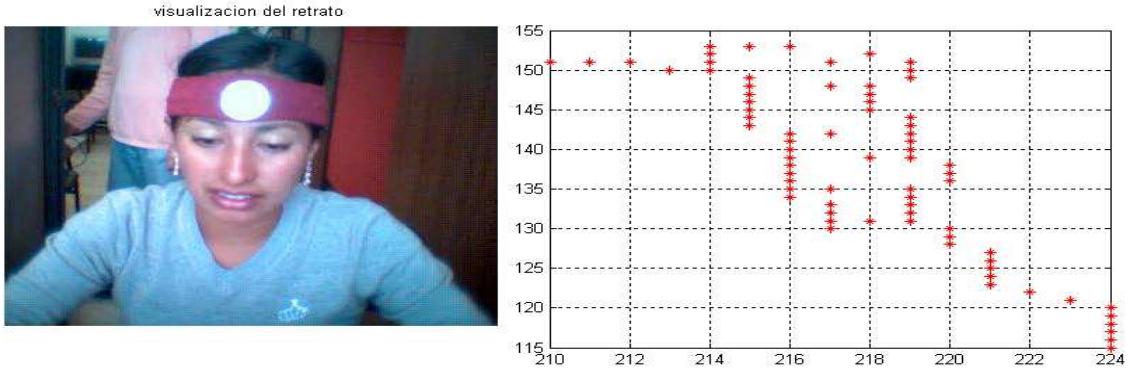
Resultado 7



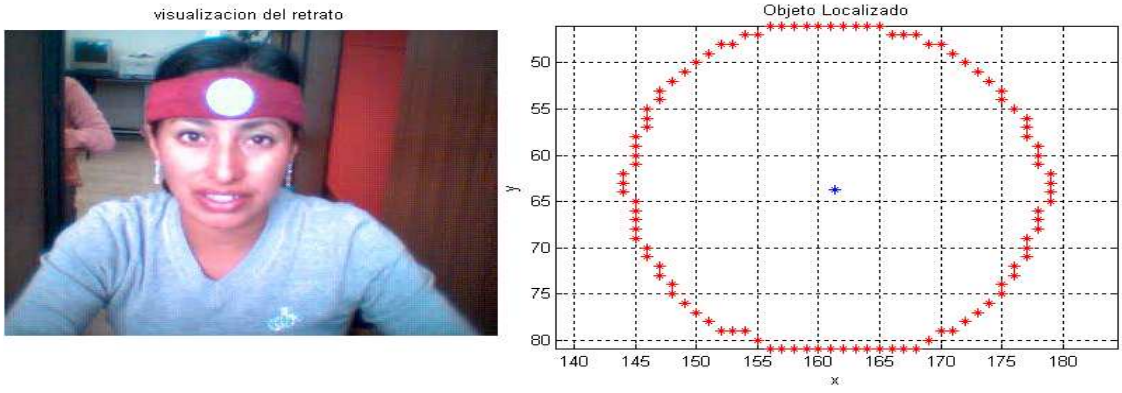
Resultado 8



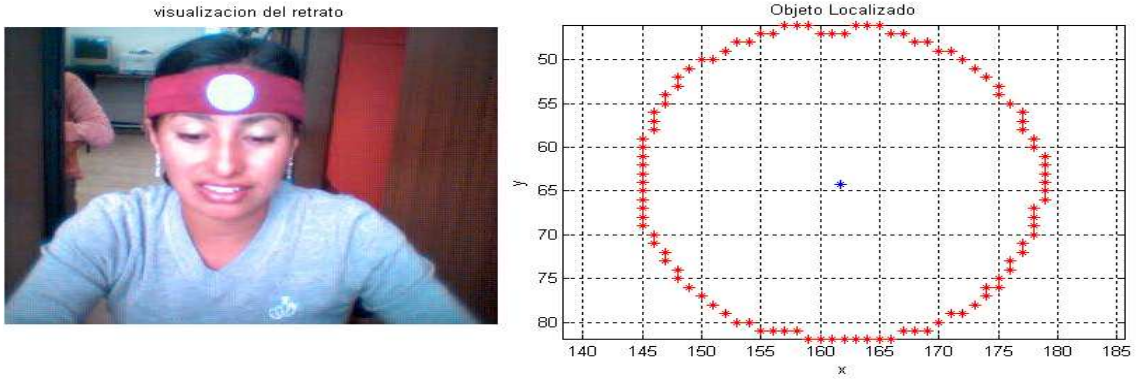
Resultado 9



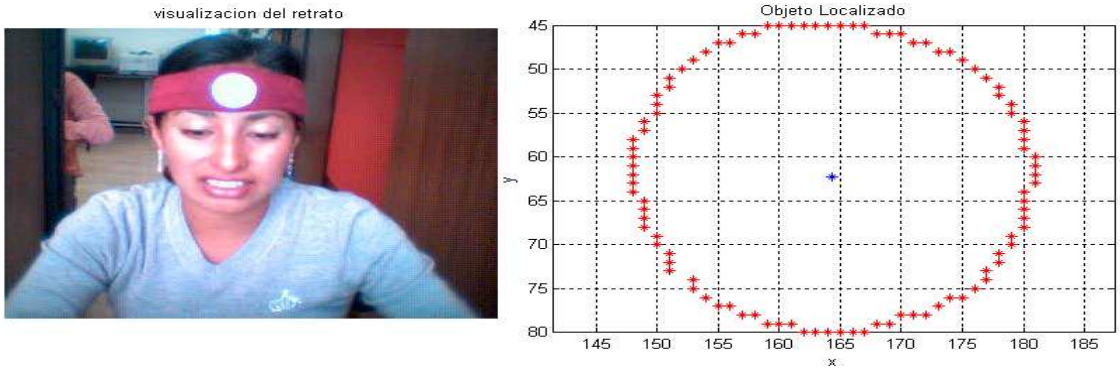
Resultado 10



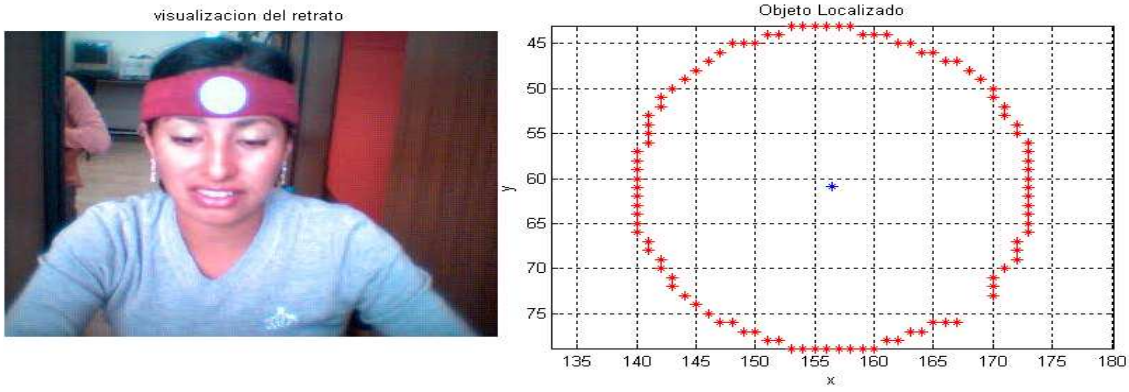
Resultado 11



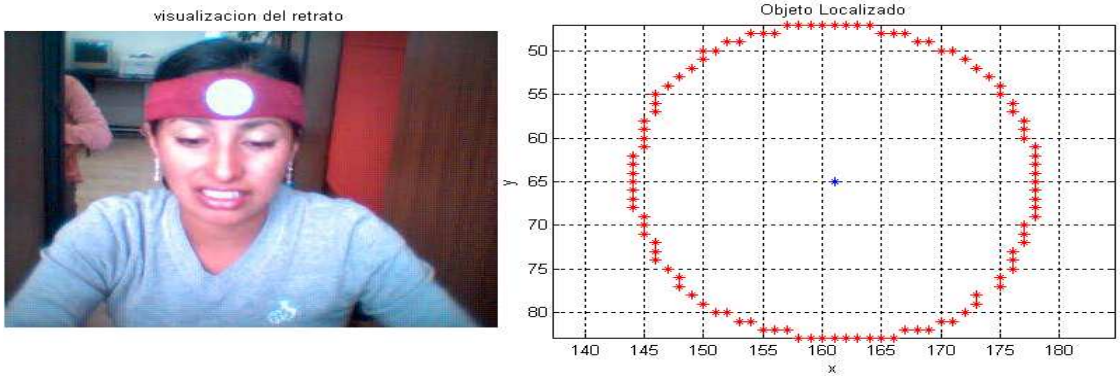
Resultado 12



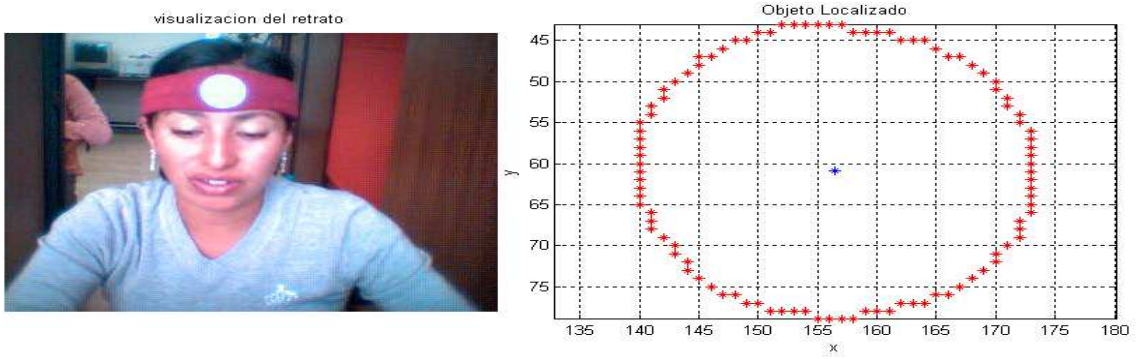
Resultado 13



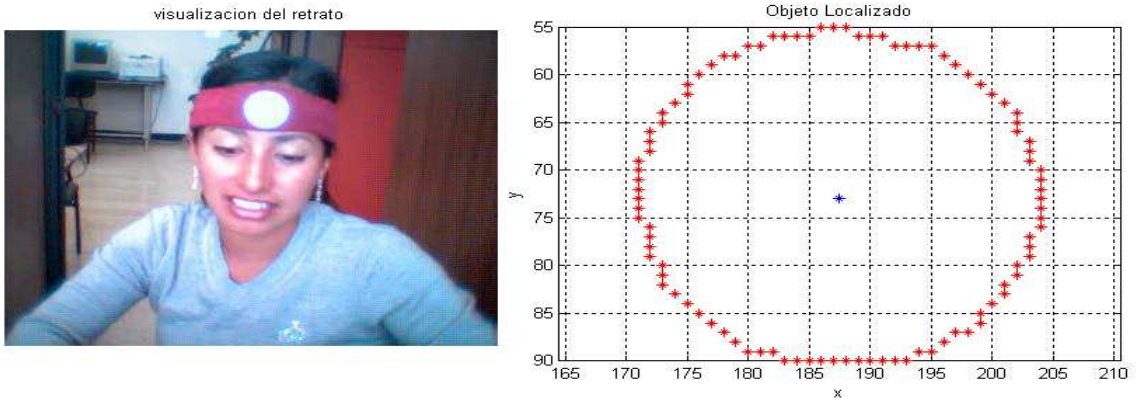
Resultado 14



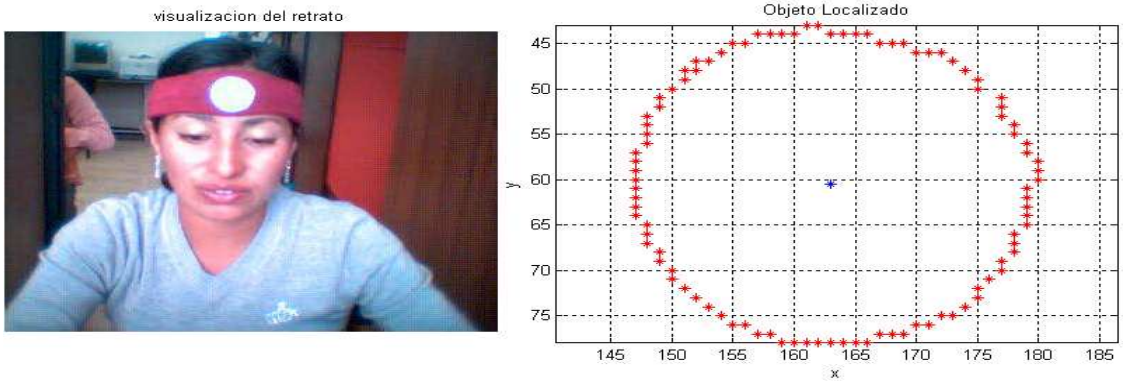
Resultado 15



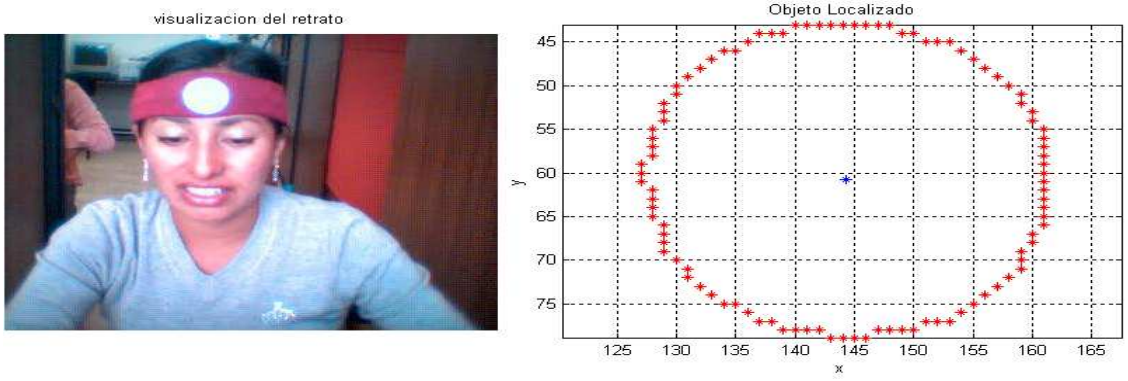
Resultado 16



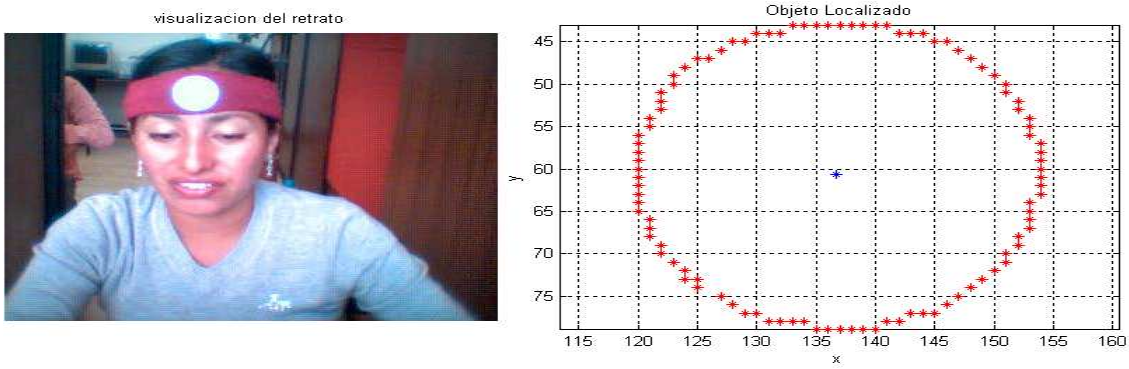
Resultado 17



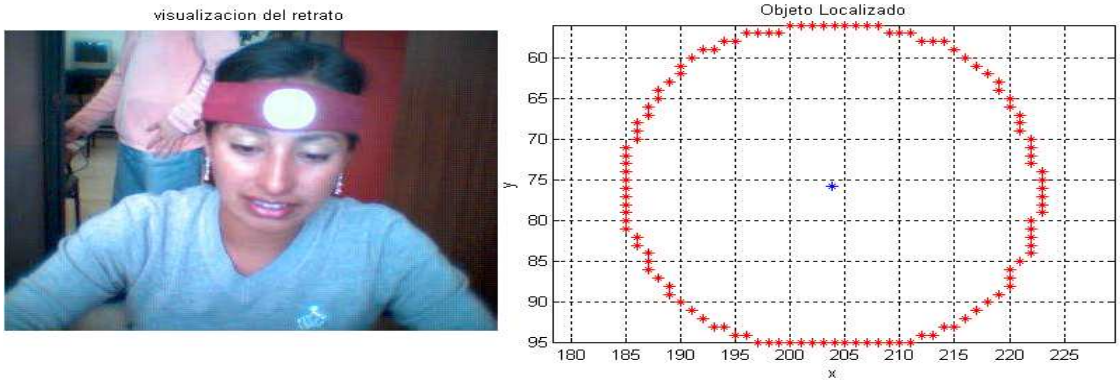
Resultado 18



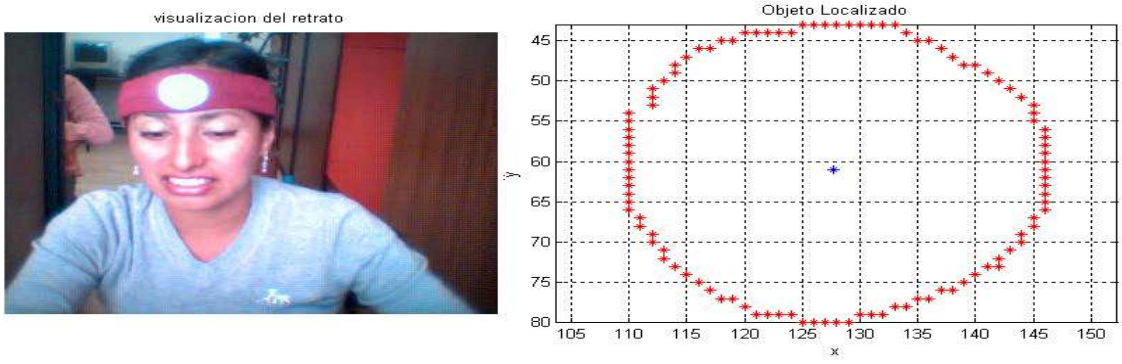
Resultado 19



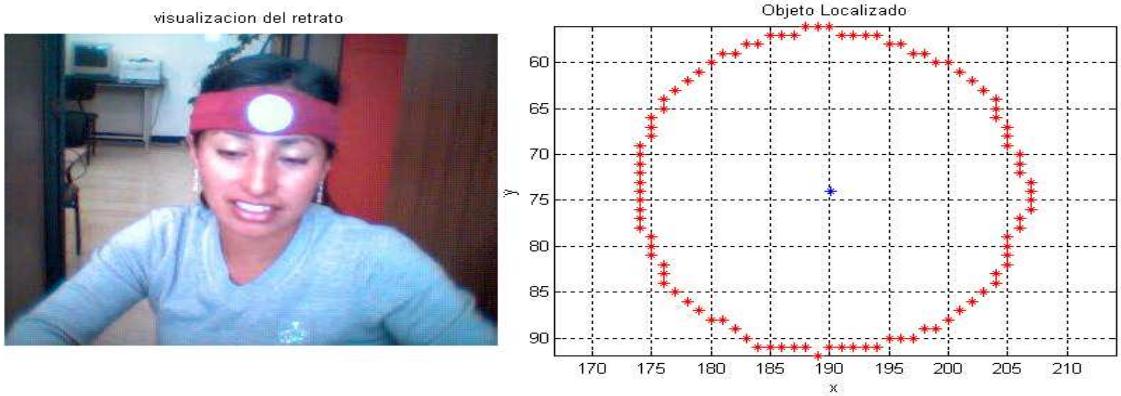
Resultado 20



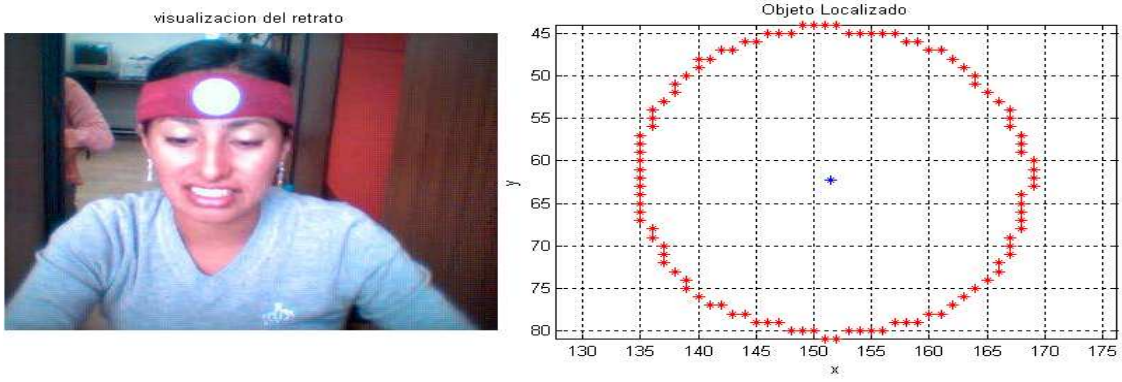
Resultado 21



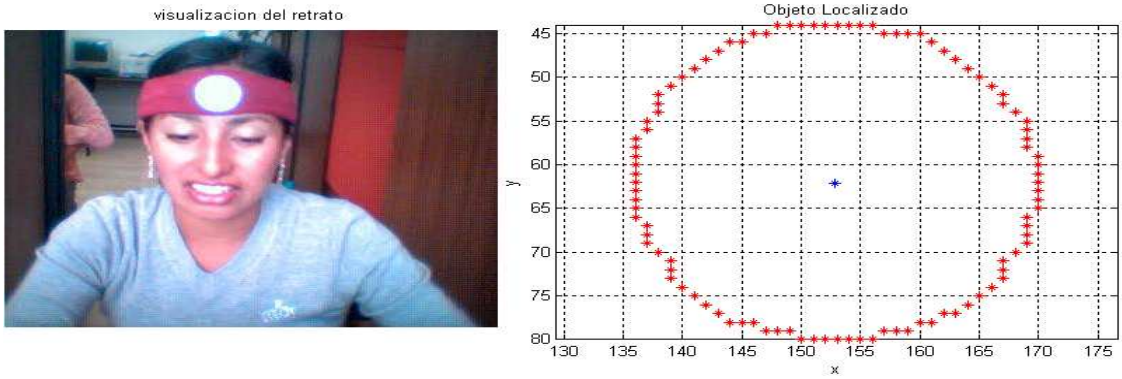
Resultado 22



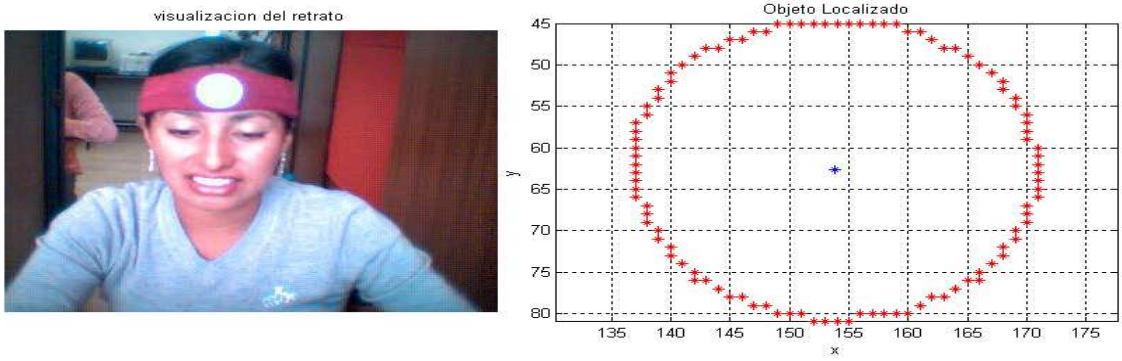
Resultado 23



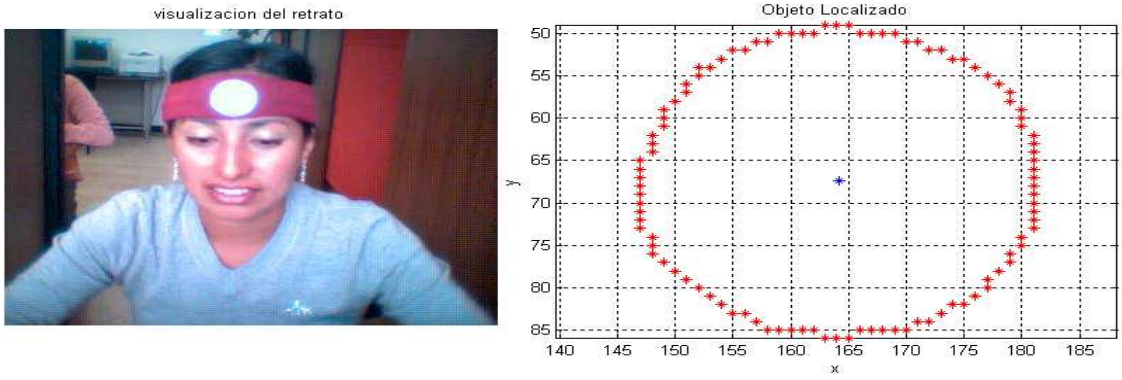
Resultado 24



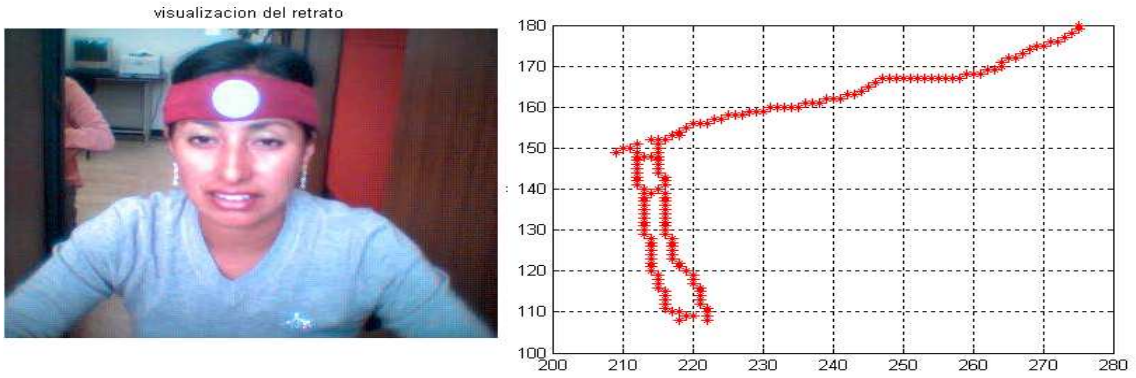
Resultado 25



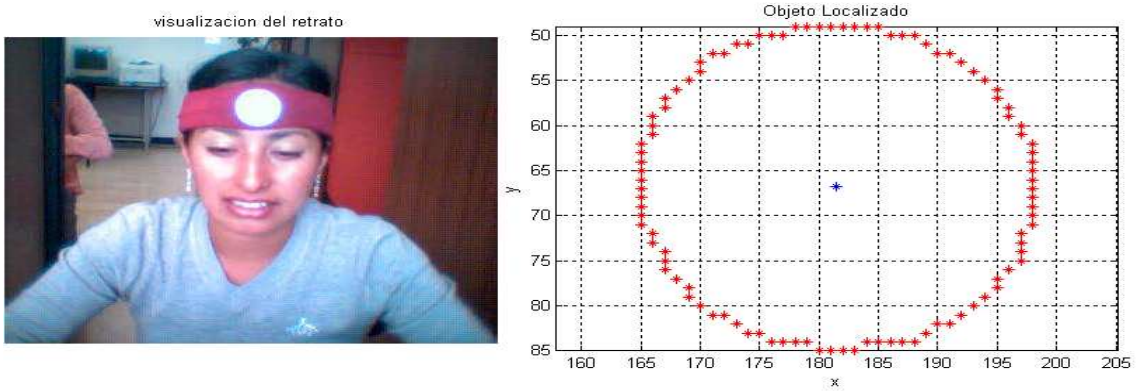
Resultado 26



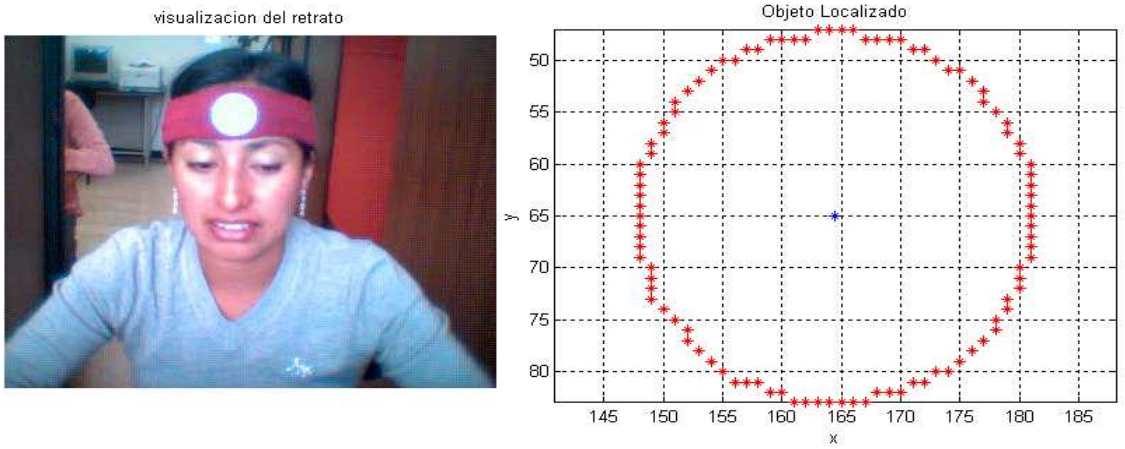
Resultado 27



Resultado 28



Resultado 29



Resultado 30

Figura 4. 2 Imágenes a 850 luxes.

El número de fallas a 850 luxes de 30 imágenes es de 2 imágenes de prueba. Entonces se procede a calcular el error de la siguiente manera:

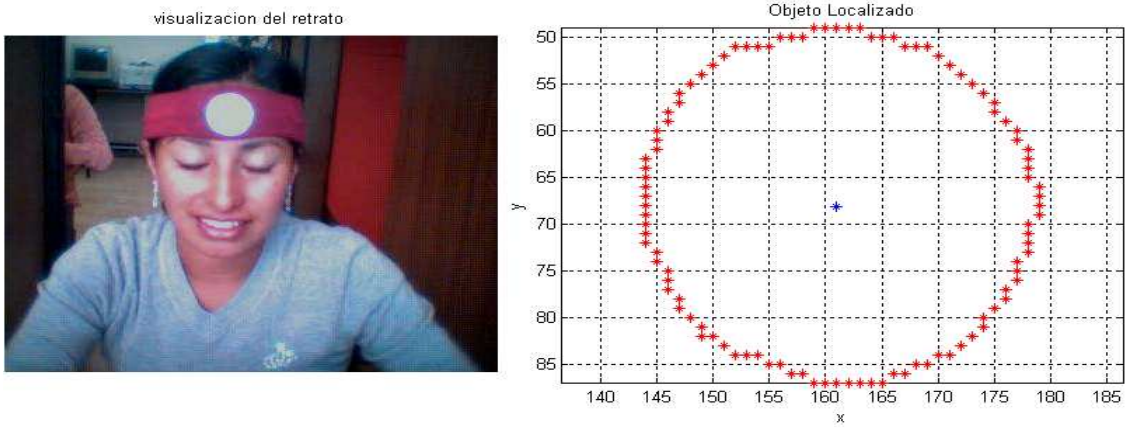
$$Er\% = \frac{V_r - V_m}{V_r} * 100$$

$$Er\% = \frac{30 - 28}{30} * 100$$

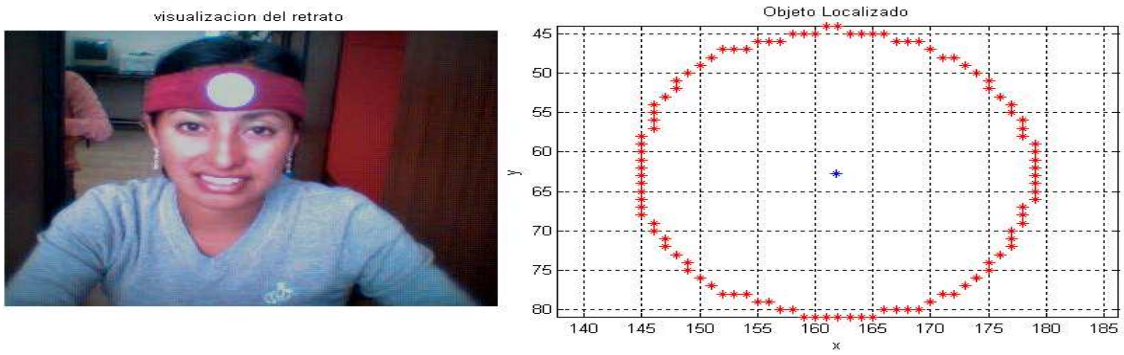
$$Er\% = 6.667\%$$

Como se puede notar en los resultados obtenidos de las imágenes de pruebas se produjeron 2 fallas, por lo tanto nuestro error relativo porcentual para 850 luxes es de 6.667%.

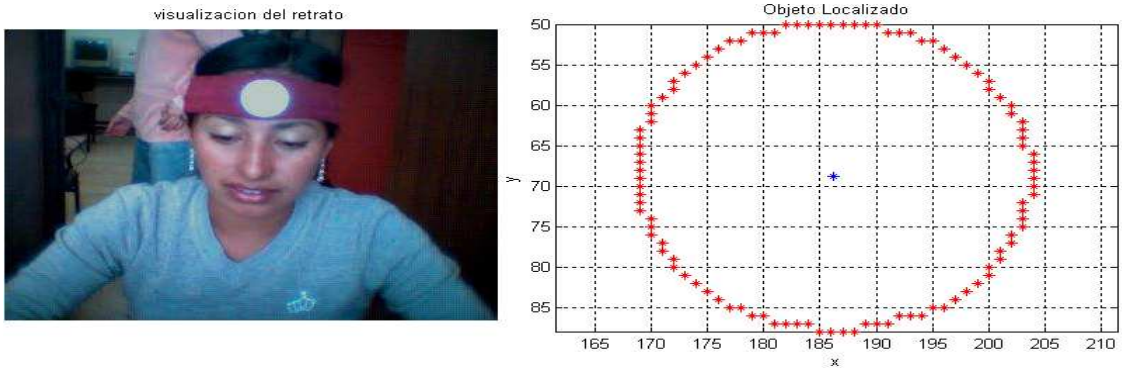
Ahora se procede a realizar las pruebas (figura 4.3) con menos iluminación donde la lectura del luxómetro indica 400 luxes. Y de la misma manera que en el anterior ejemplo se procede a realizar el cálculo del error relativo.



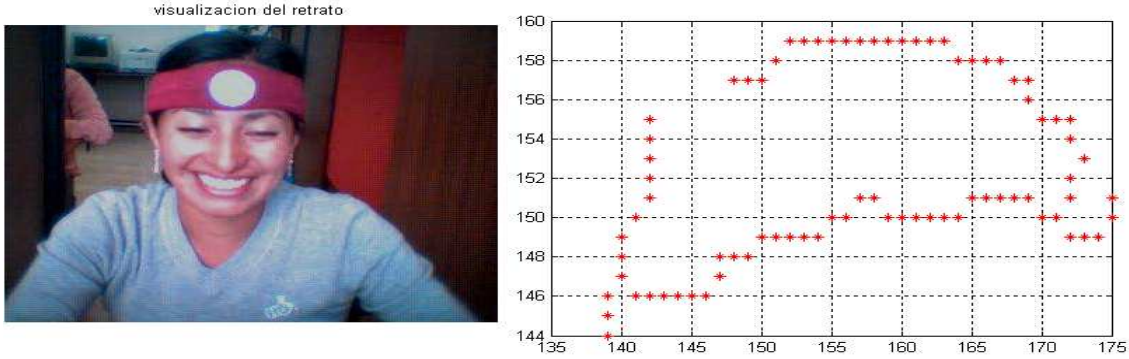
Resultado 1



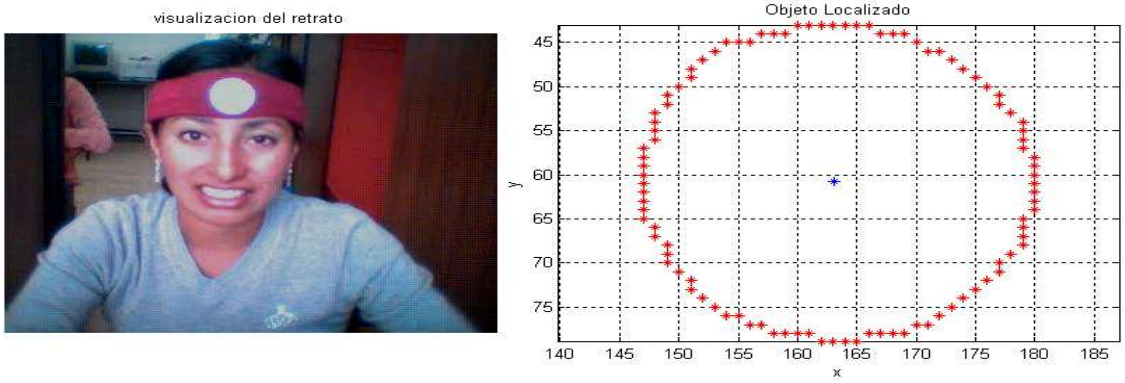
Resultado 2



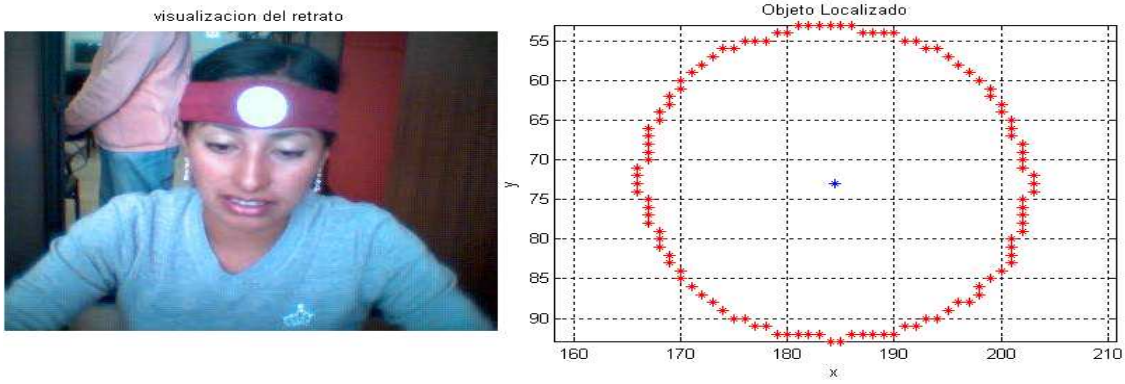
Resultado 3



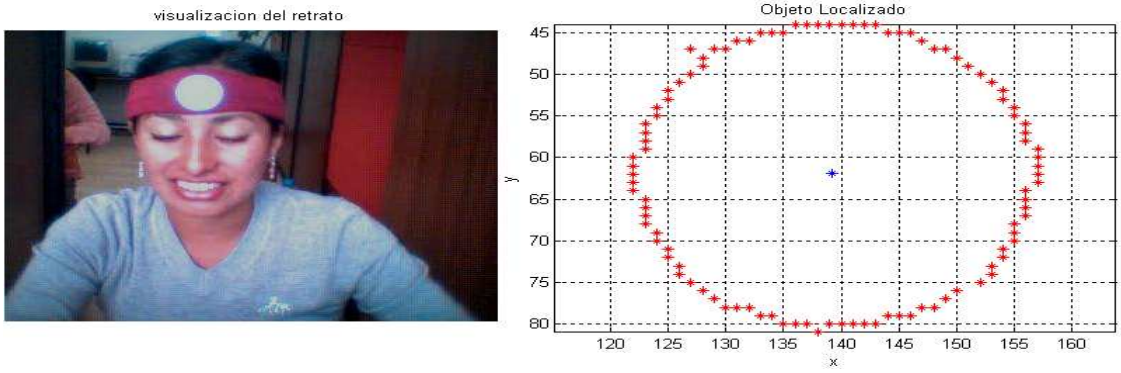
Resultado 4



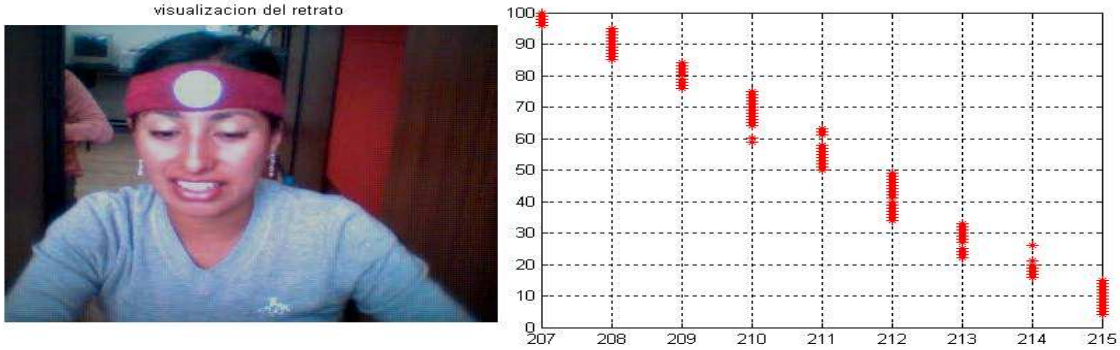
Resultado 5



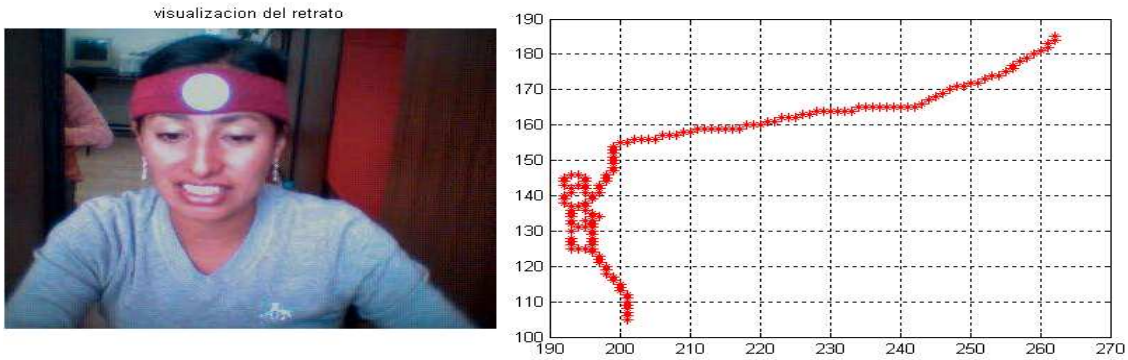
Resultado 6



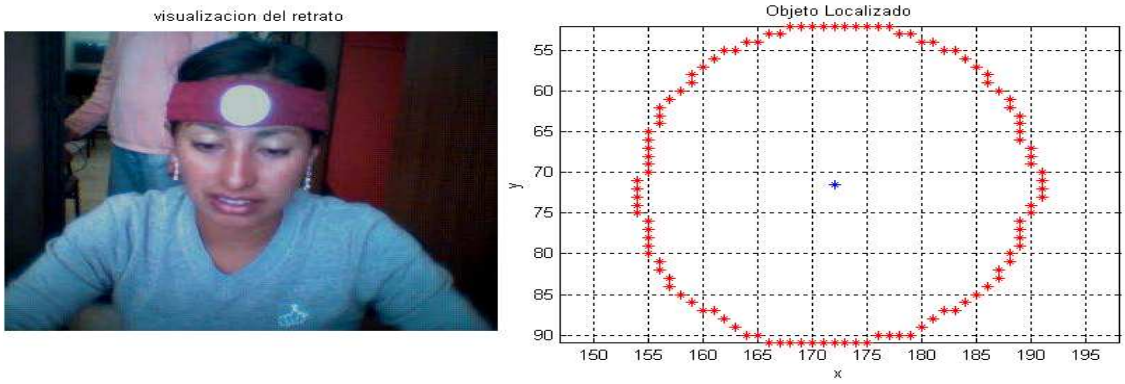
Resultado 7



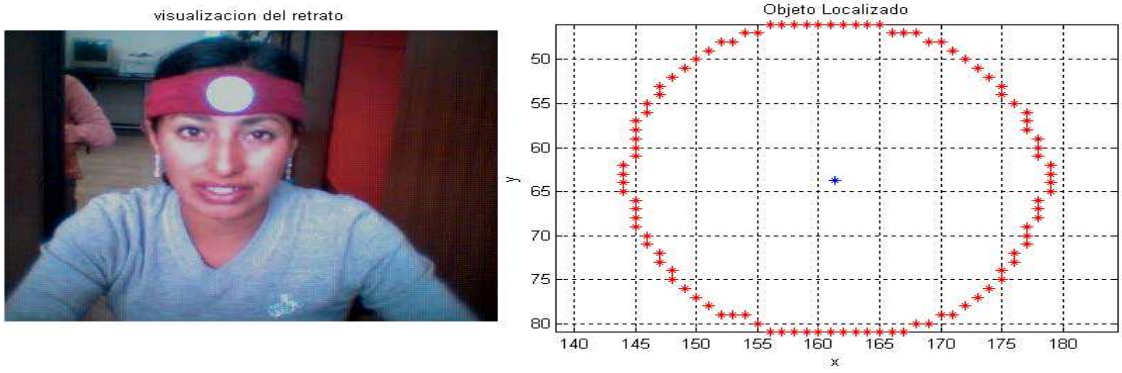
Resultado 8



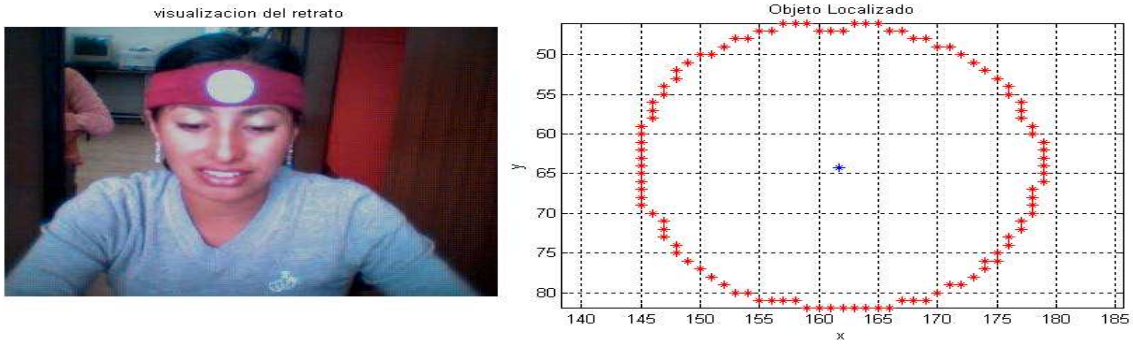
Resultado 9



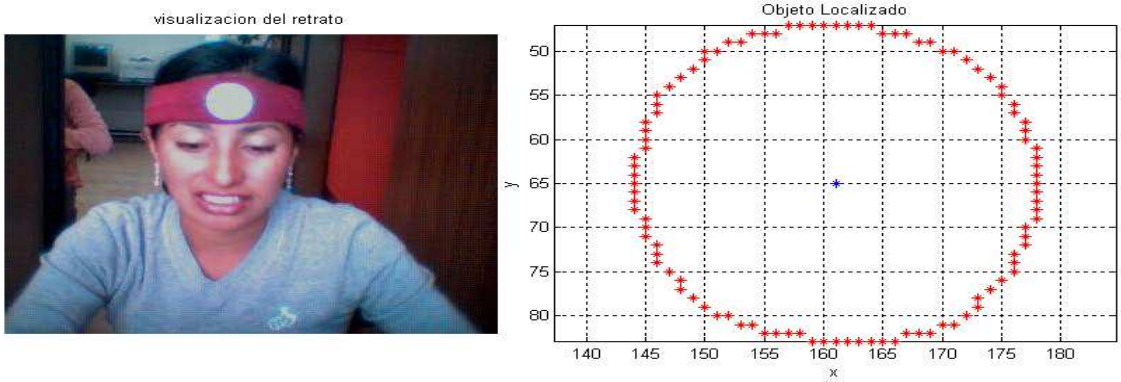
Resultado 10



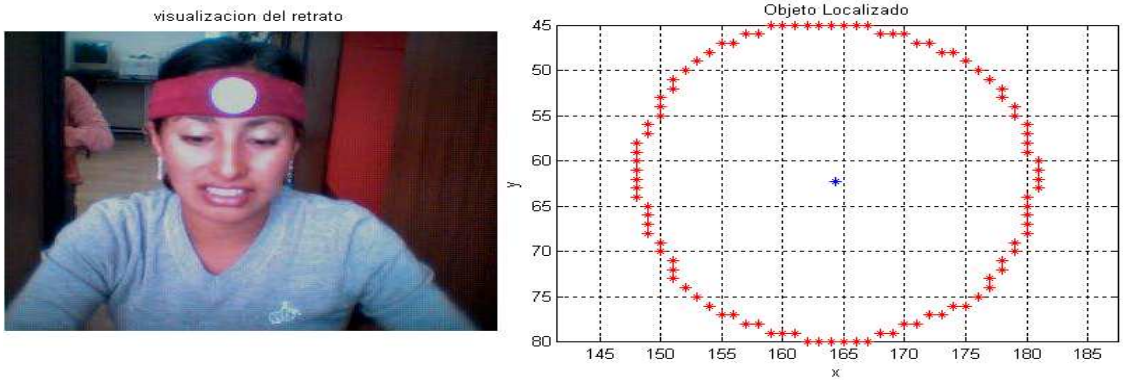
Resultado 11



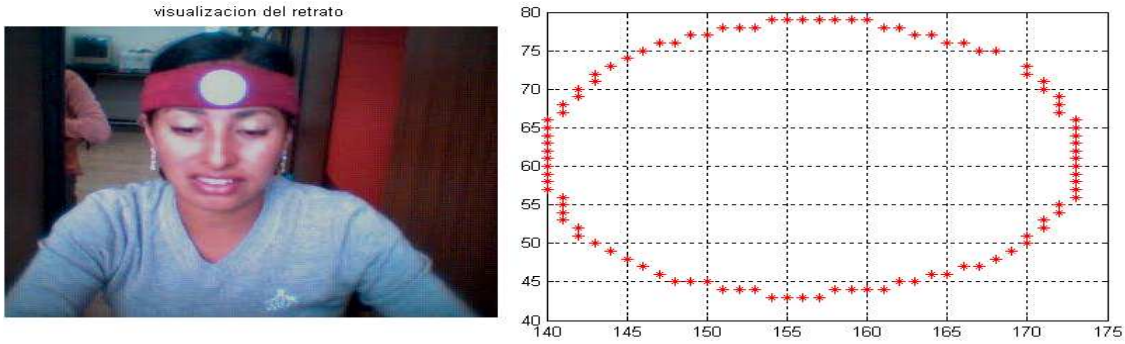
Resultado 12



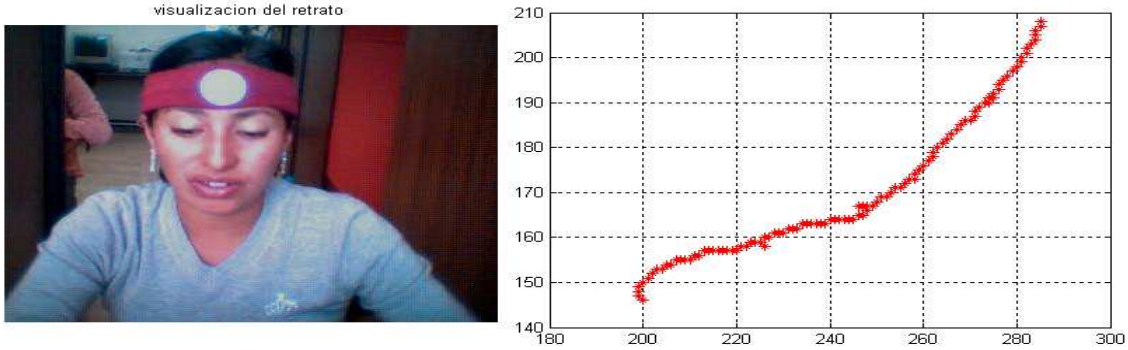
Resultado 13



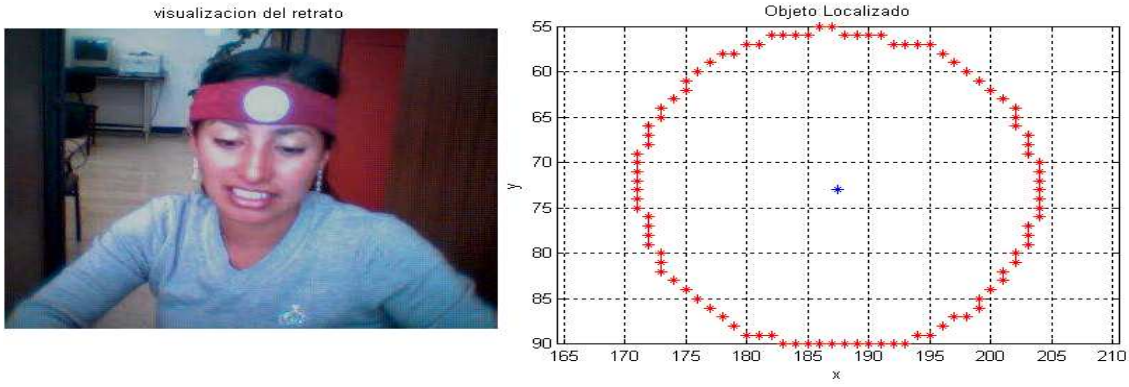
Resultado 14



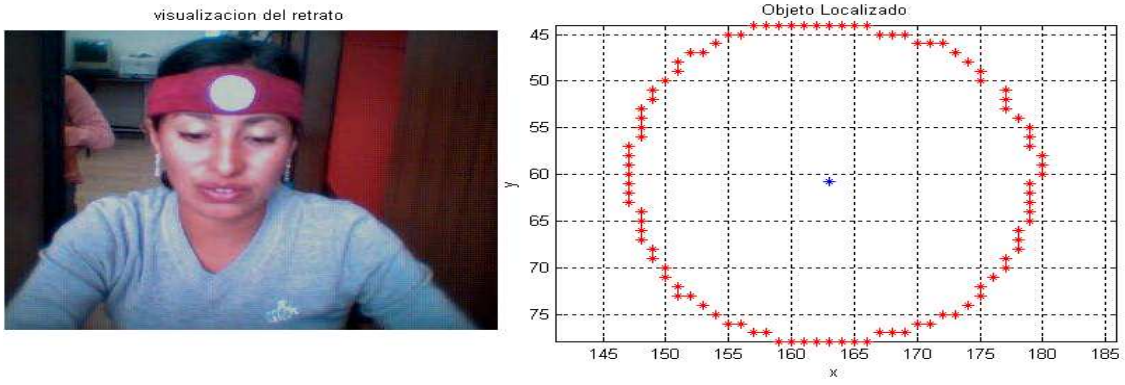
Resultado 15



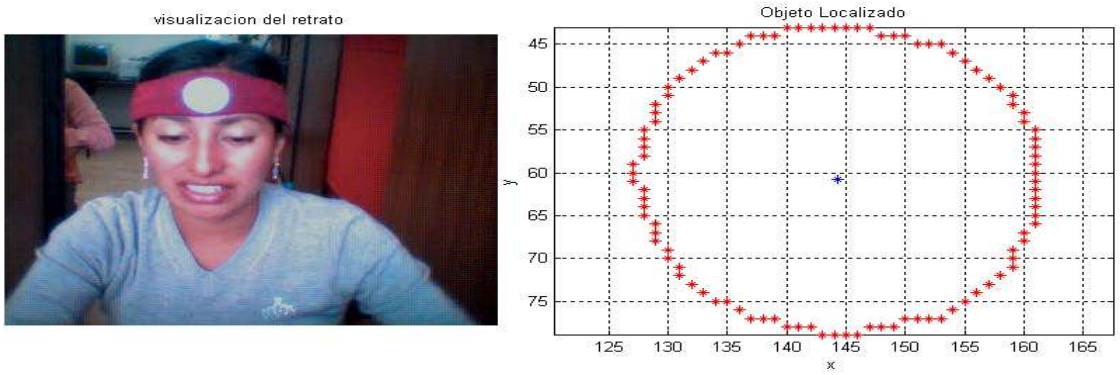
Resultado 16



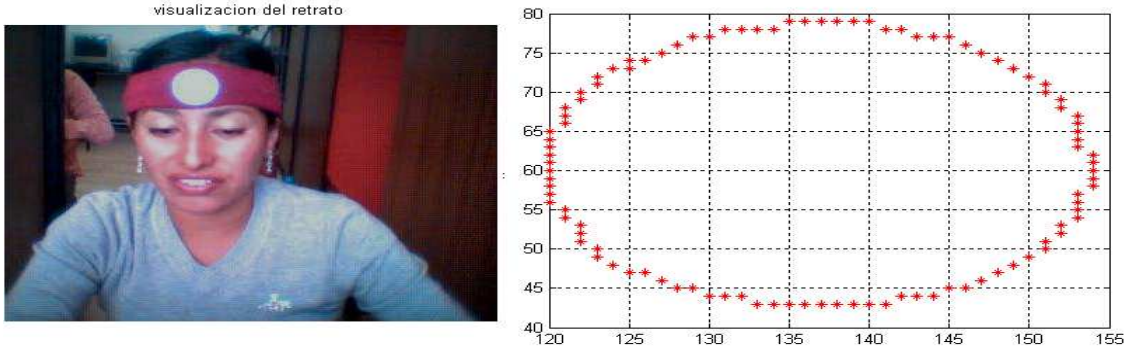
Resultado 17



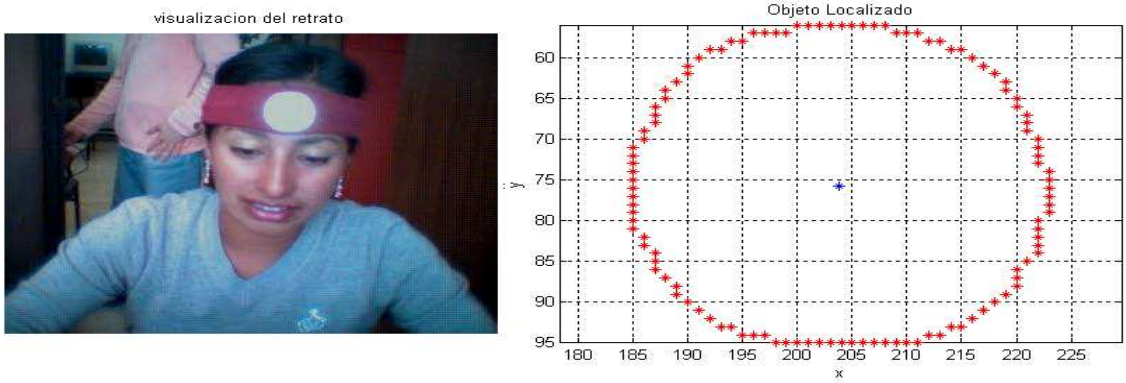
Resultado 18



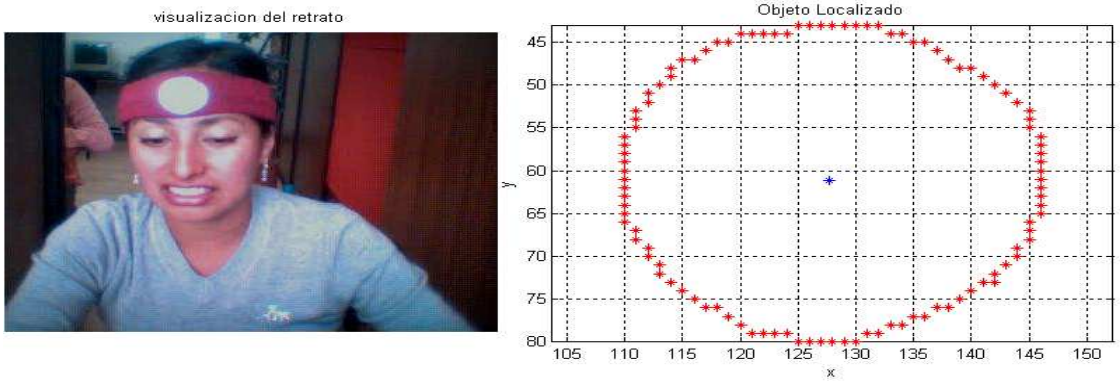
Resultado 19



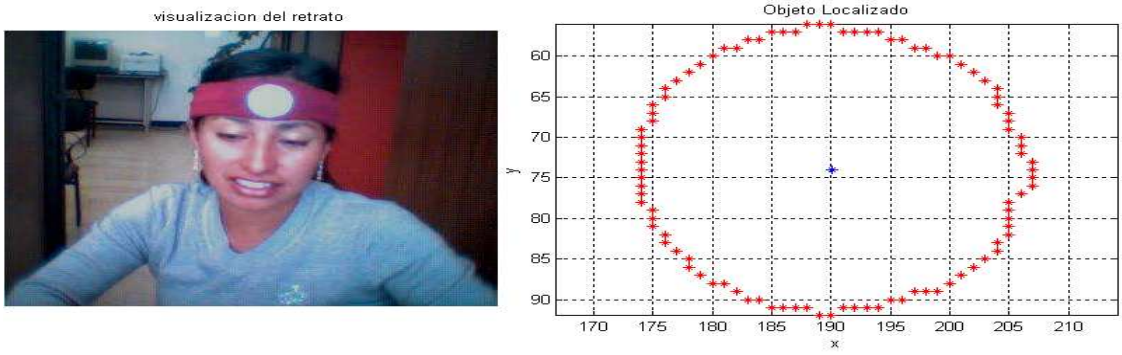
Resultado 20



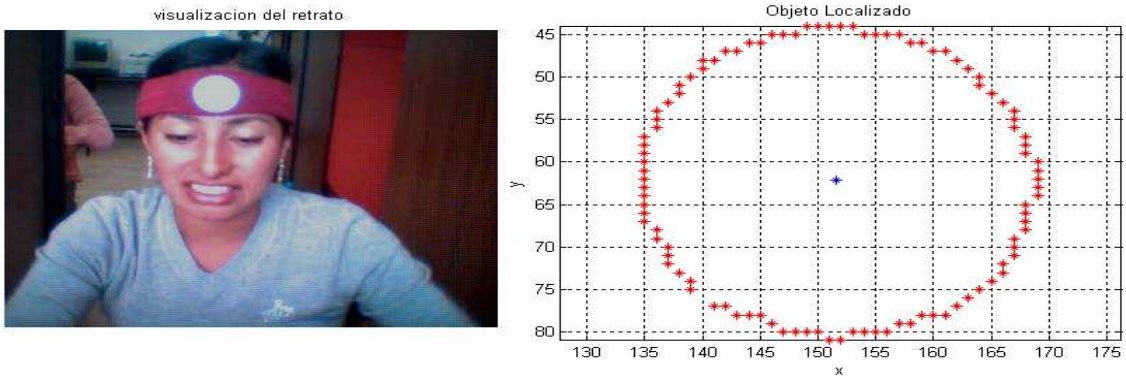
Resultado 21



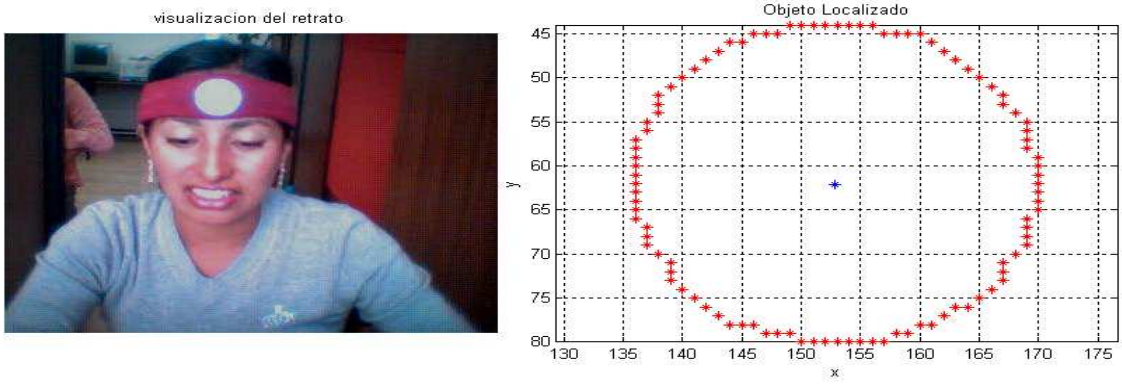
Resultado 22



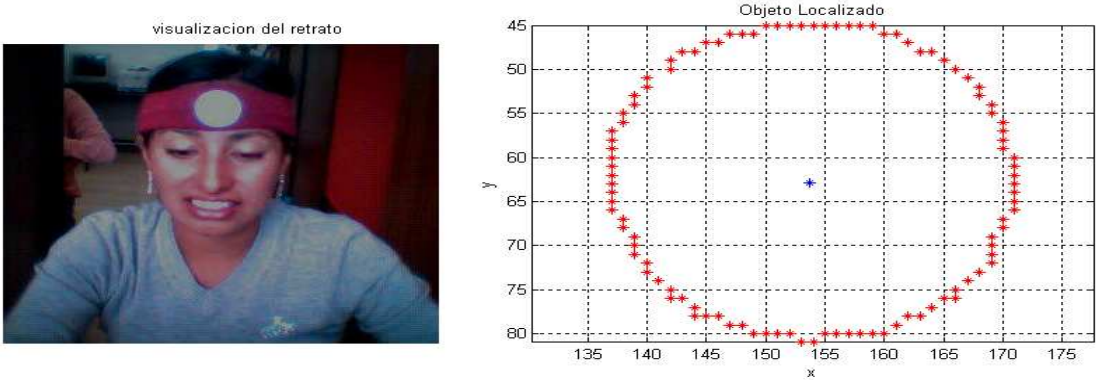
Resultado 23



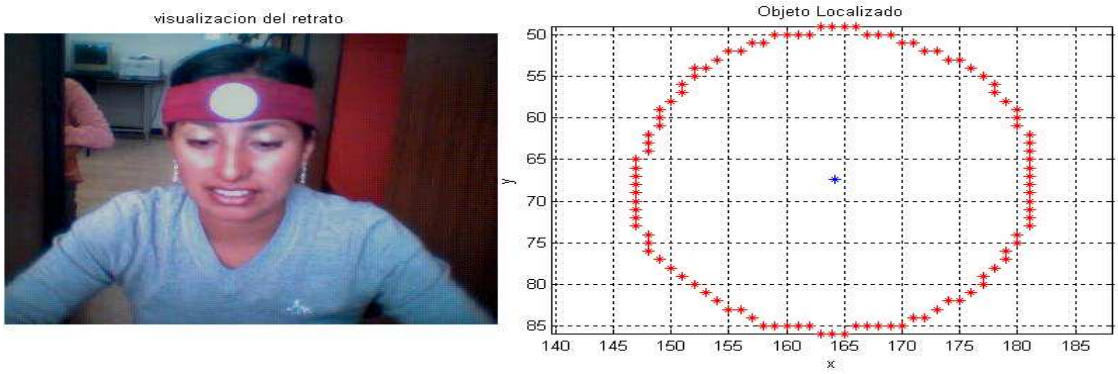
Resultado 24



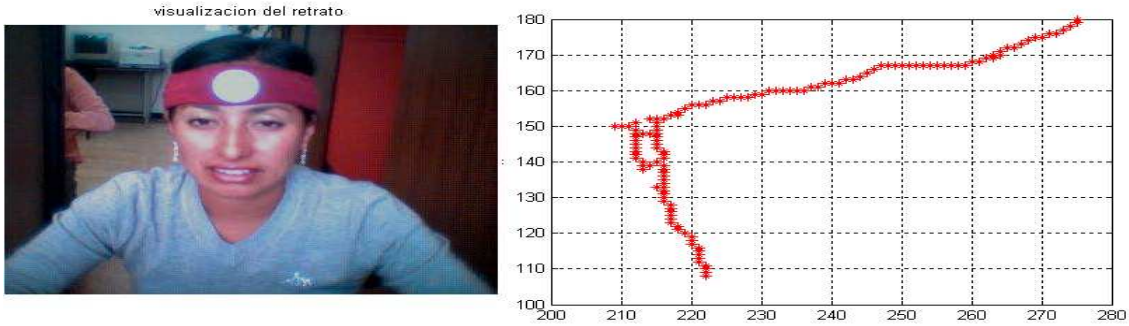
Resultado 25



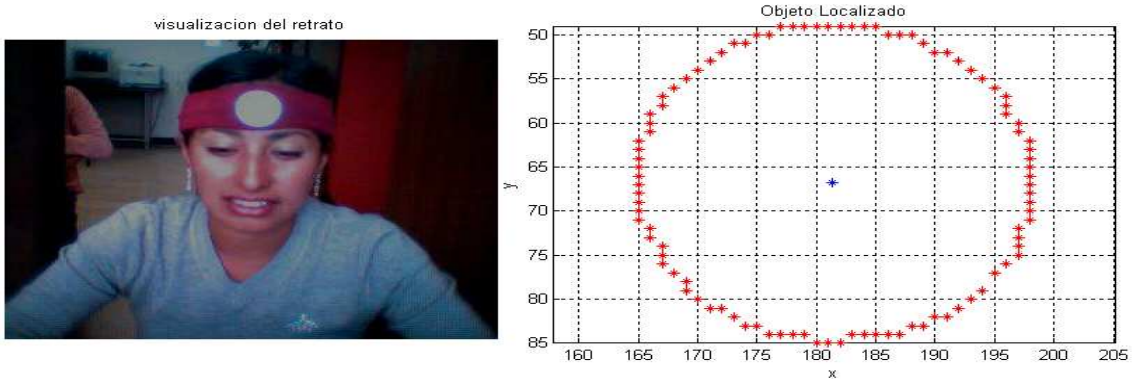
Resultado 26



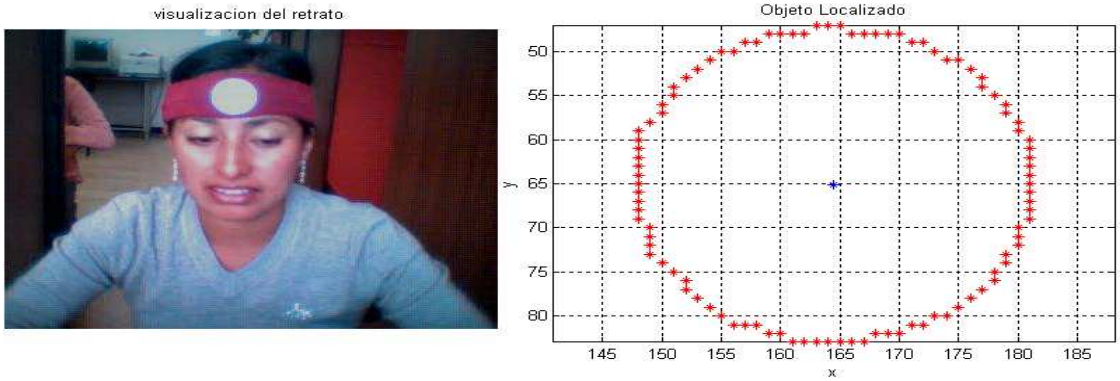
Resultado 27



Resultado 28



Resultado 29



Resultado 30

Figura 4. 3 Imágenes a 400 luxes.

El número de fallas a 400 luxes de las 30 imágenes es de 7 y ahora se procede a realizar el cálculo del error relativo.

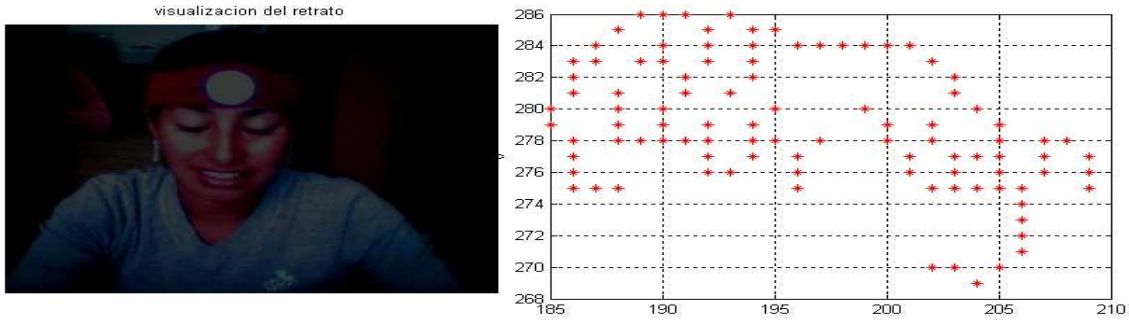
$$Er\% = \frac{V_r - V_m}{V_r} * 100$$

$$Er\% = \frac{30 - 23}{30} * 100$$

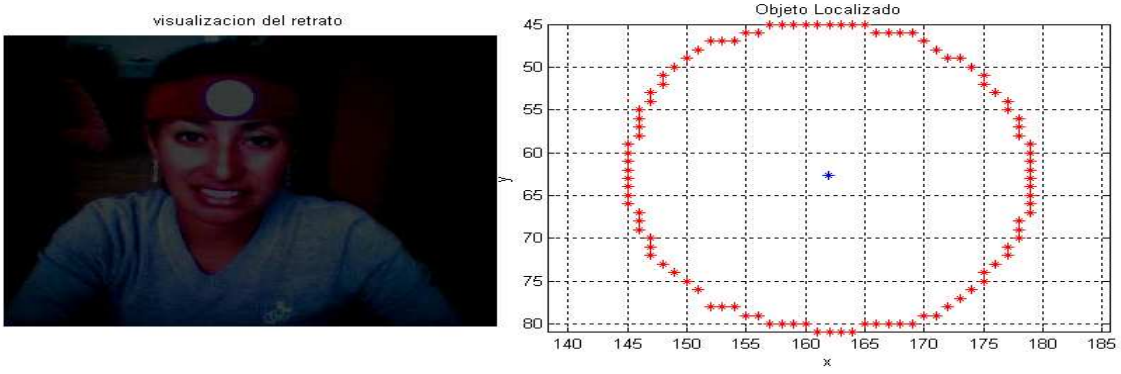
$$Er\% = 23.33\%$$

El error relativo porcentual que observamos en este caso es de 23.33%. Por lo que se puede notar conforme los niveles de iluminación disminuyen aumenta nuestro error relativo. Y esto se podrá confirmar en las siguientes pruebas.

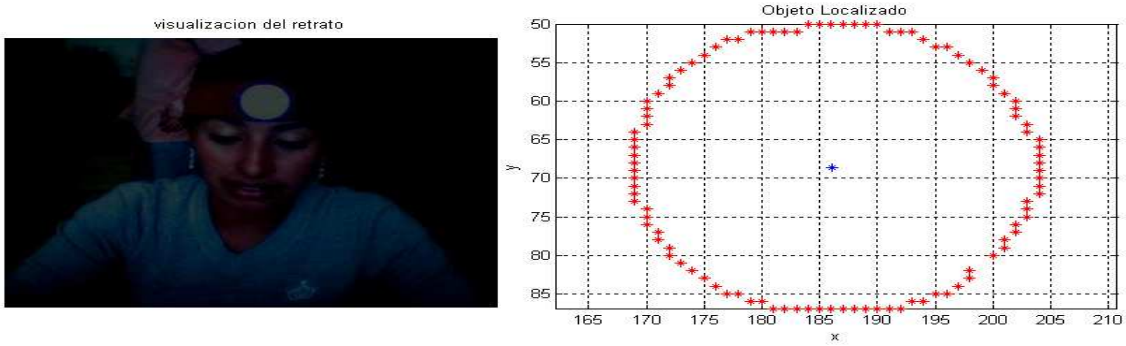
El nuevo valor que se ha medido con el luxómetro es de 150 luxes las cuales se pueden ver en la figura 4.4.



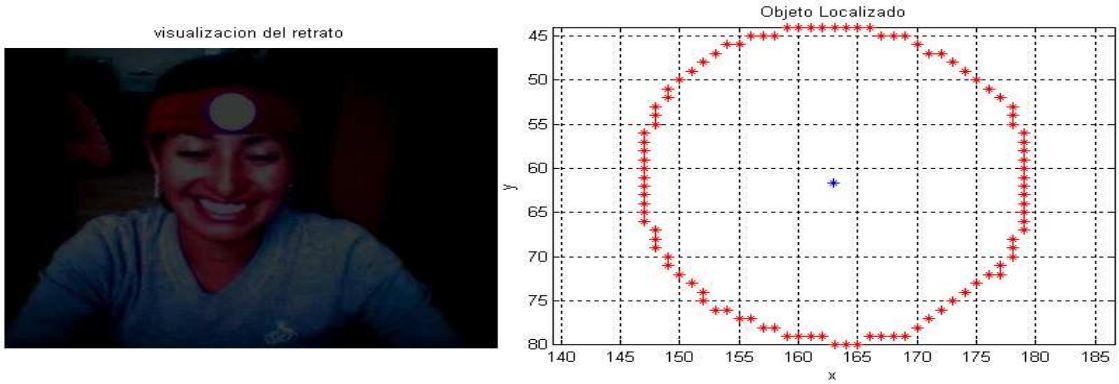
Resultado 1



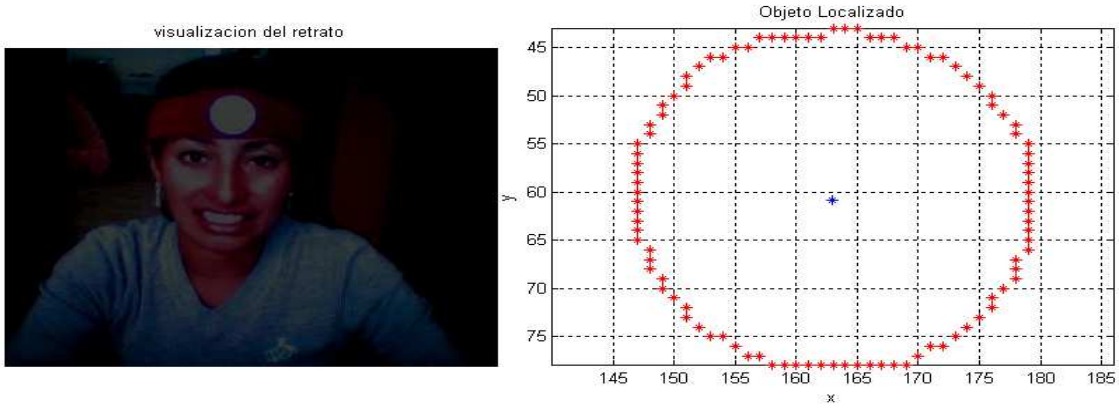
Resultado 2



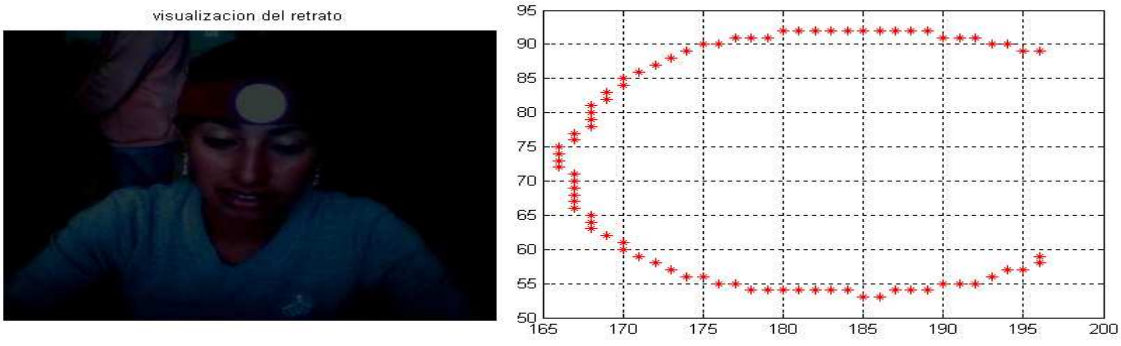
Resultado 3



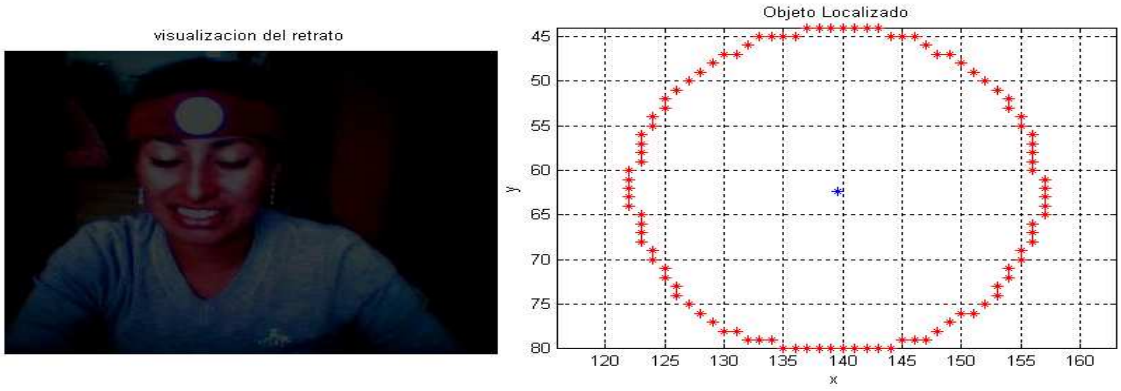
Resultado 4



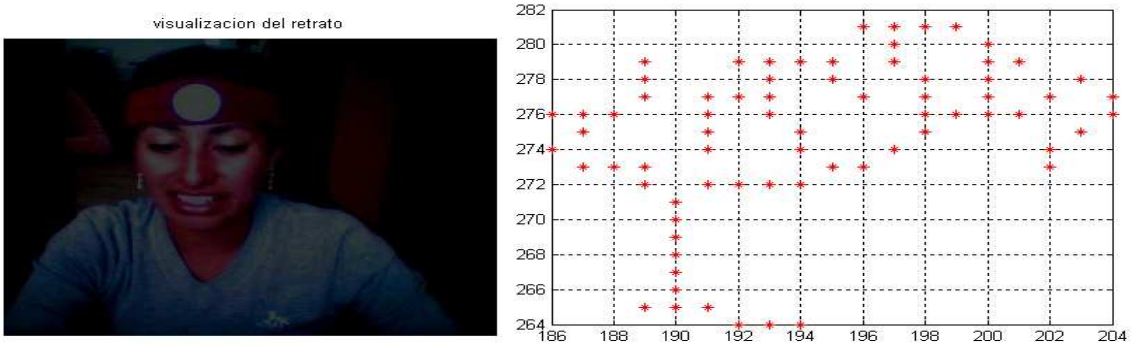
Resultado 5



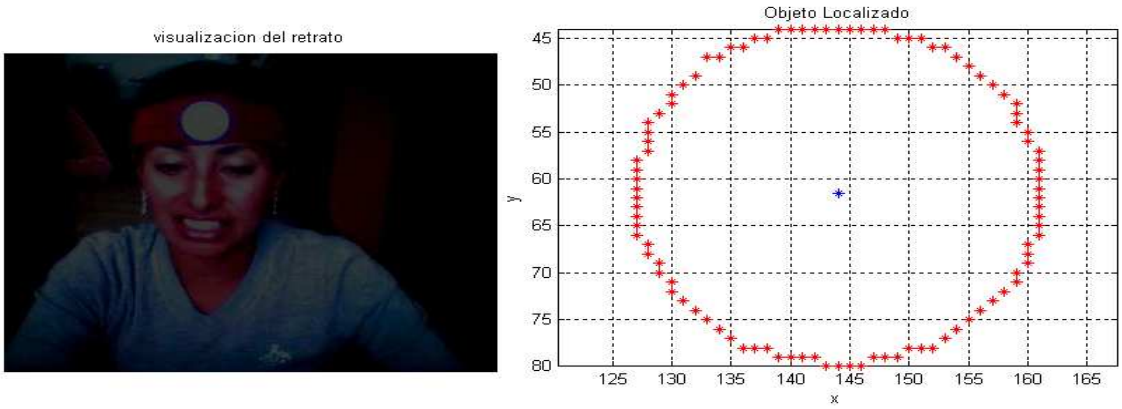
Resultado 6



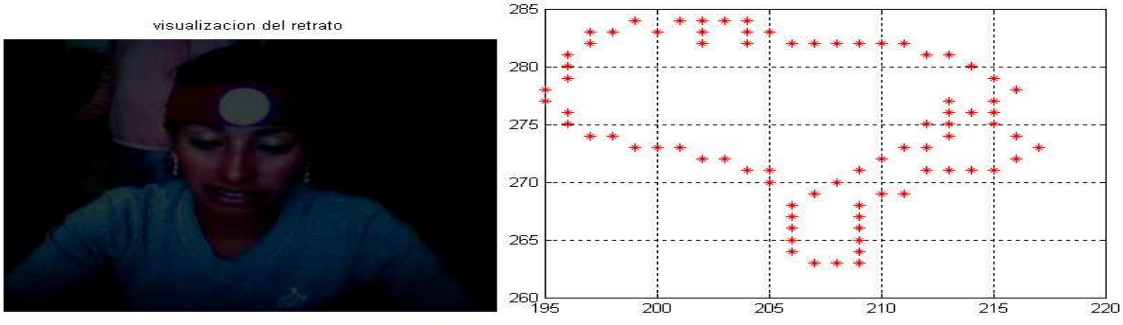
Resultado 7



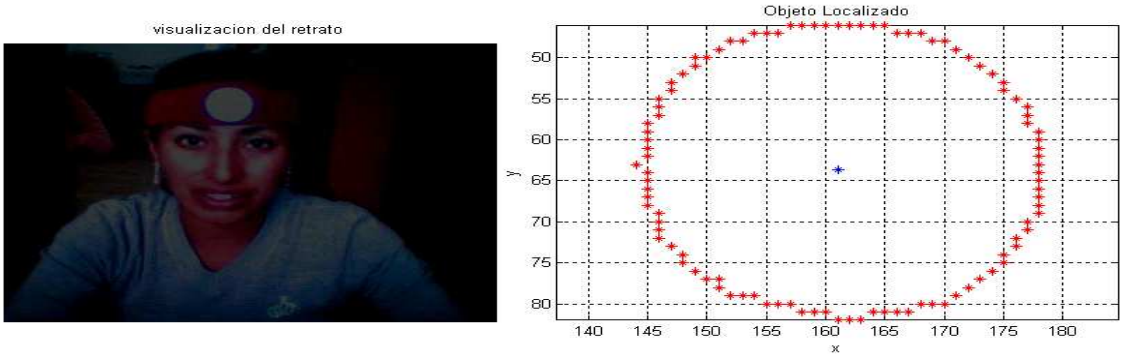
Resultado 8



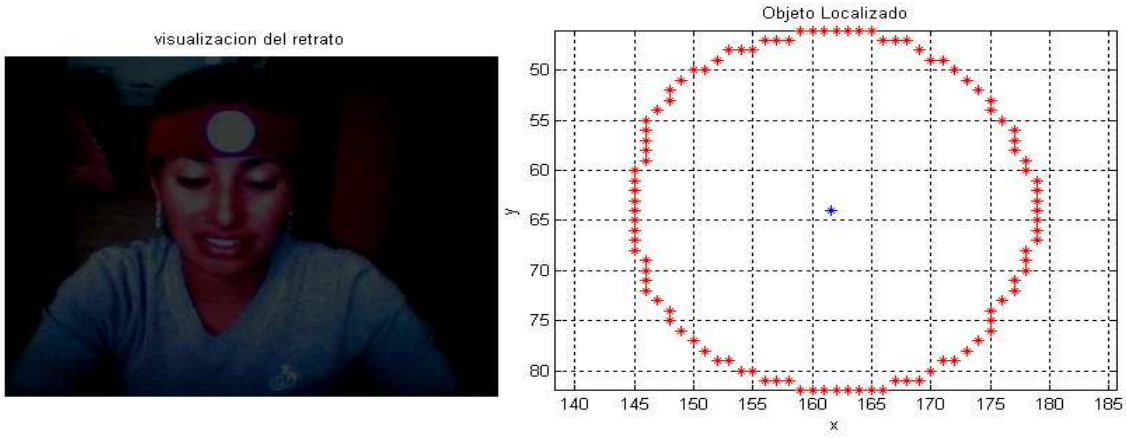
Resultado 9



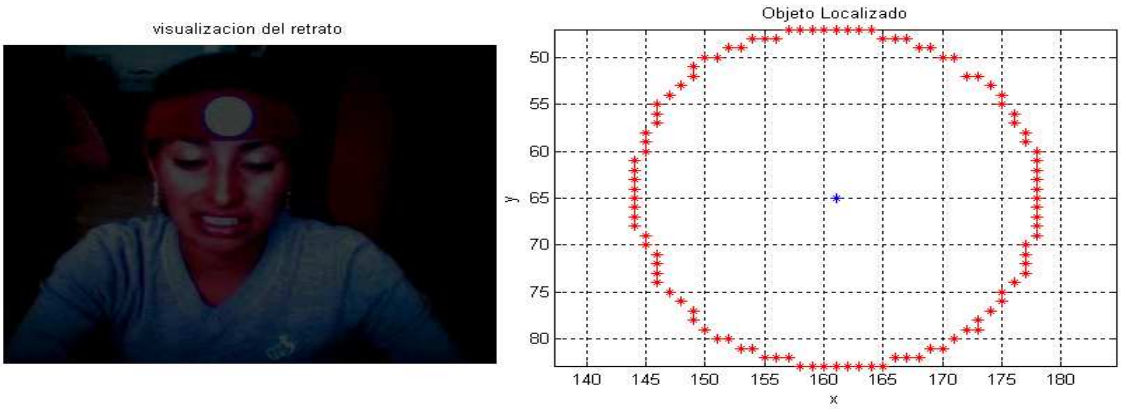
Resultado 10



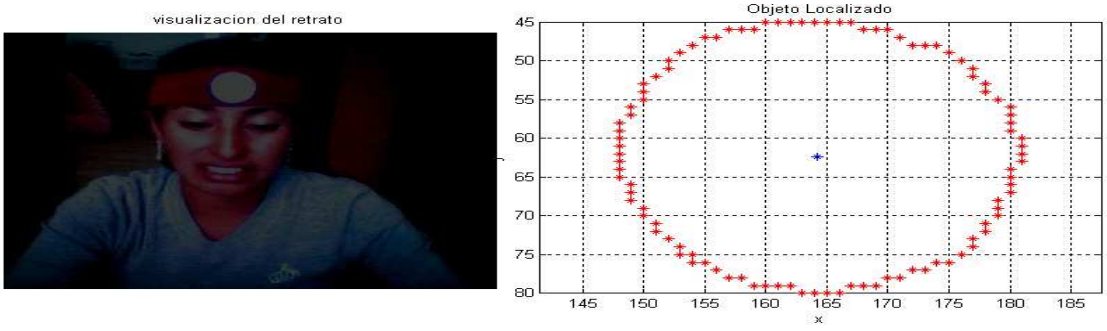
Resultado 11



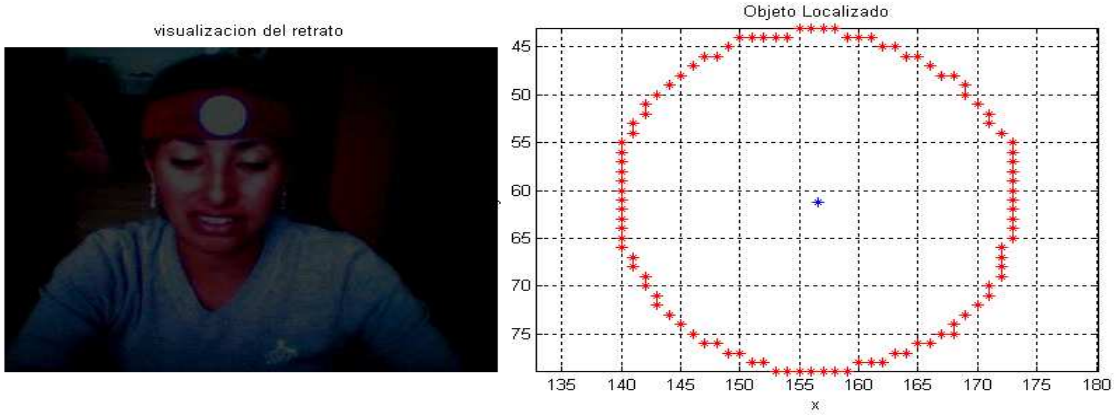
Resultado 12



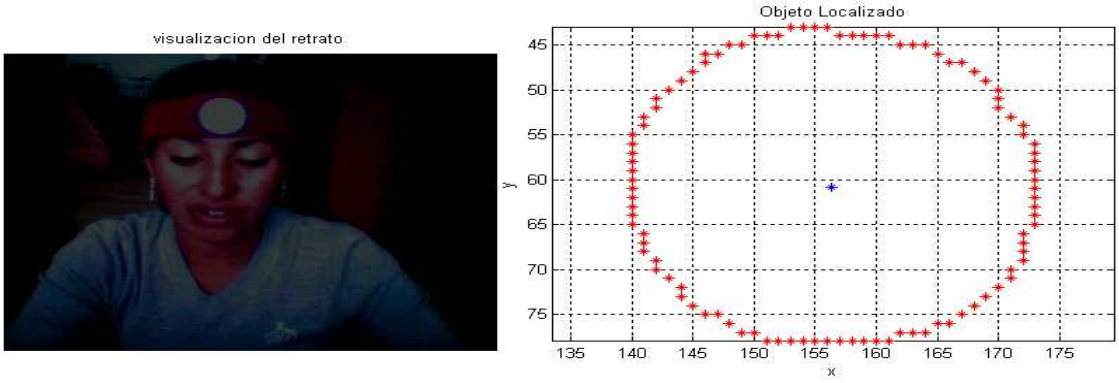
Resultado 13



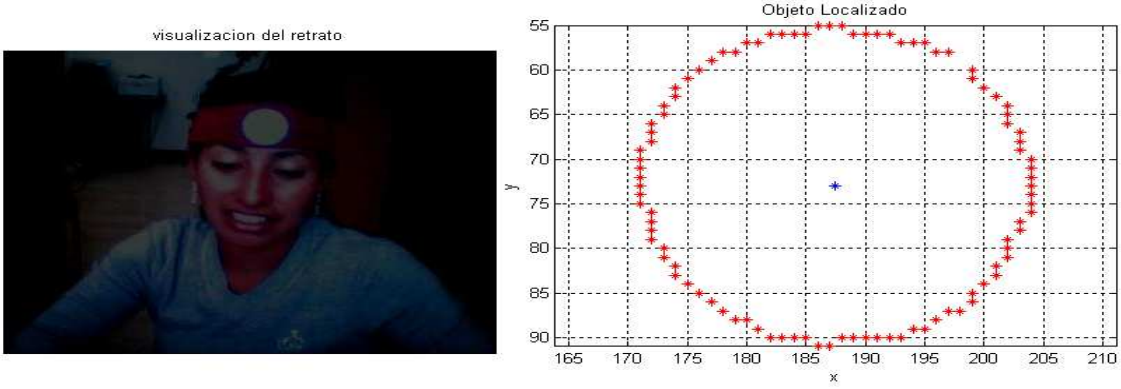
Resultado 14



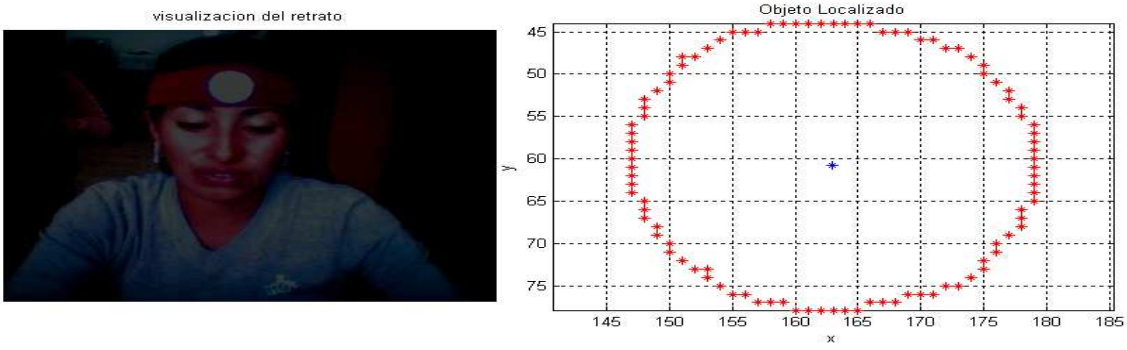
Resultado 15



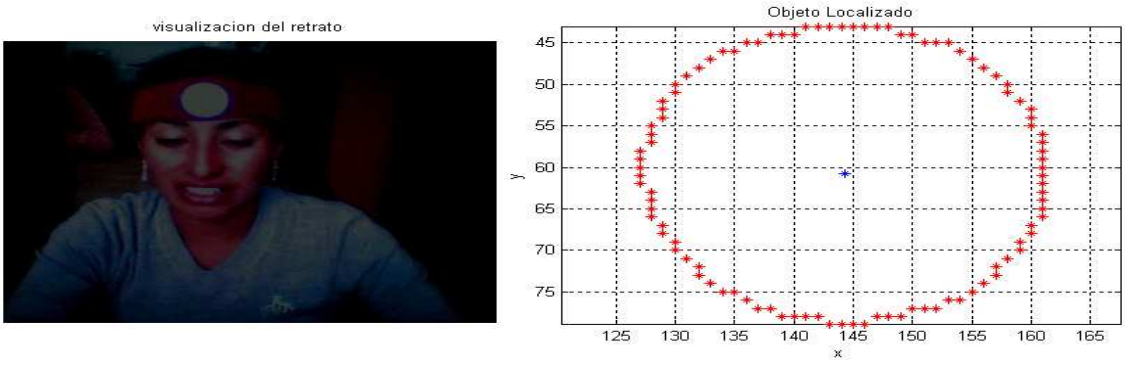
Resultado 16



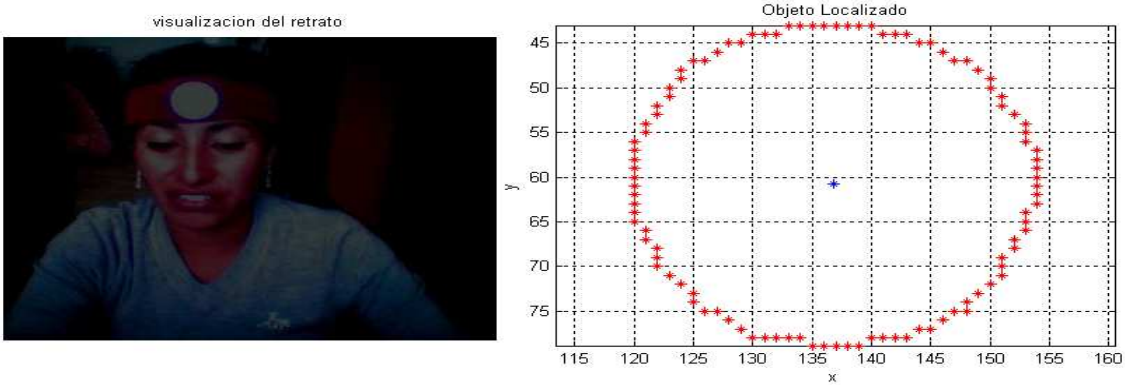
Resultado 17



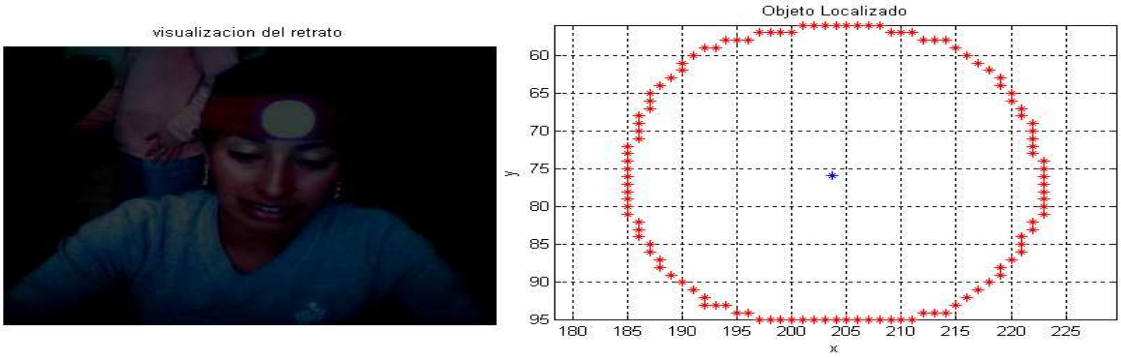
Resultado 18



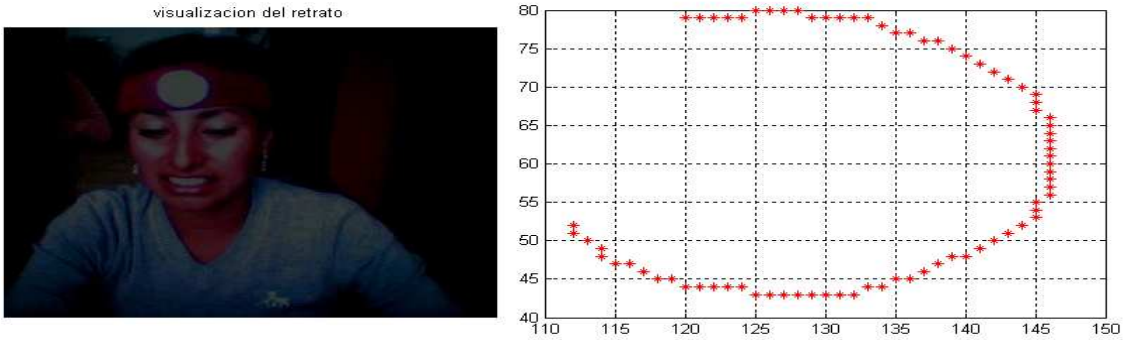
Resultado 19



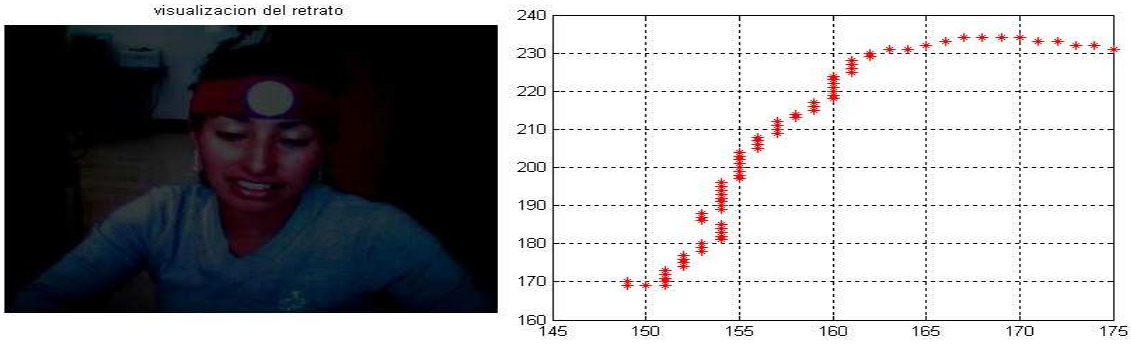
Resultado 20



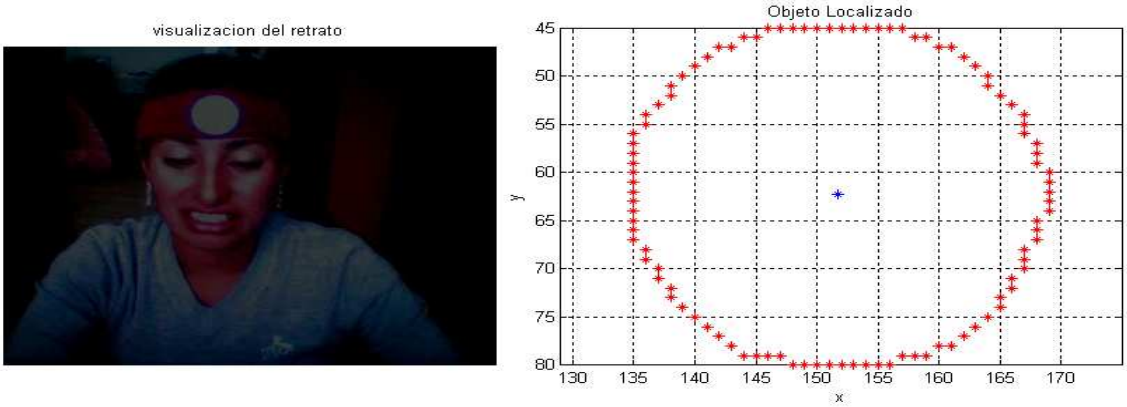
Resultado 21



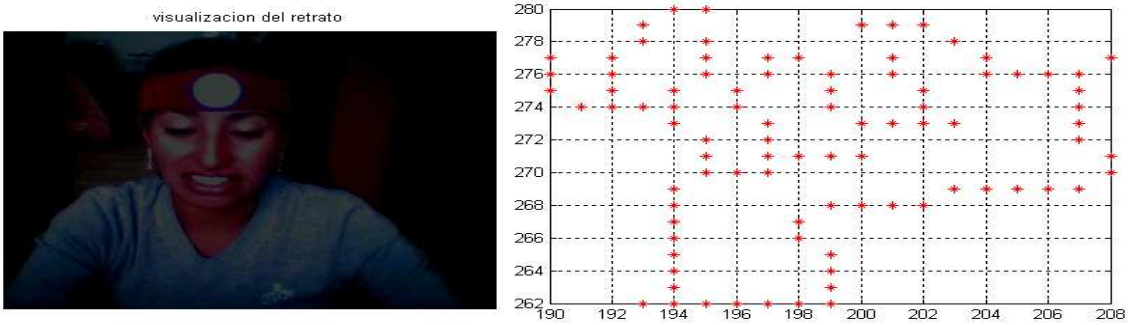
Resultado 22



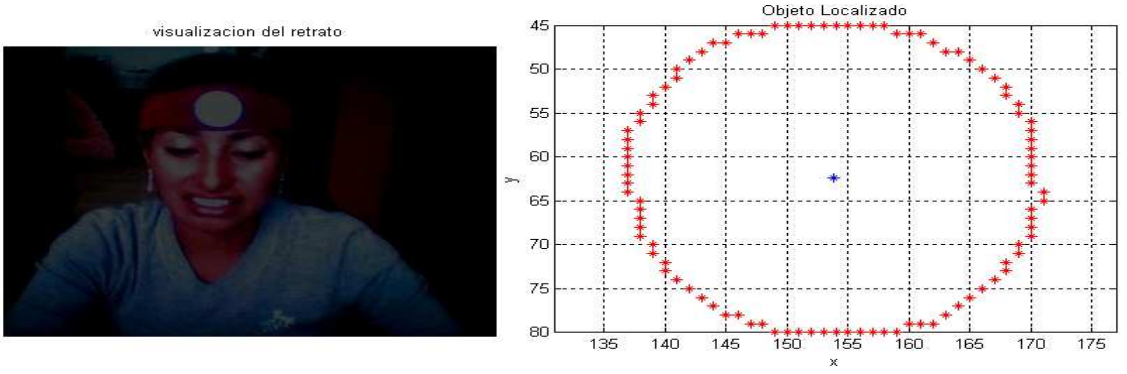
Resultado 23



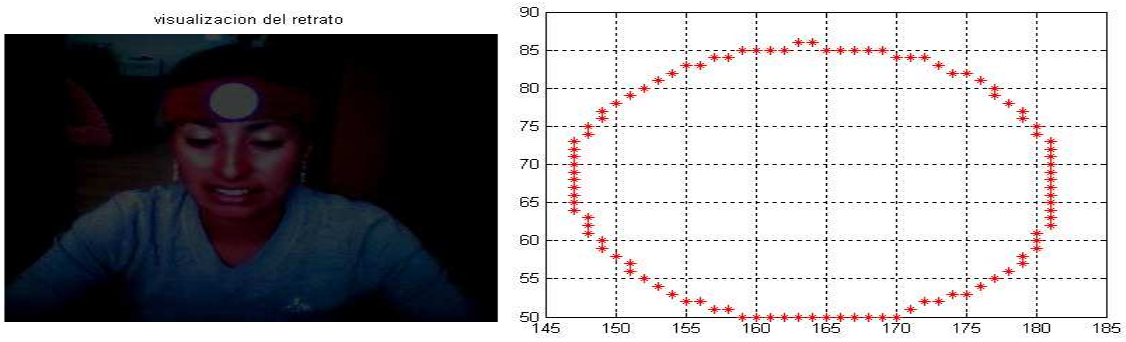
Resultado 24



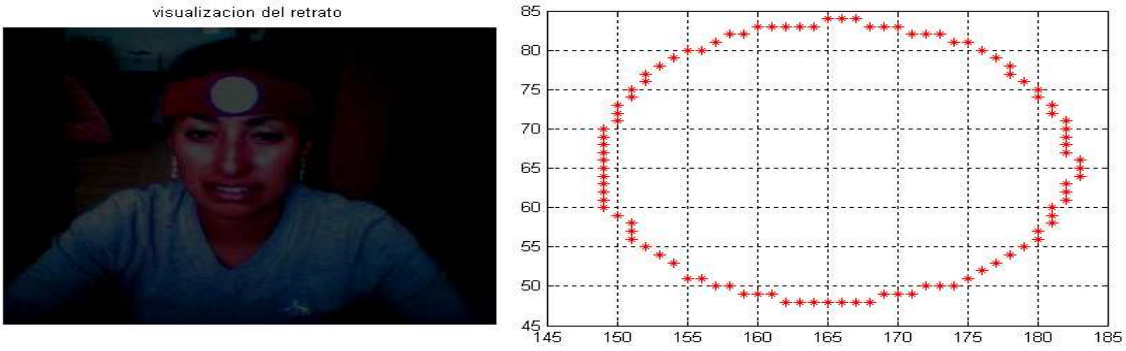
Resultado 25



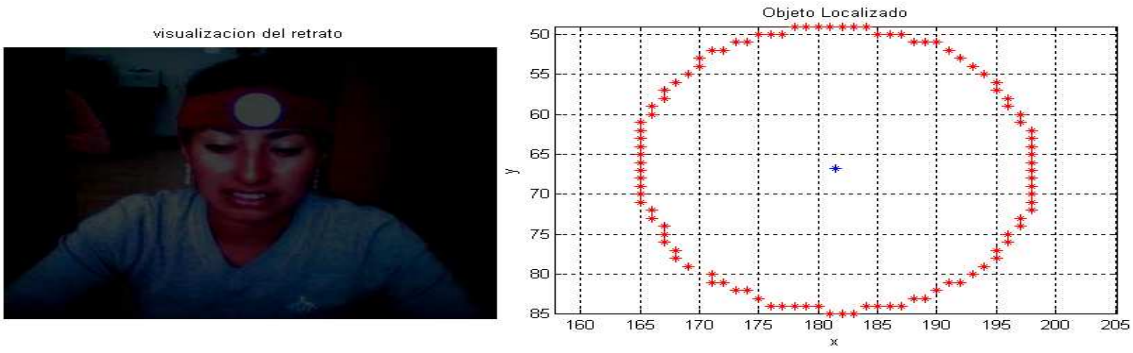
Resultado 26



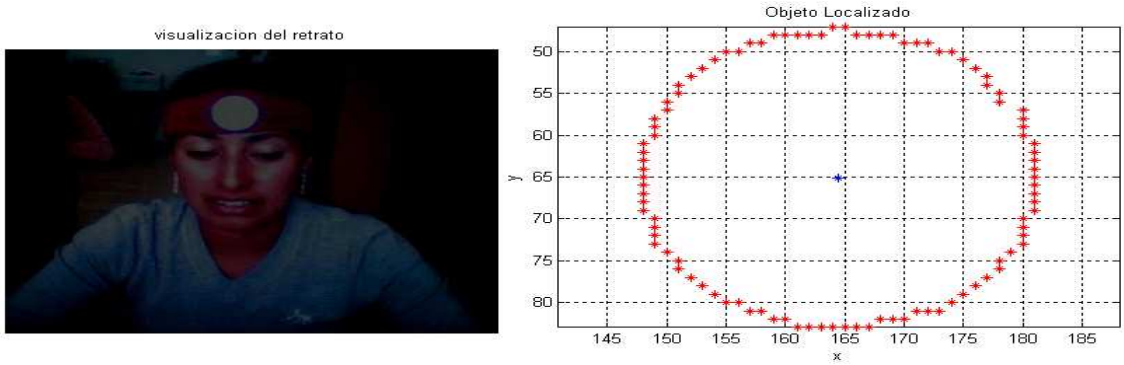
Resultado 27



Resultado 28



Resultado 29



Resultado 30

Figura 4. 4 Imágenes a 150 luxes

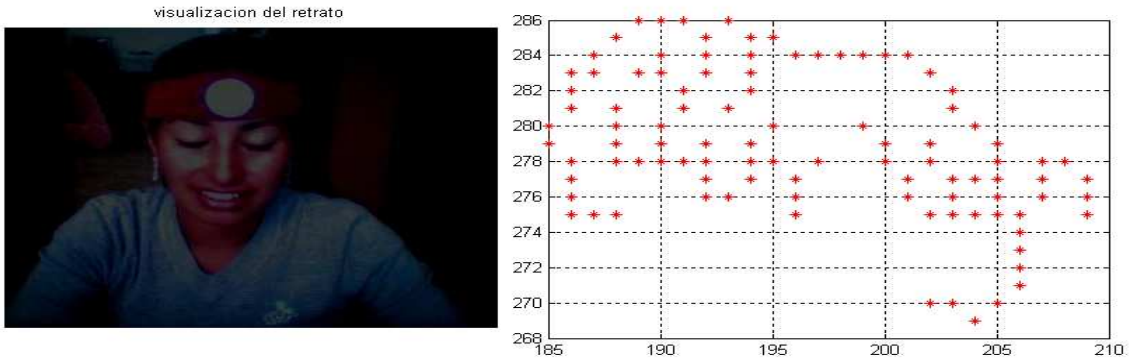
El número de fallas a 150 luxes de las 30 imágenes de prueba son 9 y ahora se procede a realizar el cálculo del error relativo.

$$Er\% = \frac{V_r - V_m}{V_r} * 100$$

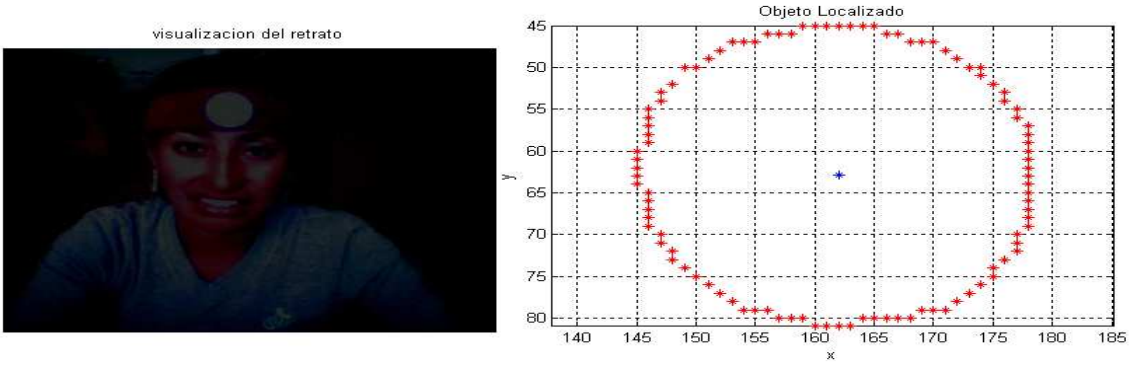
$$Er\% = \frac{30 - 21}{30} * 100$$

$$Er\% = 30\%$$

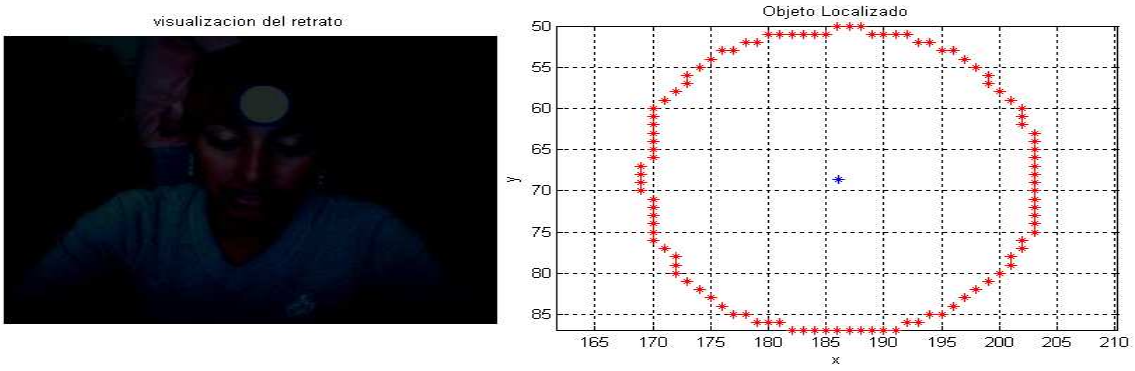
Con 100 luxes se procede a realizar las siguientes pruebas mostradas en la figura 4.5.



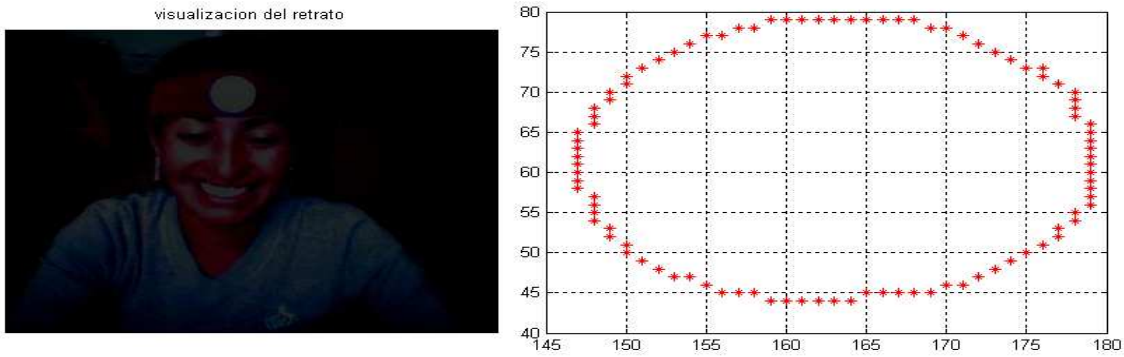
Resultado 1



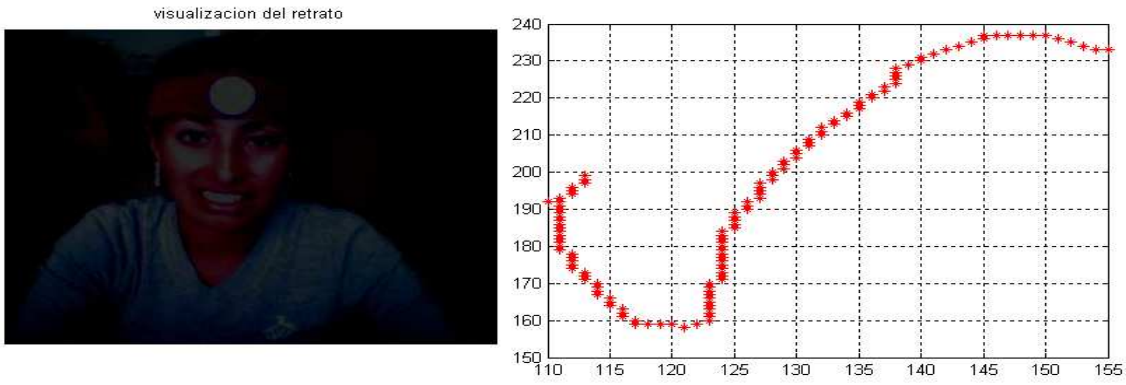
Resultado 2



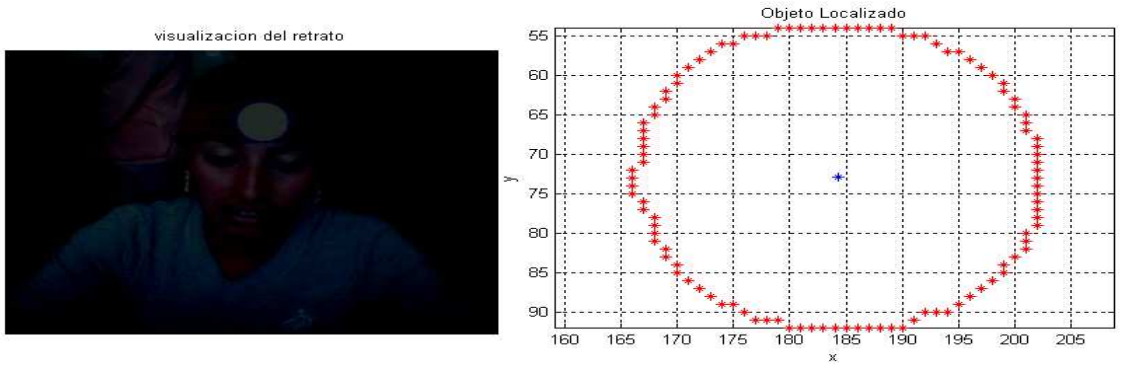
Resultado 3



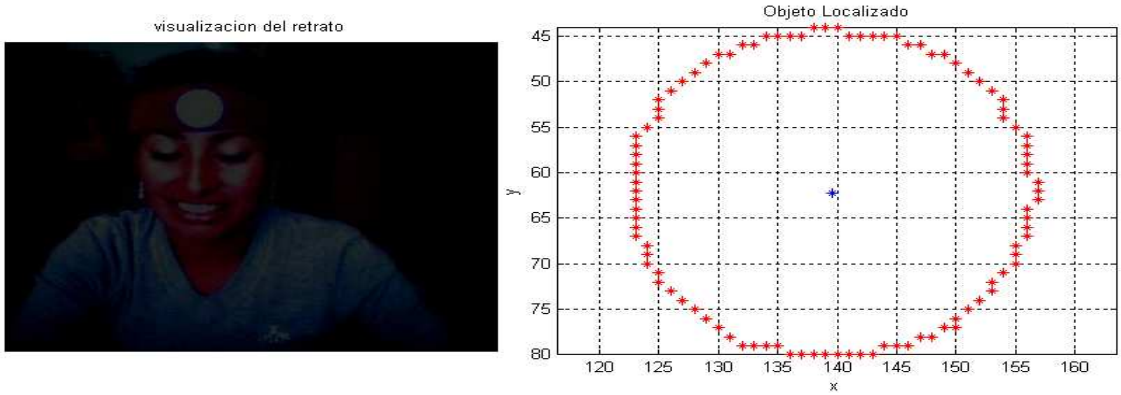
Resultado 4



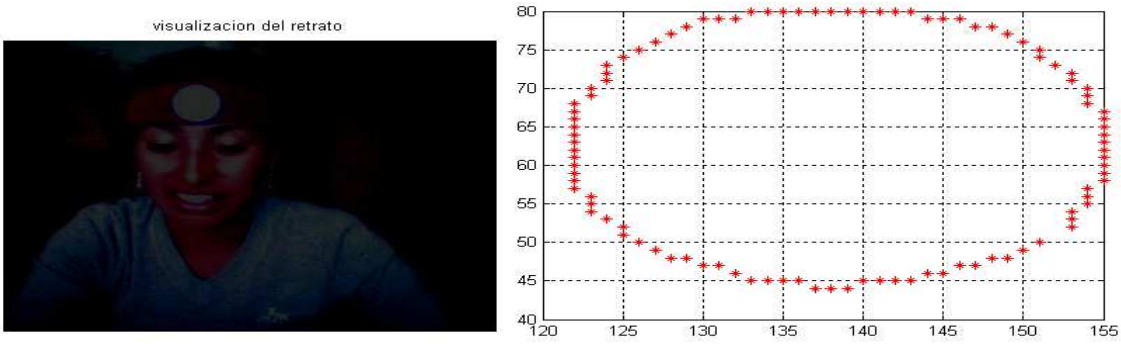
Resultado 5



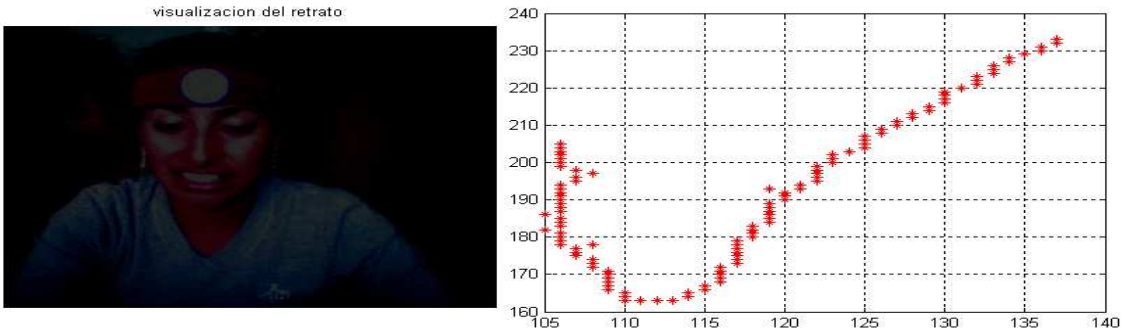
Resultado 6



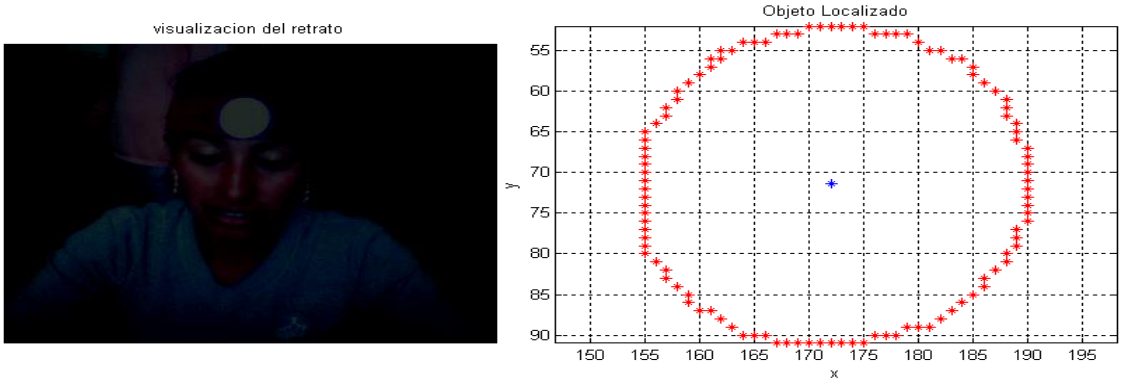
Resultado 7



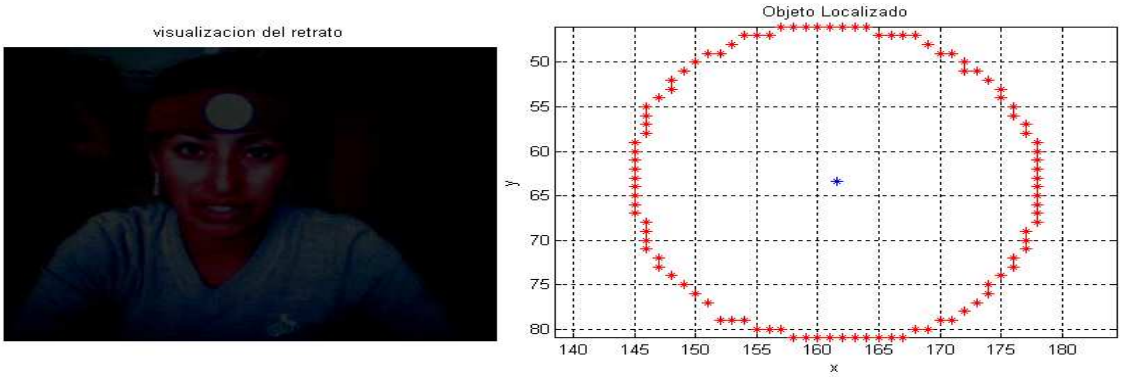
Resultado 8



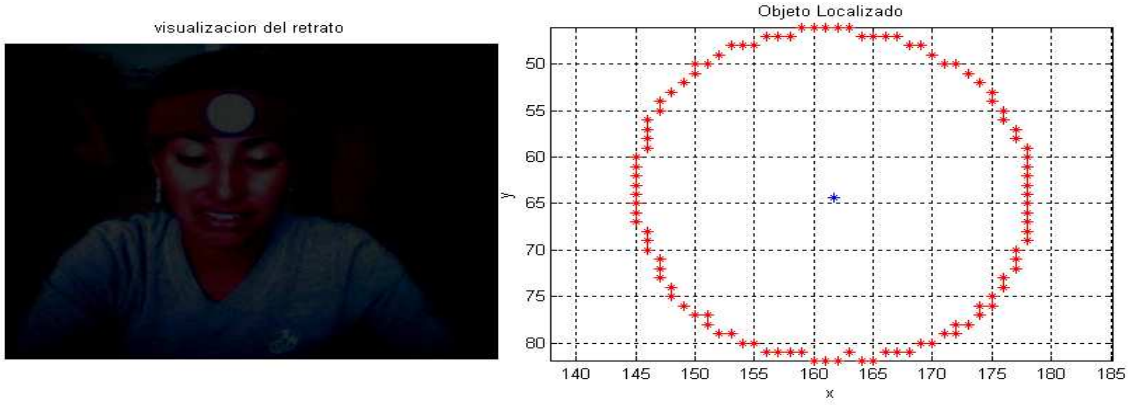
Resultado 9



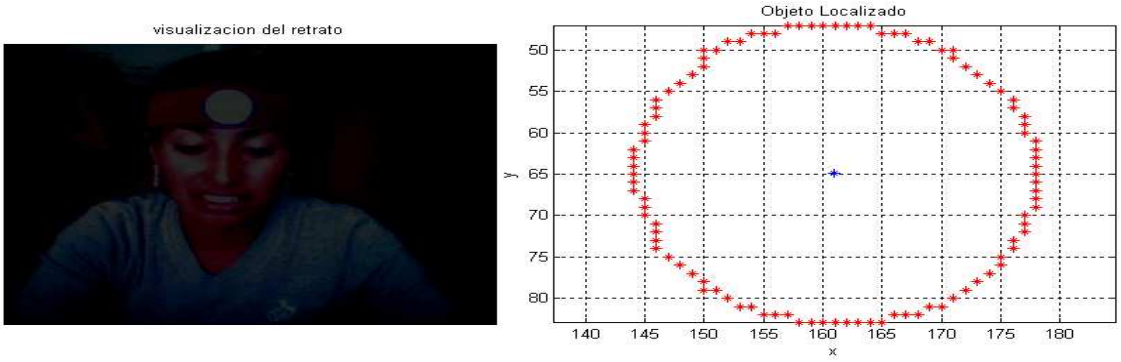
Resultado 10



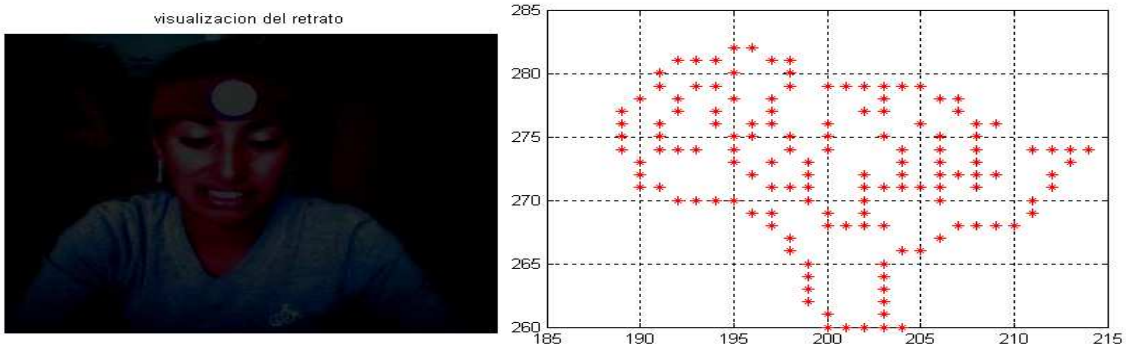
Resultado 11



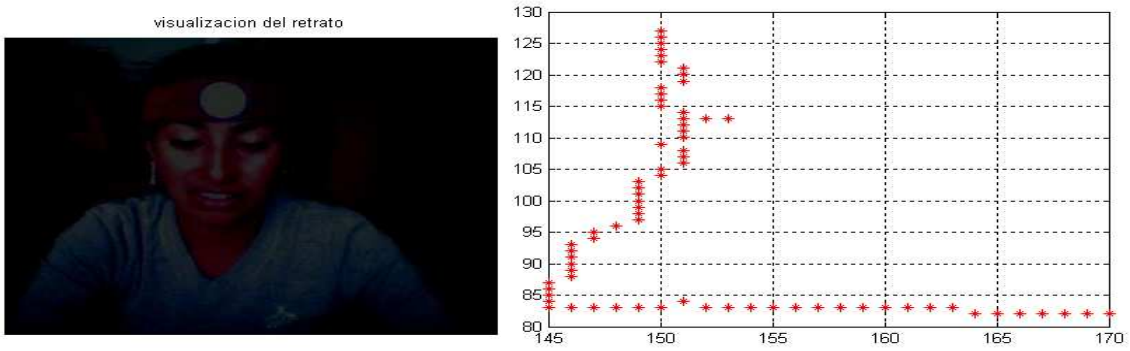
Resultado 12



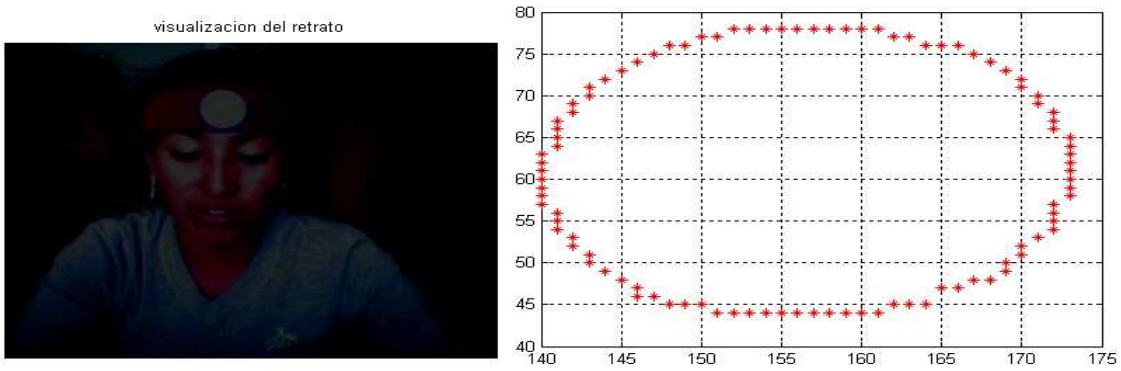
Resultado 13



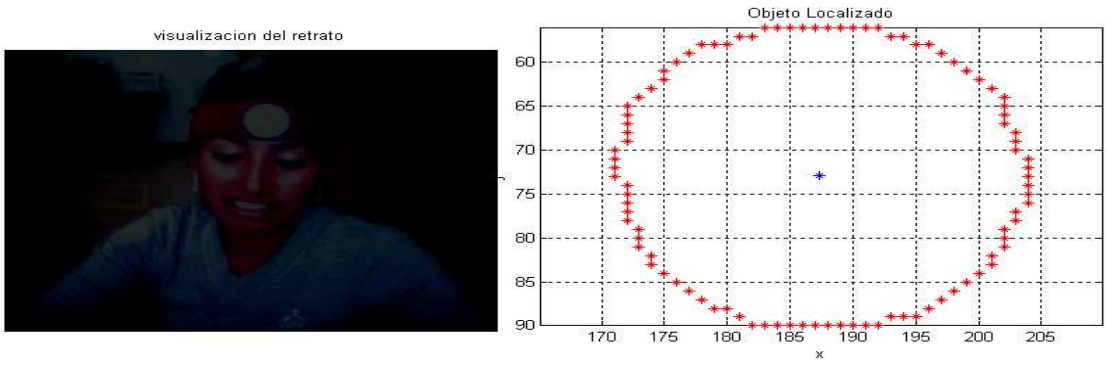
Resultado 14



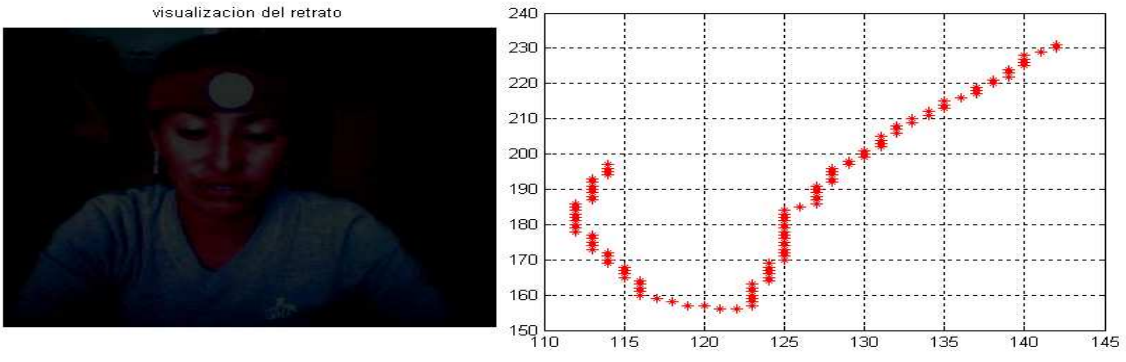
Resultado 15



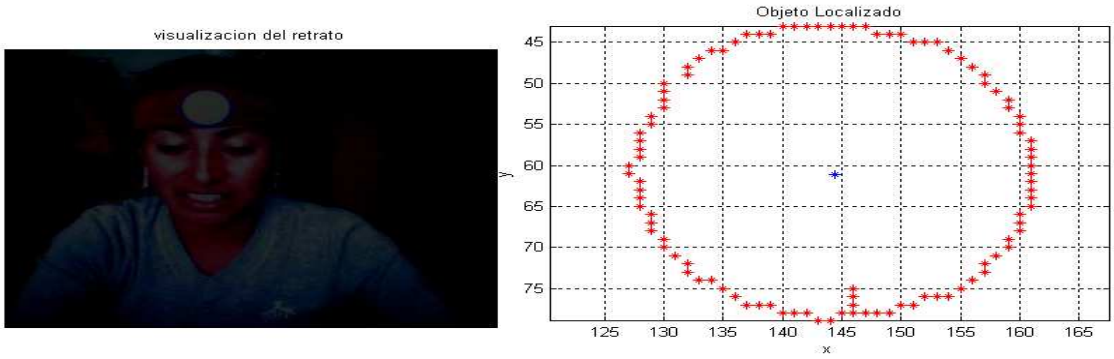
Resultado 16



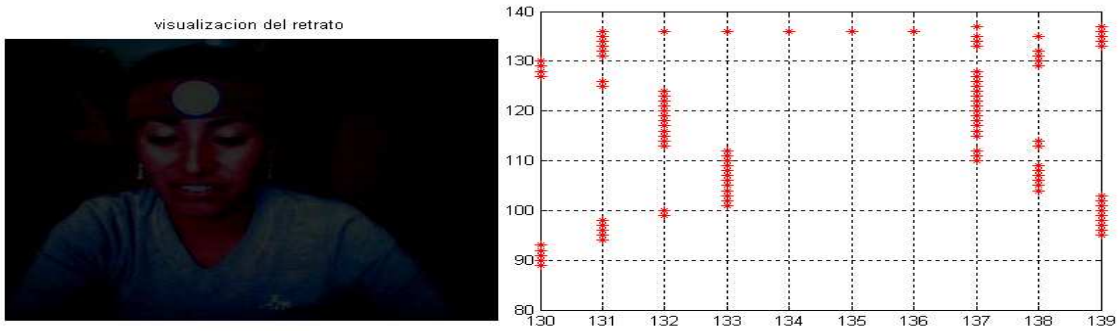
Resultado 17



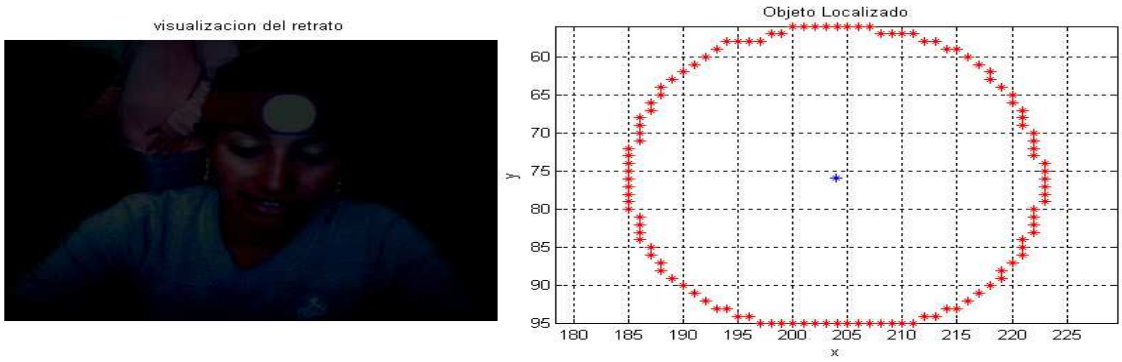
Resultado 18



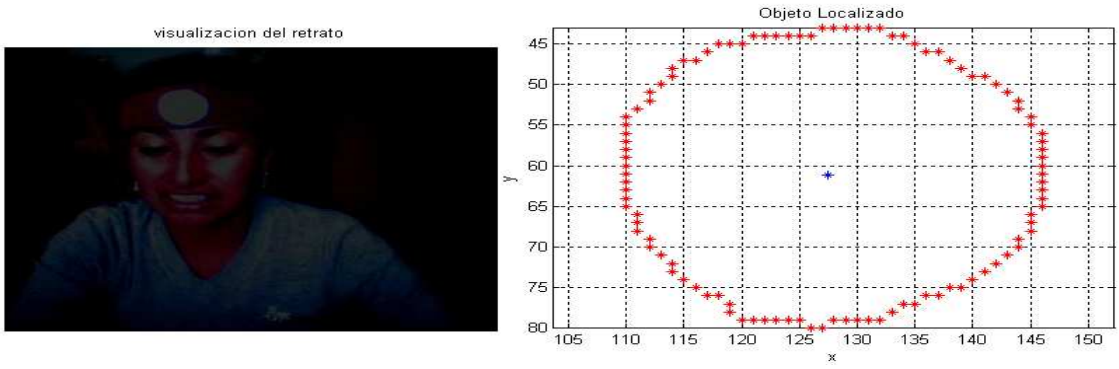
Resultado 19



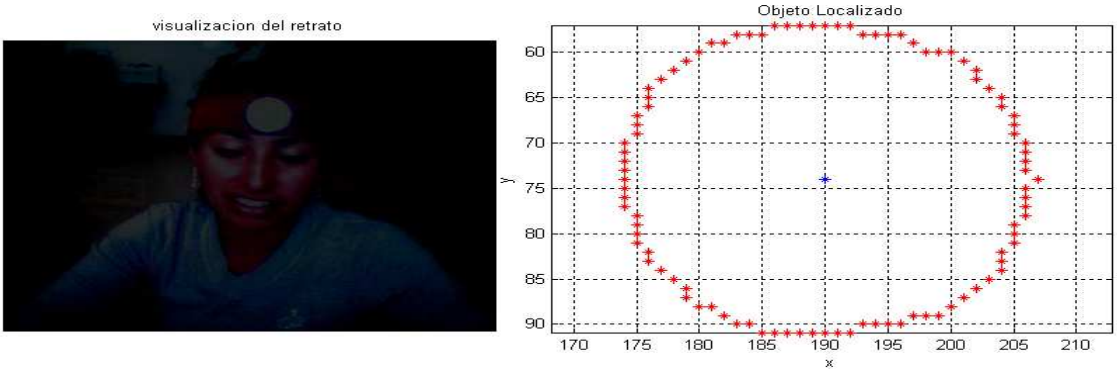
Resultado 20



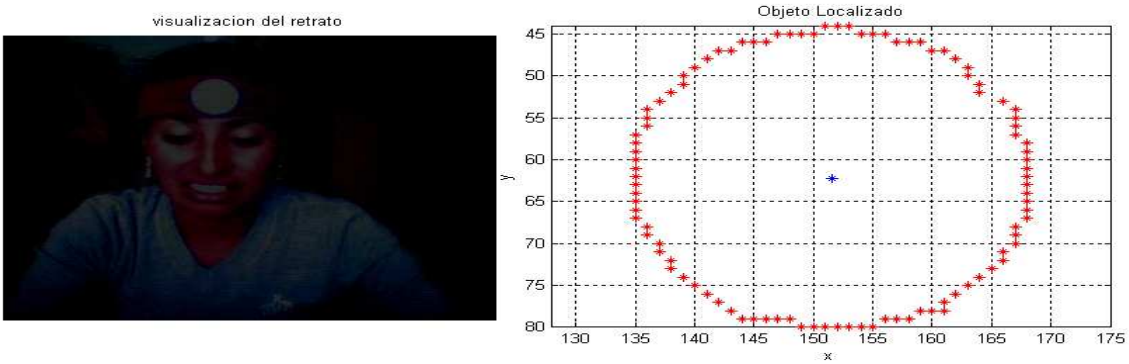
Resultado 21



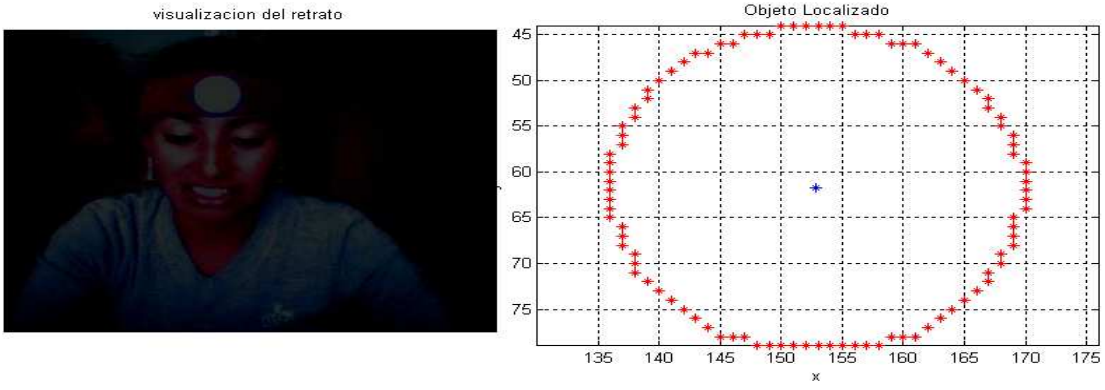
Resultado 22



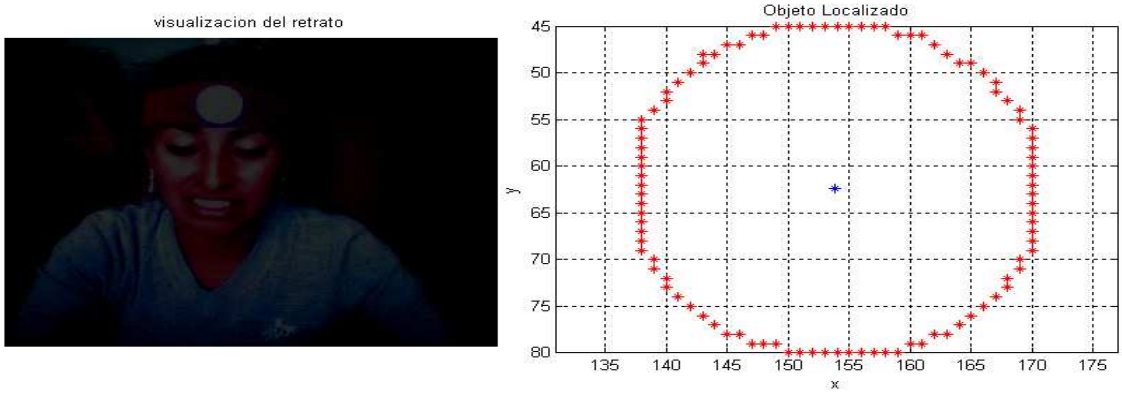
Resultado 23



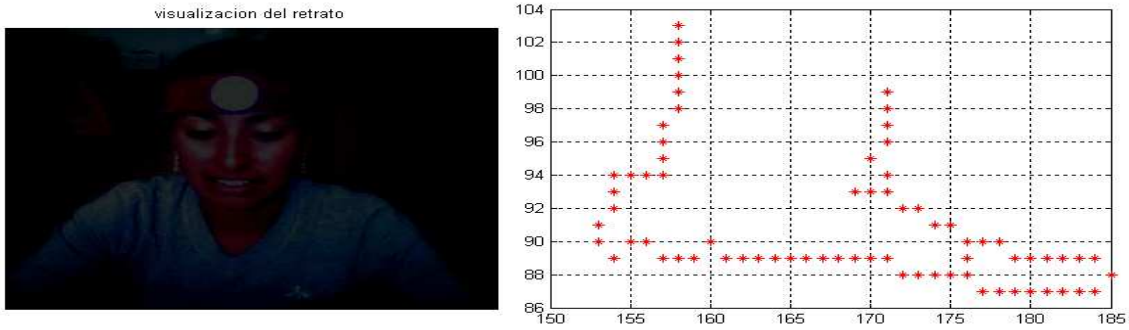
Resultado 24



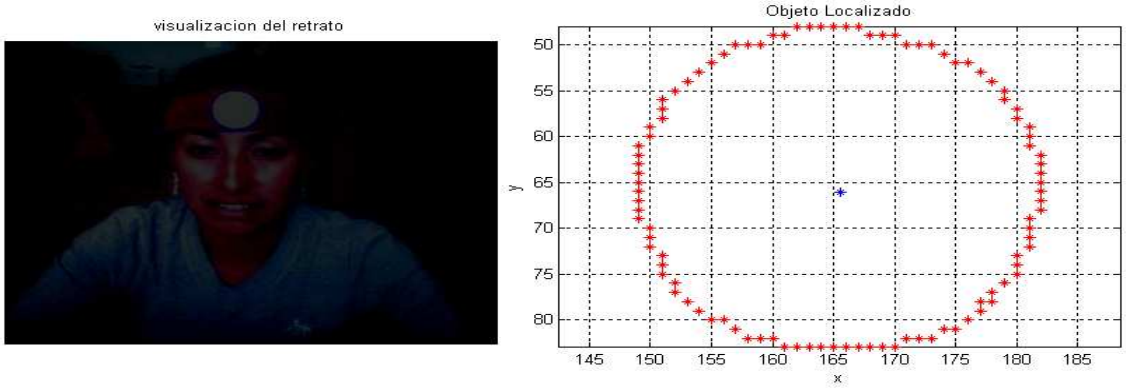
Resultado 25



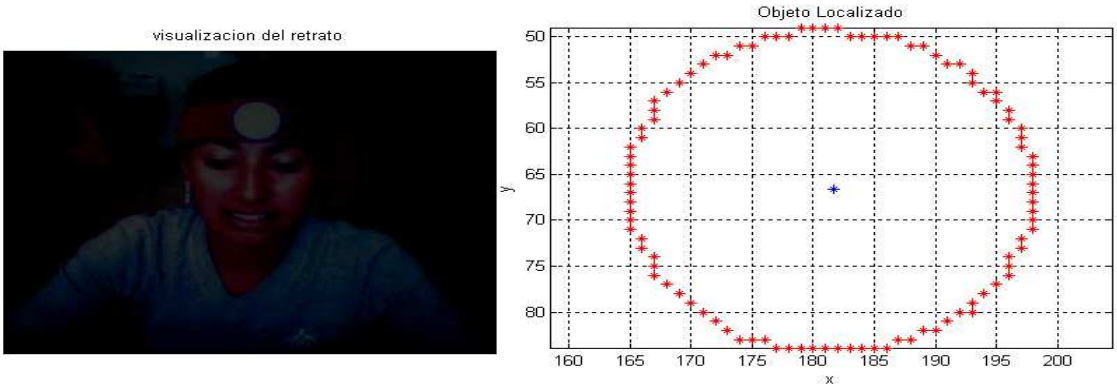
Resultado 26



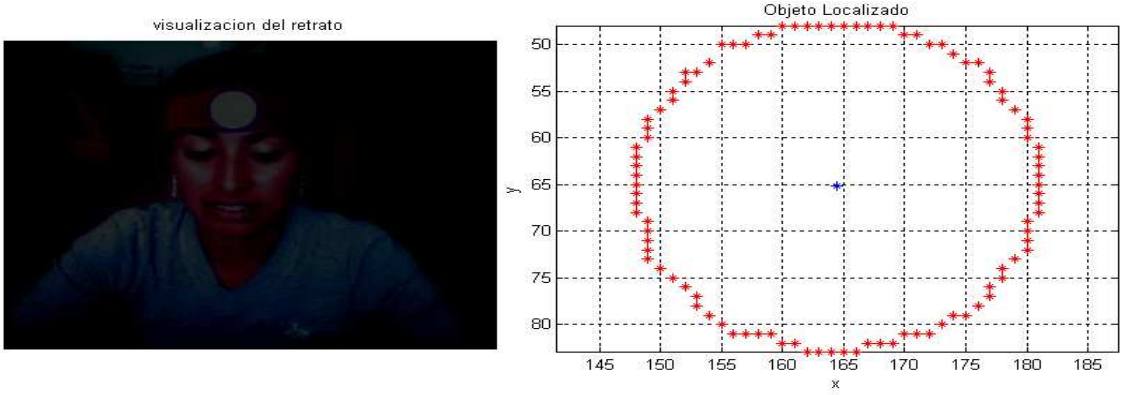
Resultado 27



Resultado 28



Resultado 29



Resultado 30

Figura 4. 5 Imágenes a 100 luxes

El número de fallas a 100 luxes de las 30 imágenes de prueba son de 11 y ahora se procede a realizar el cálculo del error relativo.

$$Er\% = \frac{V_r - V_m}{V_r} * 100$$

$$Er\% = \frac{30 - 19}{20} * 100$$

$$Er\% = 36.667\,\%$$

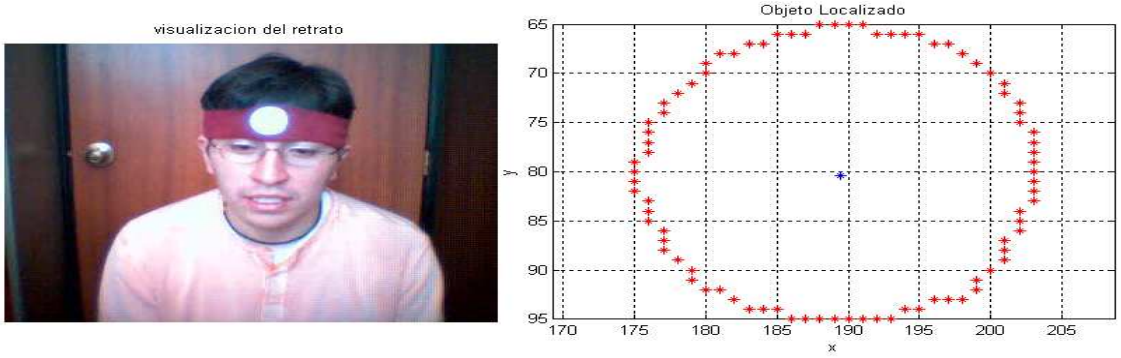
Como ya habíamos predispuesto el error relativo porcentual aumenta conforme disminuye el nivel de iluminación. Y en este caso de 100 luxes que es un valor menor de la mitad de la iluminación de una lámpara de oficina, es muy perceptible que este error vaya en aumento conforme a la disminución de la iluminación.

FUNCIONAMIENTO DEL ALGORITMO SEGÚN EL USUARIO

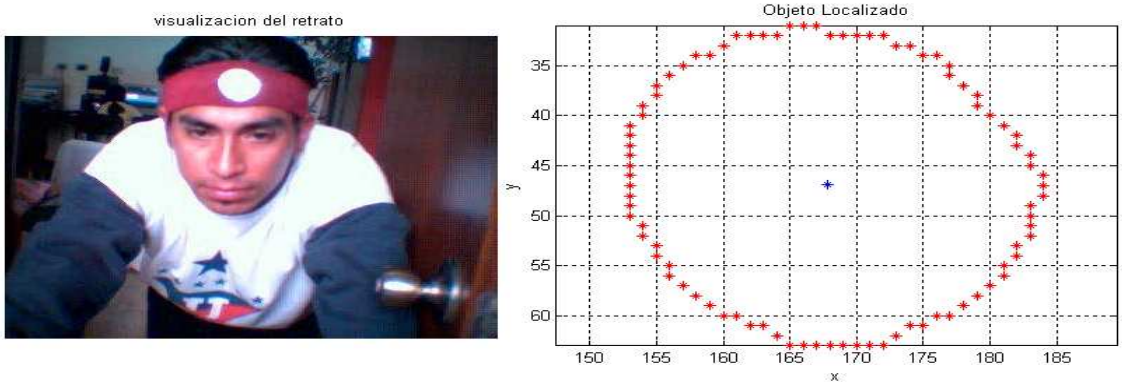
Para realizar este tipo de prueba se ha invitado a un número de personas externas a este proyecto para ver su óptimo desarrollo tanto en tiempo diferido como en tiempo real.

4.3.1.1Pruebas en tiempo diferido

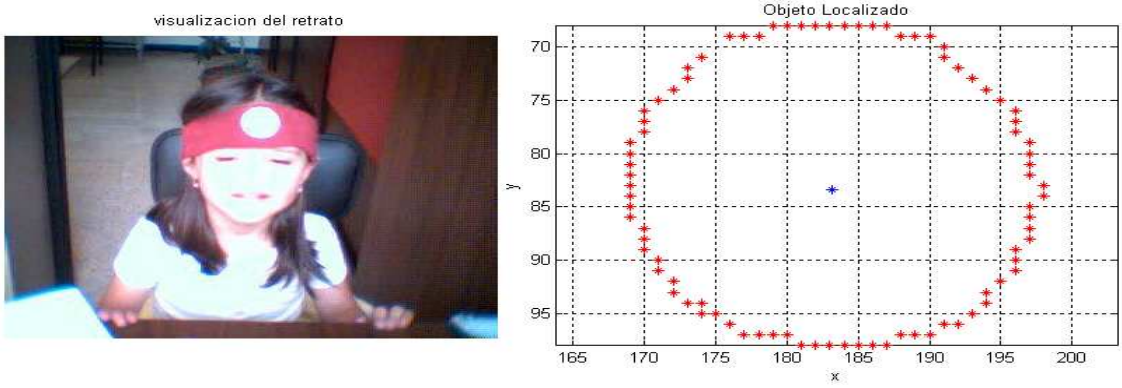
Las pruebas en tiempo diferido se realizaron sobre las mejores condiciones de iluminación. A continuación se mostrarán las diferentes figuras tomadas a los distintos usuarios.



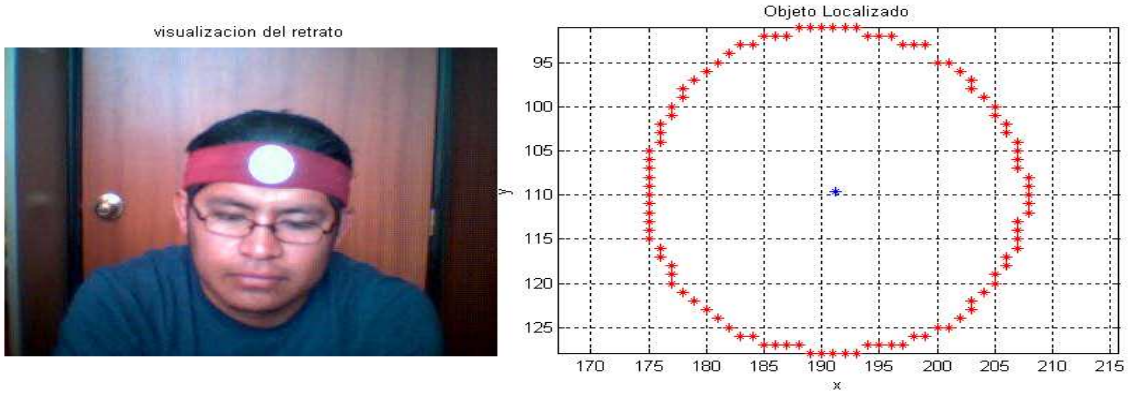
Resultado 1



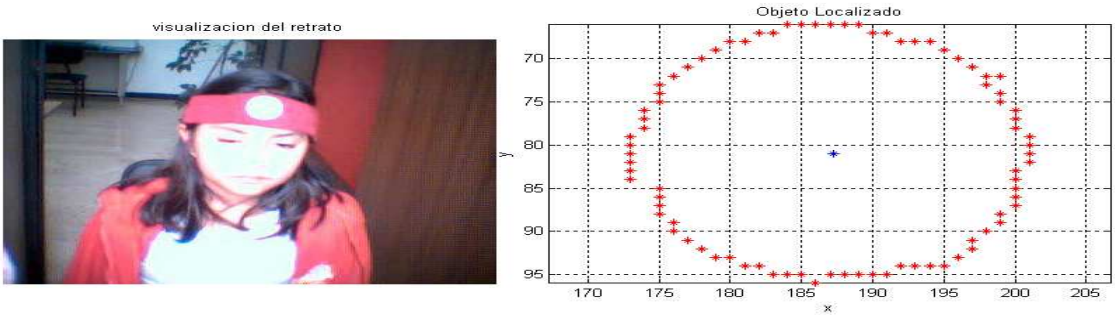
Resultado 2



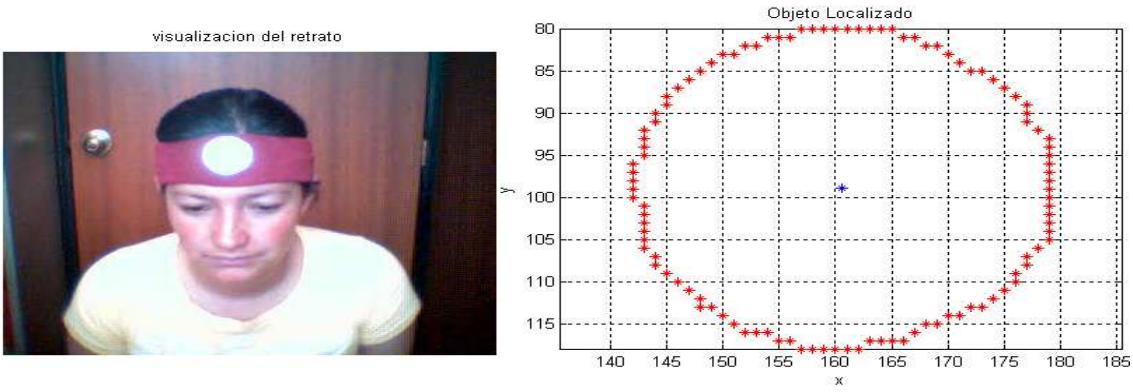
Resultado 3



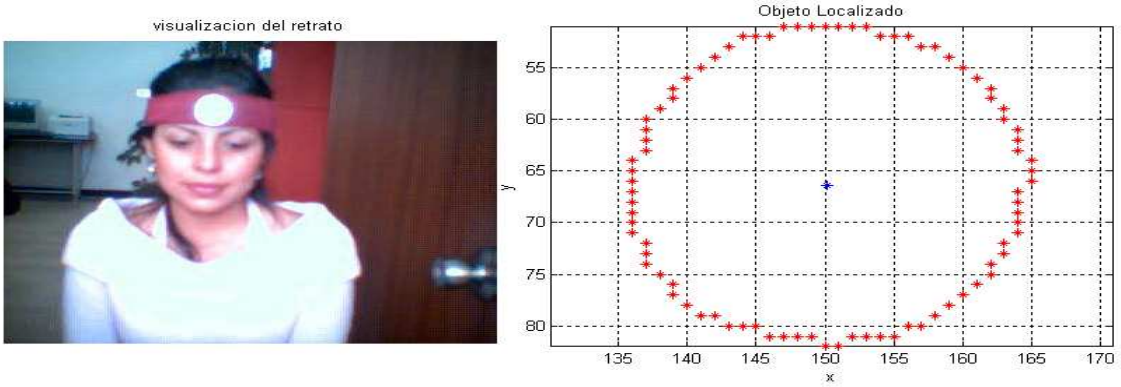
Resultado 5



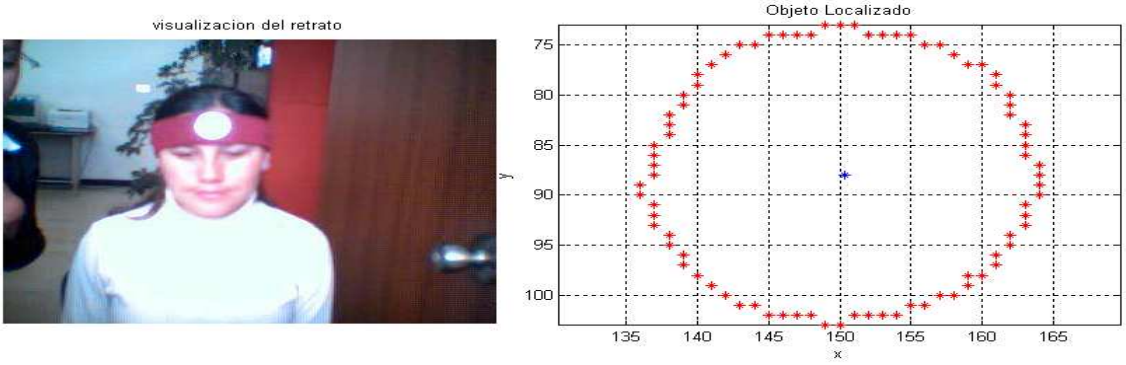
Resultado 6



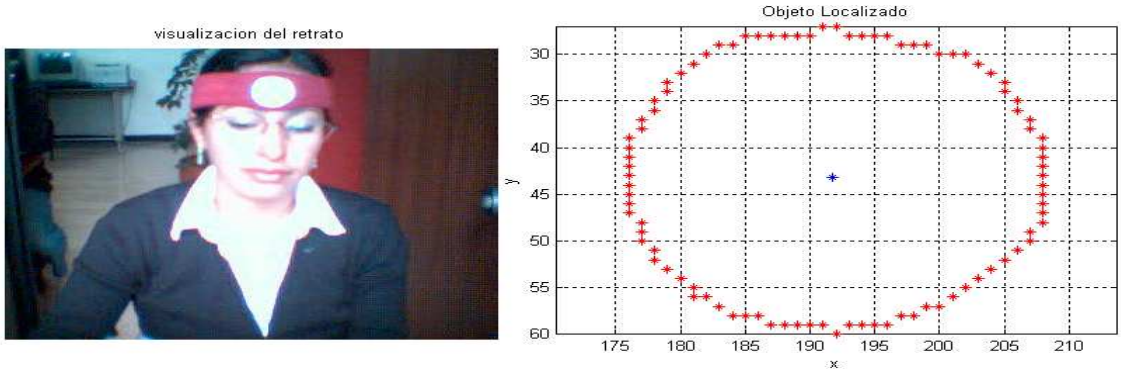
Resultado 7



Resultado 8



Resultado 9



Resultado 10

Figura 4. 6 Resultados en tiempo diferido.

Se ve que con buenas condiciones de luz se pueden obtener buenos resultados.

4.4 Algoritmo de detección de objeto reflector en tiempo real.

Para este algoritmo se utiliza herramientas de video que se explica a continuación

Primero necesitamos una herramienta que permita almacenar las imágenes que se obtienen por medio de la cámara web, para esto se utiliza la función de ***videoinput*** la cual contendrá las características de la cámara web a utilizar.

```
vid = videoinput('winvideo',1,'RGB24_352x288')
```

Ya configurada la variable **vid**, si se desea se puede utilizar la herramienta **preview** que muestra una vista preliminar del video que está filmando la cámara web que se está utilizando.

```
preview(vid)
```

Ahora se procede a configurar las propiedades de la imagen que se obtienen, esto se realiza con la función **set**, y con el parámetro **TriggerRepeat** se puede configurar para que sea infinito como se muestra a continuación:

```
set(vid,'TriggerRepeat',Inf);
```

O también se puede definir un valor que ayuda a especificar la frecuencia con la que se capturan los frames que ingresan por medio del video de entrada para esto se usa la función **FrameGrabInterval**:

```
vid.FrameGrabInterval = 30;
```

Con la función **start** se obtiene el control de la activación del dispositivo (webcam, adquisición de imágenes), pero no controla registro de datos, y funciona de la siguiente manera.

```
start(vid)
```

Para finalizar el uso de la función **start** o controlar el apagado se utiliza la función **stop** con la cual se termina la ejecución de la cámara web.

```
stop(vid)
```

La función que permite la extracción de datos, el tiempo, y la información del objeto de la adquisición de datos es la función **getdata**, la cual se representa de la siguiente manera.

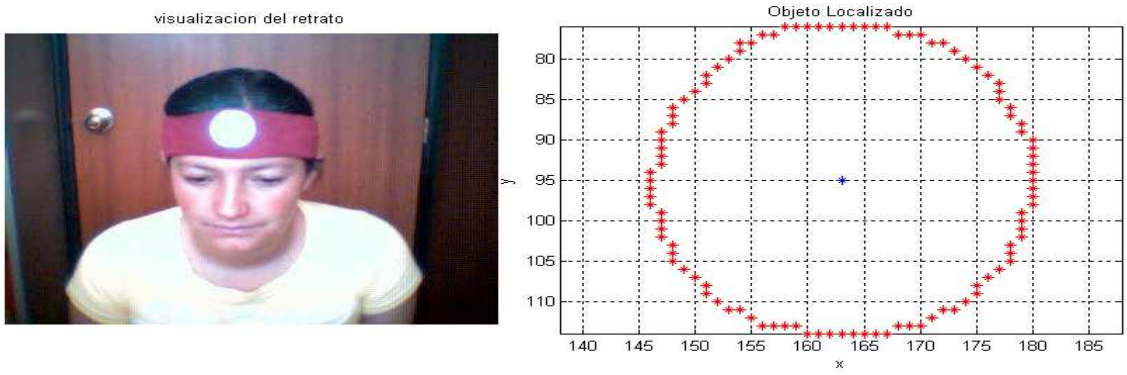
```
data = getdata(vid,1);
```

4.4.1.1 Pruebas en tiempo real

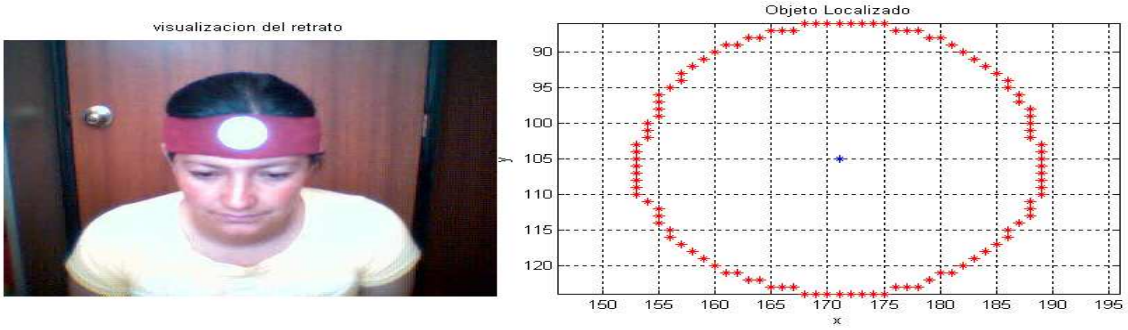
Para las prueba en tiempo real usamos diferentes usuarios, para ver el desarrollo del proyecto y como cada uno lo puede manejar.

4.4.1.2 Usuario 1

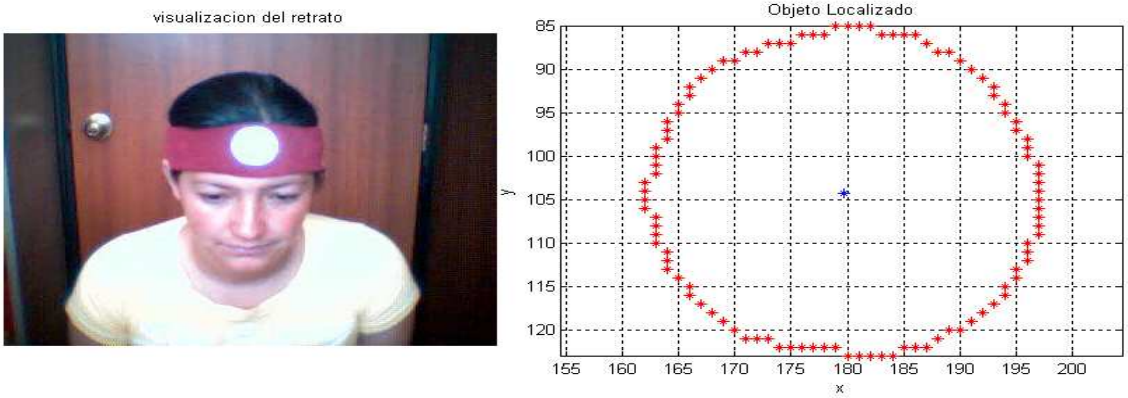
A continuación se muestra el desarrollo de las imágenes captadas en vivo del primer usuario (tiempo real). Dichas imágenes se mostrarán en la figura 4.7. También se ha de realizar el cálculo del error relativo porcentual en cada una de las pruebas.



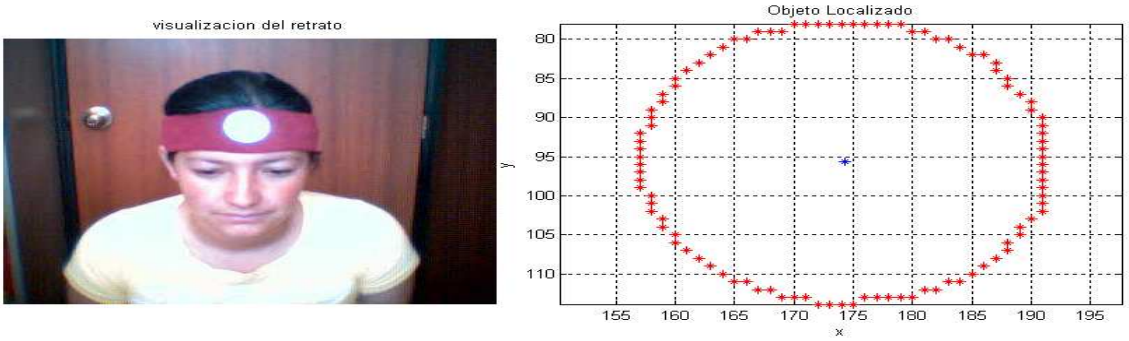
Resultado 1



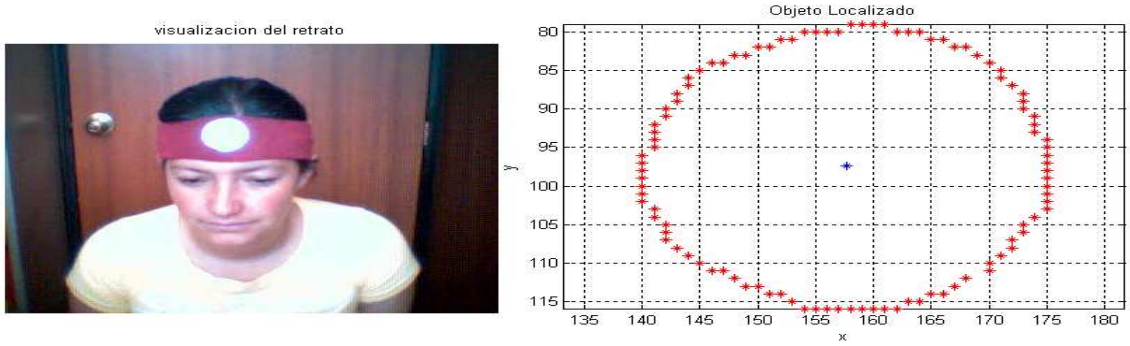
Resultado 2



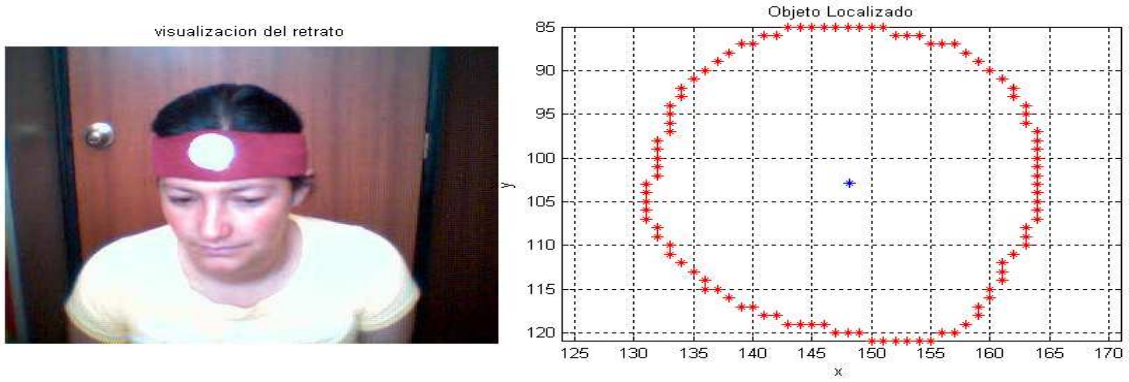
Resultado 3



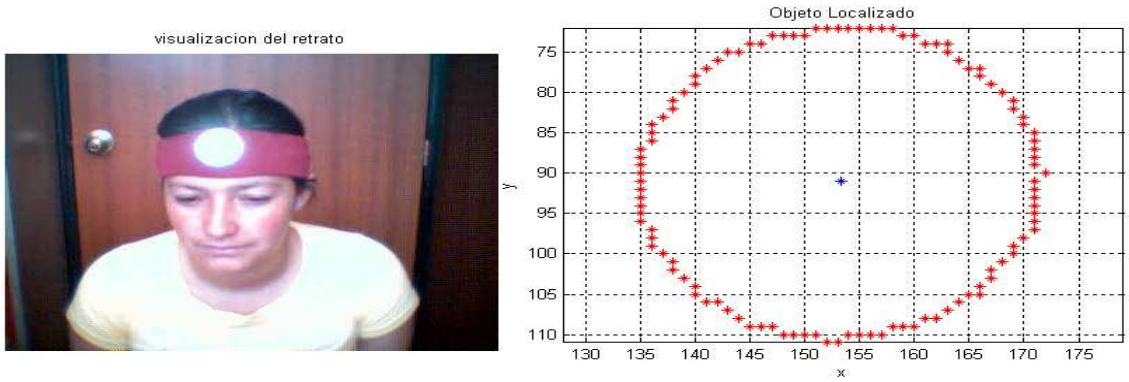
Resultado 4



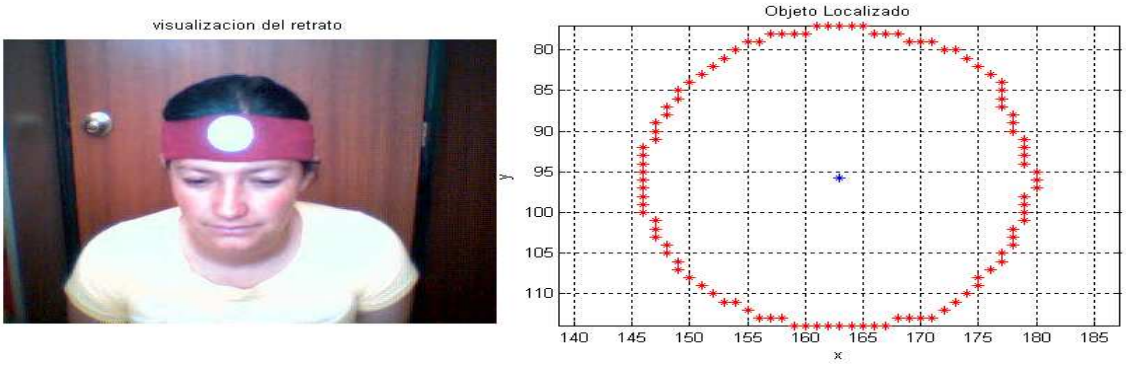
Resultado 5



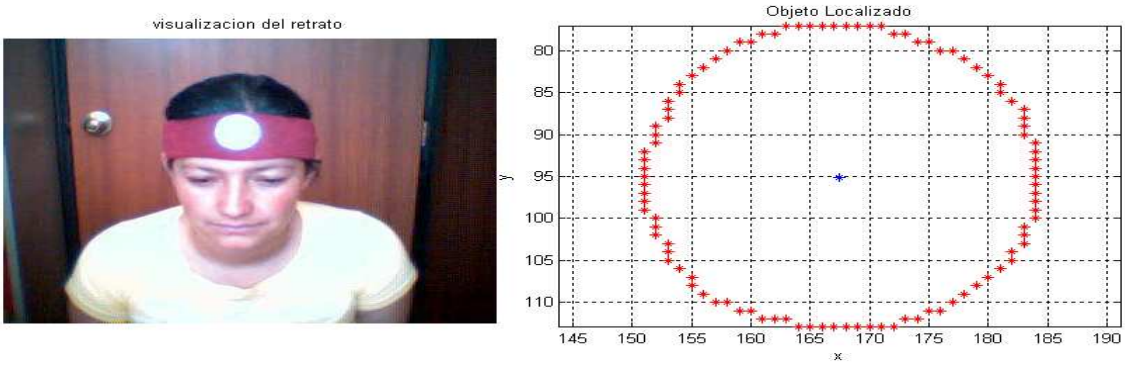
Resultado 6



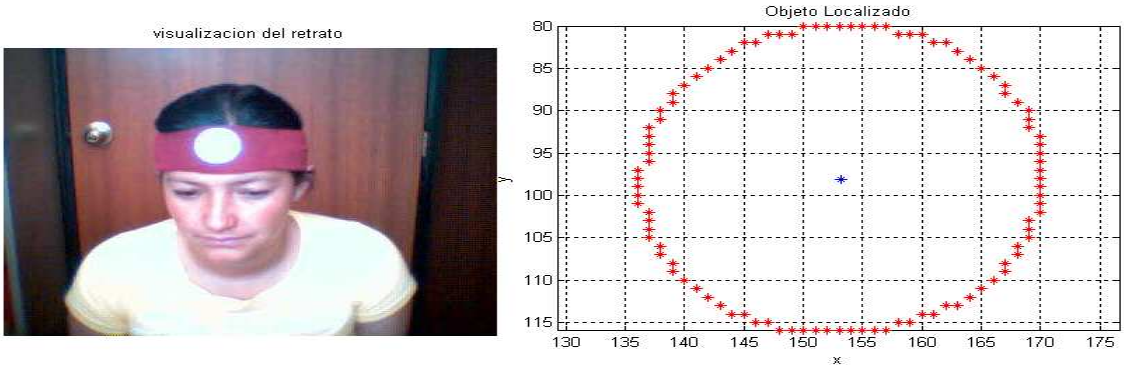
Resultado 7



Resultado 8



Resultado 9



Resultado 10

Figura 4. 7 Imagen en tiempo real, primer usuario

Como se puede ver en las pruebas el número de fallas han sido cero, entonces ahora se procede a calcular el error relativo porcentual.

$$Er\% = \frac{V_r - V_m}{V_r} * 100$$

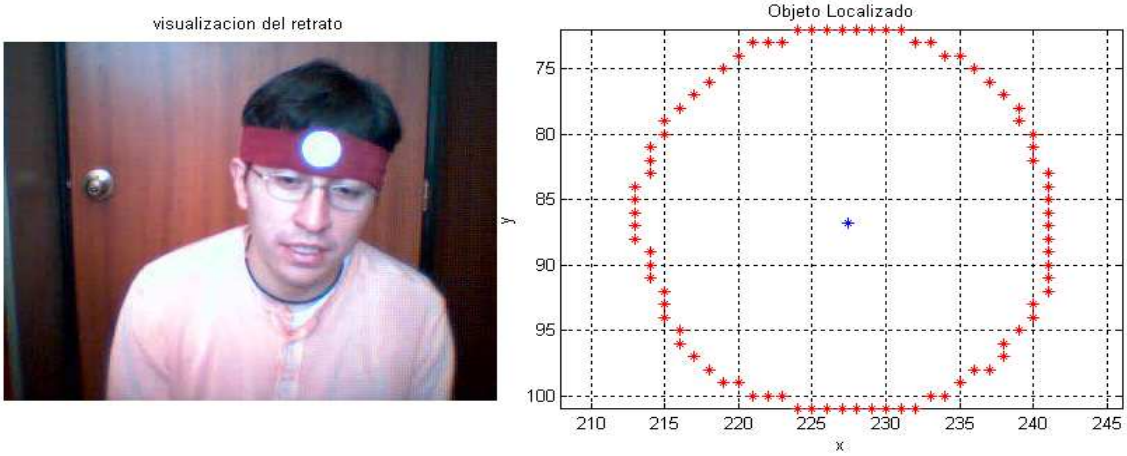
$$Er\% = \frac{10 - 10}{10} * 100$$

$$Er\% = 0\,\%$$

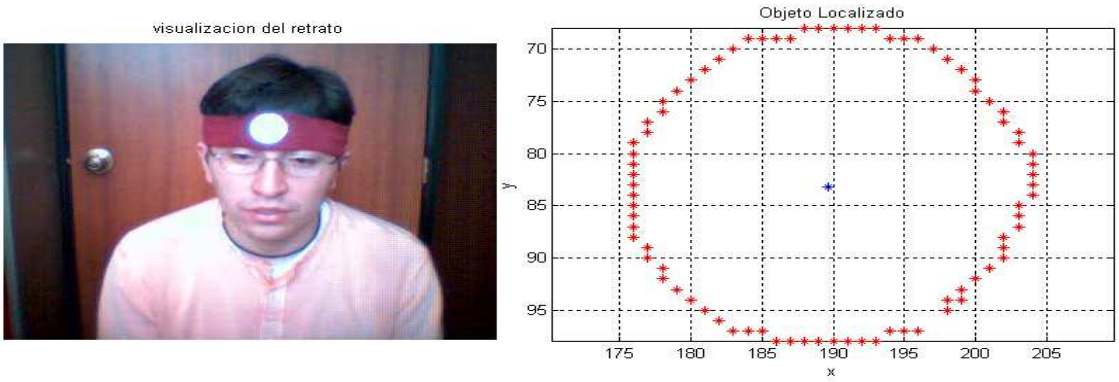
Estas pruebas se realizaron con la luz del día, es decir que se tiene una óptima iluminación, por lo tanto los errores que se esperan son los mínimos, como en este caso el error es nulo.

4.4.1.3 Usuario 2

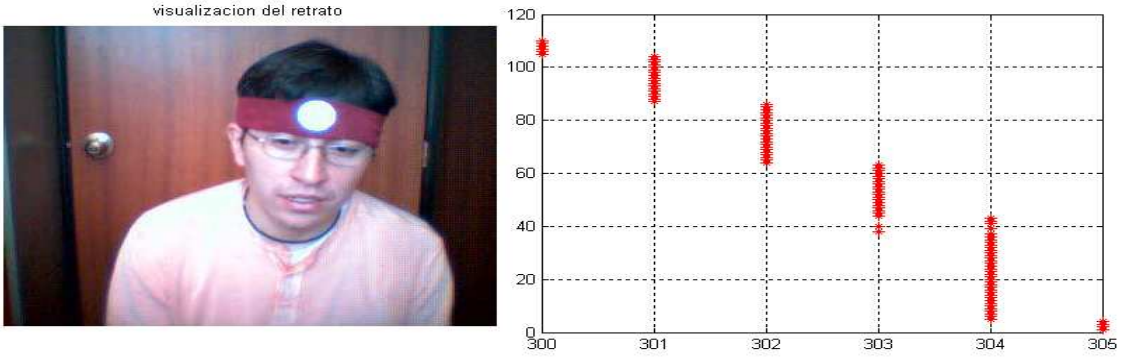
Ahora procedemos con diferente usuario (figura 4.8), explicándole a este como debe utilizar el cintillo con el objeto reflector y cuán importante es mantener la reflexión de este capturada por la cámara web.



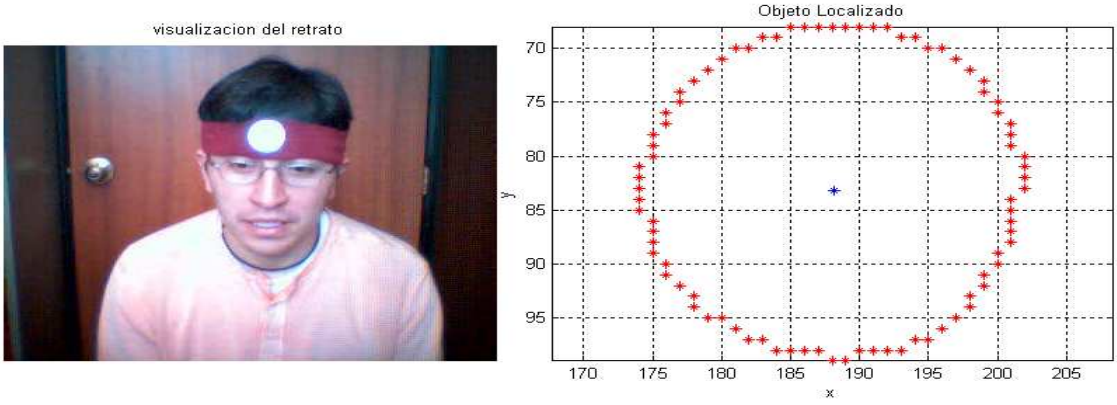
Resultado 1



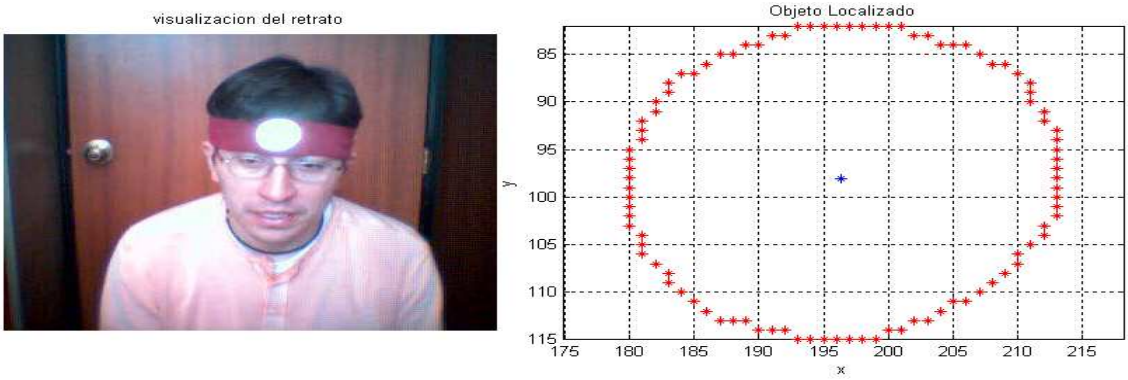
Resultado 2



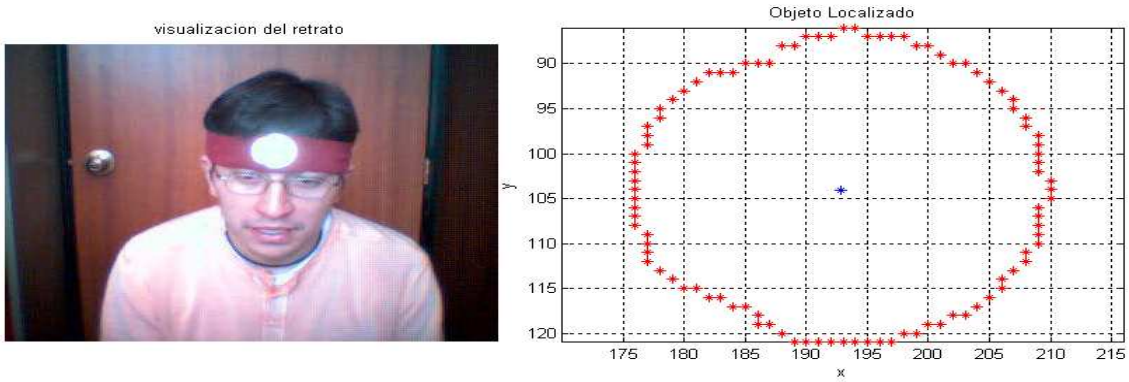
Resultado 3



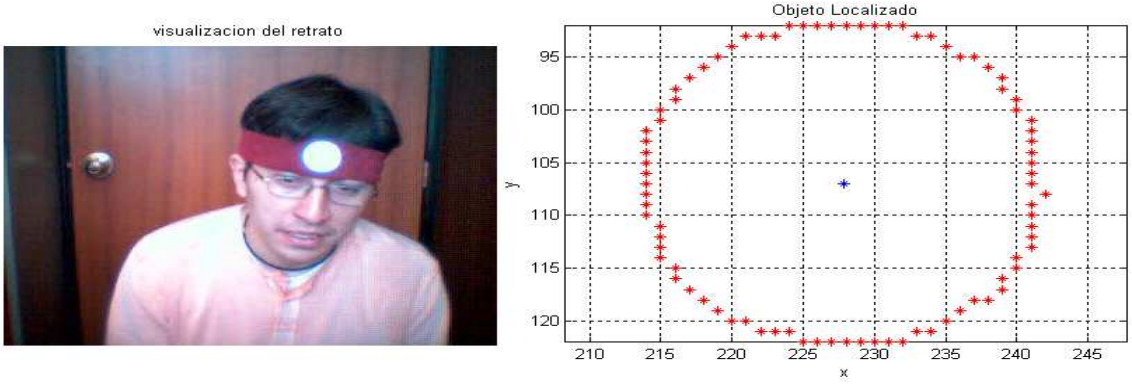
Resultado 4



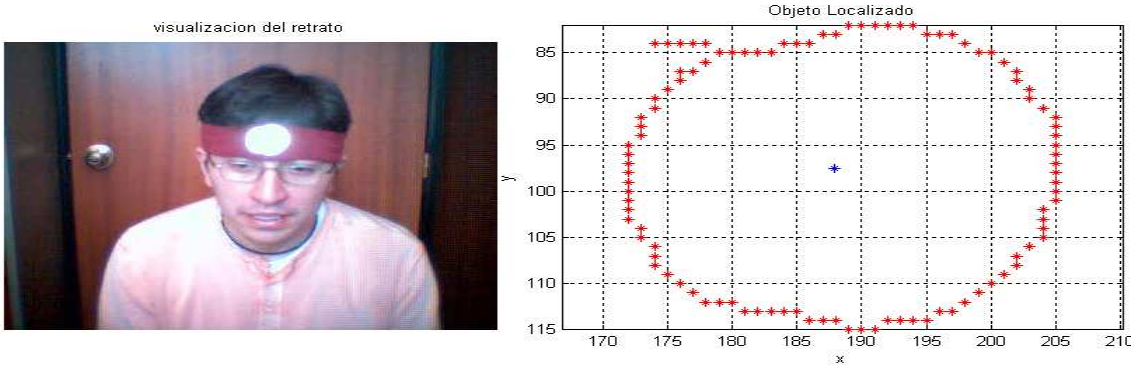
Resultado 5



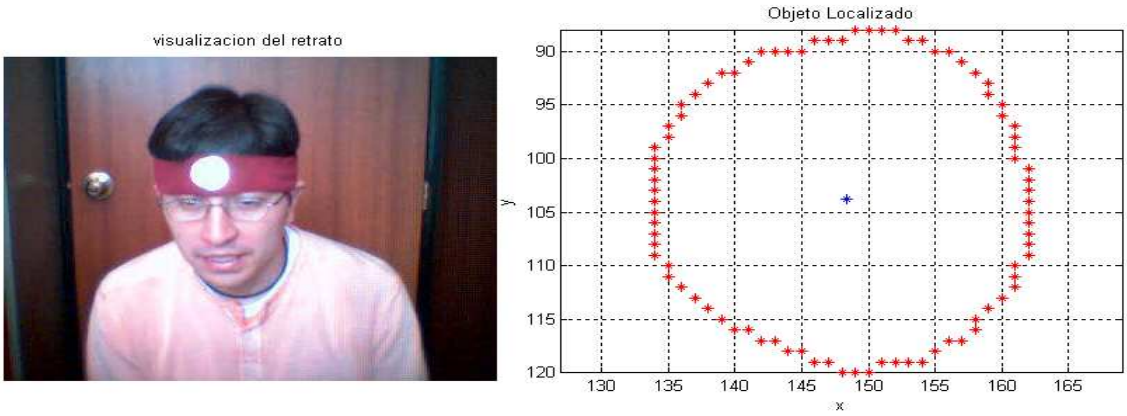
Resultado 6



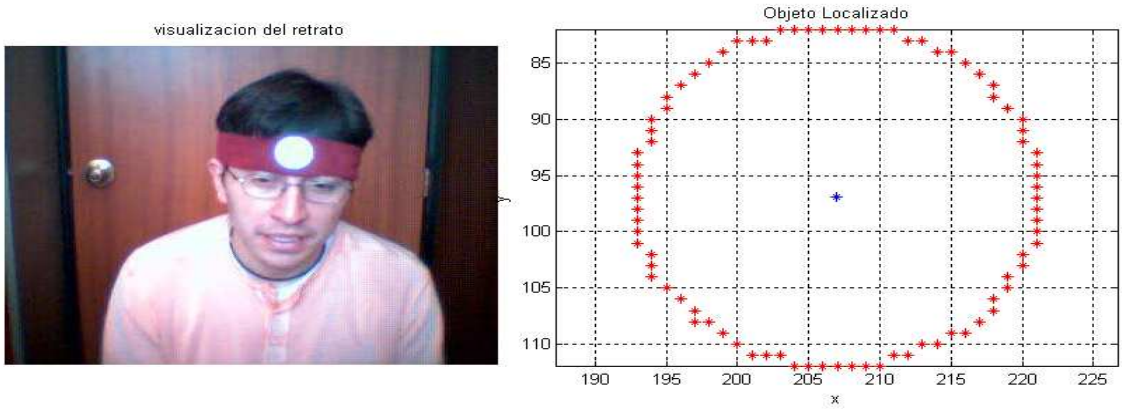
Resultado 7



Resultado 8



Resultado 9



Resultado 10

Figura 4. 8 Imagen en tiempo real, segundo usuario.

El número de fallas que se han obtenido es una, y el error relativo se calcula a continuación.

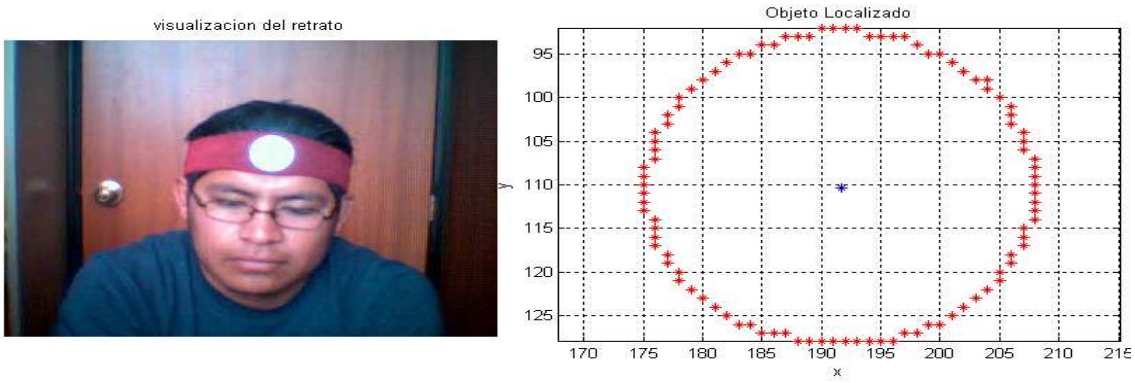
$$Er\% = \frac{V_r - V_m}{V_r} * 100$$

$$Er\% = \frac{10 - 09}{10} * 100$$

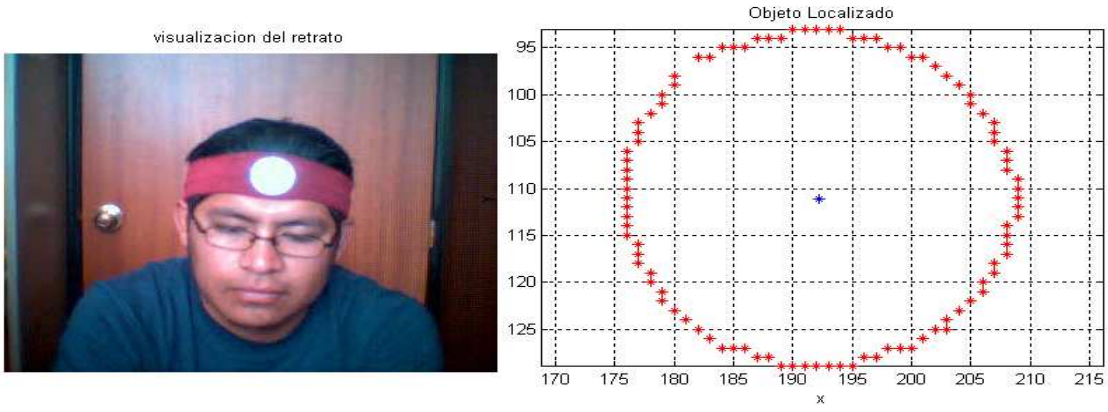
$$Er\% = 10\%$$

4.4.1.4 Usuario 3

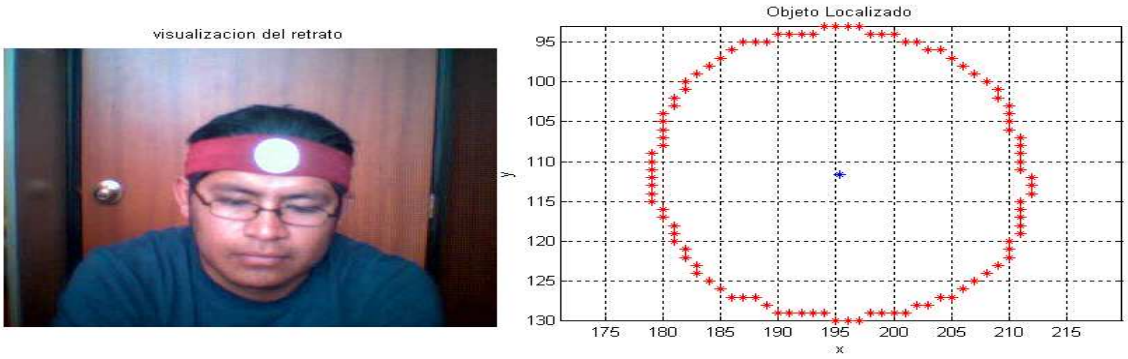
Con el último usuario (figura 4.9) se procede a realizar las pruebas en tiempo real, buscando optimismo en este proyecto y desenvolvimiento para diferentes usuarios.



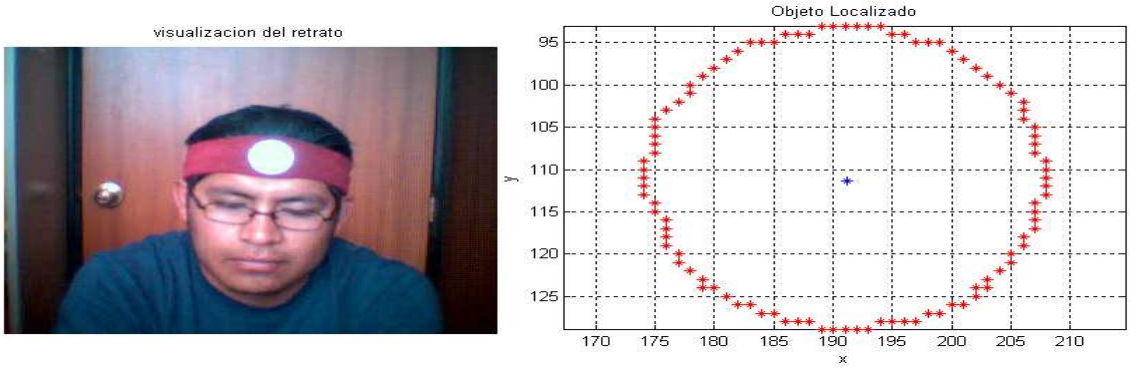
Resultado 1



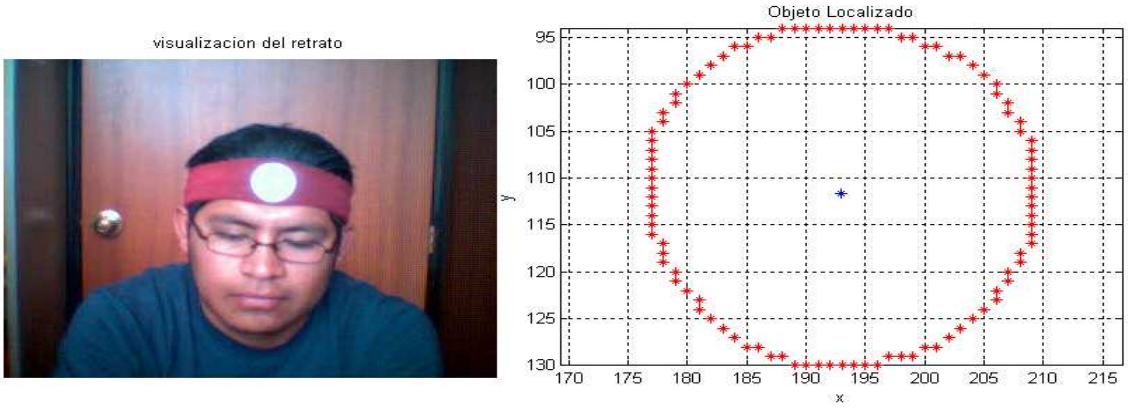
Resultado 2



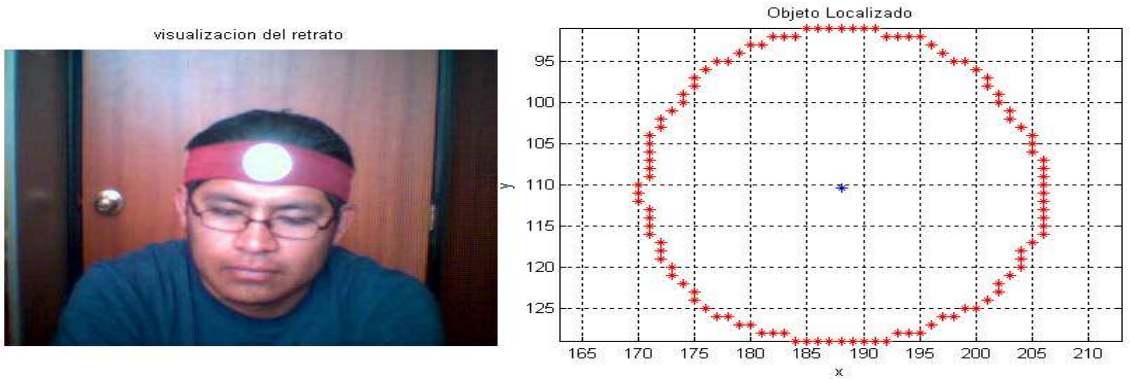
Resultado 3



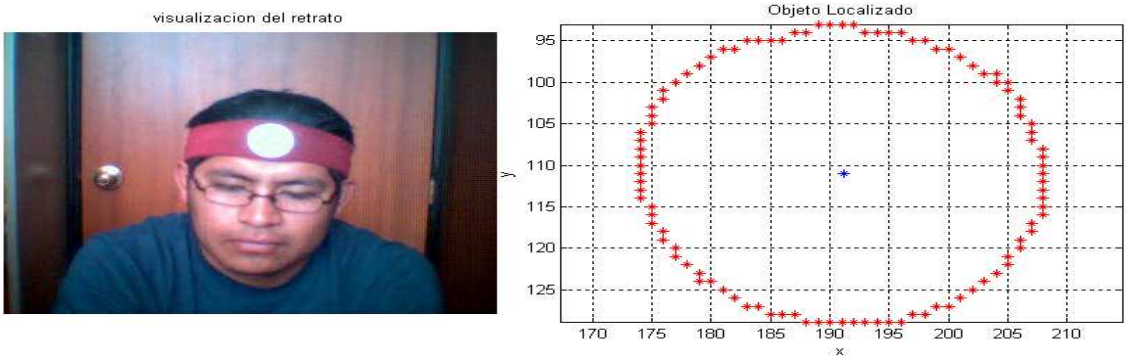
Resultado 4



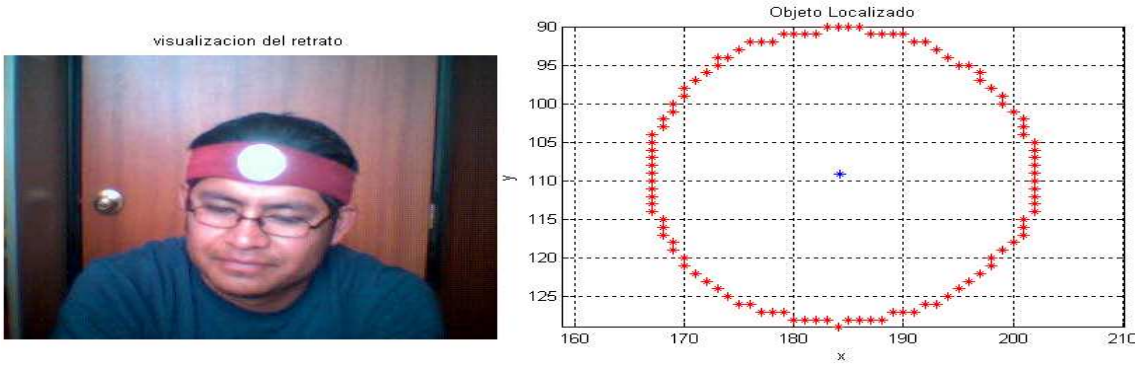
Resultado 5



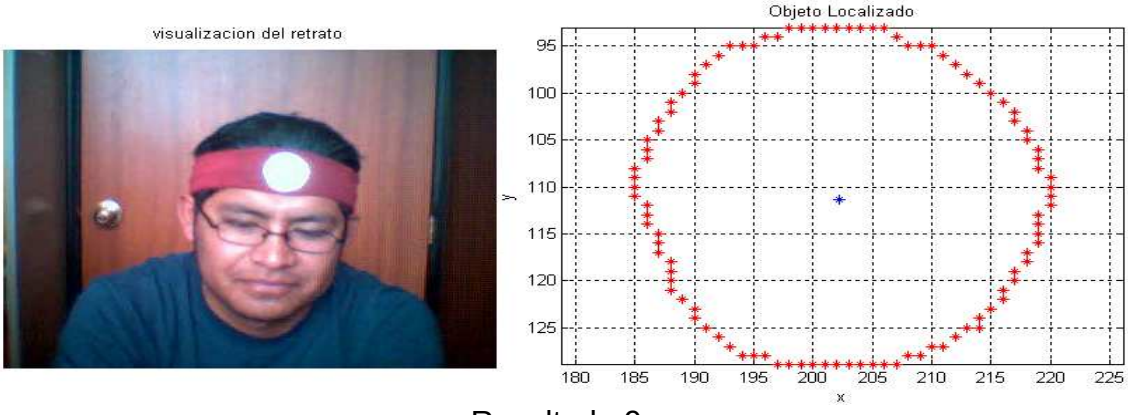
Resultado 6



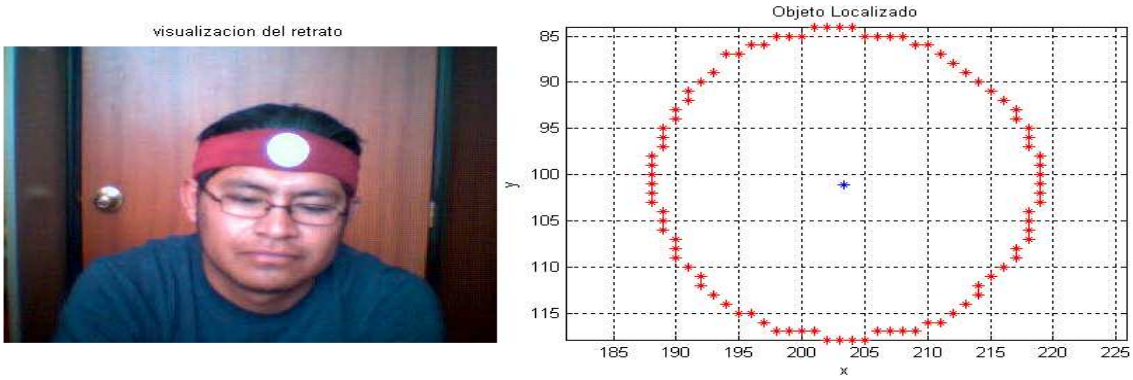
Resultado 7



Resultado 8



Resultado 9



Resultado 10

Figura 4. 9 Imagen en tiempo real, tercer usuario.

El número de fallas que se han detectado es cero y el cálculo relativo se calcula a continuación.

$$Er\% = \frac{V_r - V_m}{V_r} * 100$$

$$Er\% = \frac{10 - 10}{10} * 100$$

$$Er\% = 0\%$$

4.5 Discusión de los resultados

Al igual que proyectos anteriores, se ha buscado la mejor alternativa para que el proyecto sea más accesible para personas con discapacidad motriz y que puedan hacer uso de este tipo de tecnología, viendo la factibilidad de uso y alcance para el usuario.

El proyecto actual se está implementando por primera vez en nuestro país lo que es una gran iniciativa para que las personas con discapacidad motriz puedan hacer uso de este tipo de mouse.

Haciendo un breve análisis de los problemas de iluminación que se dan en otros proyectos similares, como el de detección facial, se determinó que el uso del objeto reflector servirá para minimizar este inconveniente.

Se probaron varios métodos para eliminar objetos no deseados (ruido), para esto se intentó con Filtros en 2D prediseñados, con métodos basados en pixeles (umbralización) pero ninguno dio los resultados esperados**.** Entonces la mejor solución que se encontró fue la eliminación de pixeles “suaves” por medio de la herramienta que el mismo toolbox de Matlab posee, y finalmente se eliminaron los objetos no deseados.

Ahora en base a las características del objeto se extrae el valor del centroide y se lo hace coincidir con el puntero del mouse, mediante librerías de Java y Robot se ejecuta el control del puntero del mouse en Matlab, logrando así el control del

desplazamiento de este en la pantalla, con ayuda del movimiento de la cabeza, pero siempre tratando de que el haz de luz incida en el objeto reflector y la cámara web detecte el reflejo.

Analizando aspectos económicos de otros proyectos se ha considerado que nuestro proyecto es más económico debido a que simplemente se usa una cámara web con requerimientos básicos y un objeto reflector como un espejo redondo de radio pequeño y fácil de encontrar en el mercado, haciendo así que el requerimiento presupuestario sea mínimo en comparación a otros proyectos.

4.5.1 Método de detección de clic

Como ya se había descrito, para que se muestre el cuadro de selección, el usuario deberá poseer una cierta posición por 10 segundos para que el cuadro de selección se muestre.

Para el uso de este cuadro fue necesaria la utilización de librerías y herramientas java y matlab como se describe a continuación.

Las librerías que se muestran a continuación permiten realizar la interacción de las librerías internas del puntero del mouse en un ordenador.

```
import java.awt.Robot;
import java.awt.event.*;
mouse = Robot;
```

Una vez que se ha obtenido el valor del centroide se lo convierte en valor entero para facilitar su operación, ahora para tener almacenado el seguimiento del movimiento del puntero del mouse se lo guarda en una hoja txt, de la siguiente manera.

```
dlmwrite('dato_vector_tiempo.txt', vector1, '-append')
```

Tomo este valor y se ejecuta la herramienta **mouse.mouseMove**, la cual ayudará a coincidir el puntero del mouse donde la posición del centroide lo indique.

```
mouse.mouseMove(xc, yc);
```

Una vez que el puntero se ha ubicado, si el usuario se ha mantenido el tiempo programado, entonces se procede a mostrar el cuadro (figura 4.12), de la siguiente manera.

```
%ayuda a ver la localizacion del mouse
mouse_loc = java.awt.MouseInfo.getPointerInfo.getLocation
pos_cuadro = [xc yc]
pause (0.001)
tiempo = etime(clock,t0);
tiempo1=int32(100*(tiempo))
dlmwrite('dato_vector_tiempo.txt', tiempo1, '-append',...
             'newline', 'pc');

%muestra el cuadro de click derecho y click izquierdo
            if tiempo1>=10
            cuadro=figure('Position',[x 14*y 260 70],...
                'Menu','None',
               'Name','HeadMouse','NumberTitle','off');
                pos=get(cuadro,'Position');

                                izq=uicontrol('ButtonDownFcn',
               'pushbutton',...
                'String', 'Izquierdo',...
                'Position', [10 10 100 50],...
                'BackgroundColor',[.2 .8 0],...
                'FontWeight','bold',...
                'TooltipString','Izquierdo');
                  disp('aplasto el izquierdo')
```

```
                    der=uicontrol('ButtonDownFcn',
'pushbutton',...
 'String', 'Derecho',...
 'Position', [150 10 100 50],...
 'BackgroundColor',[.2 .8 0],...
 'FontWeight','bold',...
 'TooltipString','Derecho');
     disp('aplasto el derecho')
         pause(3)
 delete (cuadro)
```

Figura 4. 10 Cuadro de detección de clic

Para concluir con esta explicación se muestra el algoritmo en tiempo real.

```
clear all
close all
clc

import java.awt.Robot;
import java.awt.event.*;
mouse = Robot;
%creacion del video, vid= video de entrada
vid = videoinput('winvideo',1,'RGB24_352x288');

%vista preliminar del video de entrada
preview(vid)

set(vid,'TriggerRepeat',Inf);
```

```
vid.FrameGrabInterval = 30;

start(vid)

while(vid.FramesAcquired<=30)
data = getdata(vid,1);
     I=data;

%---------------------------------------
%Obtencion de una imagen

          I2=rgb2gray(I);
          I3 = im2double(I2); % CAMBIO A DOUBLE

          %Detección de bordes:
          BW1=edge(I3,'prewitt');

          %ELIMINAR DETALLES PEQUEÑOS EN LAS IMAGENES

           BW1 = bwareaopen(BW1,80);
          [M,NUM]=bwlabel(BW1,8);

           % Operaciones morfológicas:
           se = strel('disk',1);
           BW11 = imclose(BW1,se);
           BW12 = imfill(BW11,'holes');

%ubicacion de los valores uno en la Matriz M
[F,C] = find (M==NUM);

%PROPIEDADES DE LA IMAGEN
stats = regionprops(M,'all'); %todas las propiedades
```

```
for k=1:NUM
 a = stats(k).Area;
p=stats(k).Perimeter;
end

%DETERMINACION DEL OBJETO REDONDEADO
for  k=1:1:NUM
         [F,C] = find (M==k);
        for i=1:length(C)
            puntf=F(i);
            puntc=C(i);
        end
            %LONGITUDES DE LAS FILAS
            z=1;
        for j=min(C):max(C);
            numeroc=find(C==j);
            longitudc(z)=length(numeroc);
            z=z+1;
        end
            longitudc;

         %LONGITUDES DE LAS COLUMNAS
            y=1;
        for m=min(F):max(F);
            numerof=find(F==m);
            longitudf(y)=length(numerof);
            y=y+1;
        end
             longitudf;

              % CENTROIDE
            c=stats(k).Centroid;
            x = c(:, 1);  %centroide componente x
```

```
y = c(:, 2);    %centroide componente y

%RECONOCIMIENTO DEL MOUSE
x1=int32(x);  %numero entero de x
y1=int32(y);  %numero entero de y

%obtencion del numero decimal
x2=1000*x;
x3=1000*x1;
x4=abs(x3-x2);
x5=0;
if x4<=499 && x4>=450;
    x5=x1+1;
else x4<450 && x4>499;
    x5=x1;
end

y2=1000*y;
y3=1000*y1;
y4=abs(y3-y2);
y5=0;
if y4<=499 && y4>=450;
   y5=y1+1;
else y4<450 && y4>499;
    y5=y1;
end
x5;
y5;

 respf=int32(max(F)+min(F))/2;
 respc=int32((max(C)+min(C))/2);

 distcmax=abs(x5-max(C));
```

```
 distcmax1=int32(distcmax);
 distcmin=abs(x5-min(C));
 distcmin1=int32(distcmin);
 distfmax=abs(y5-max(F));
 distfmax1=int32(distfmax);
 distfmin=abs(y5-min(F));
 distfmin1=int32(distfmin);

 %obtención del numero decimal para distsmax
distcmax2=1000*distcmax;
distcmax3=1000*distcmax1;
distcmax4=abs(distcmax3-distcmax2);
distcmax5=0;
if distcmax4<=499 && distcmax4>=450; %467
    distcmax5=distcmax1+1;
else distcmax4<450 && distcmax4>499;
    distcmax5=distcmax1;
end

 %obtención del numero decimal para distcmin
distcmin2=1000*distcmin;
distcmin3=1000*distcmin1;
distcmin4=abs(distcmin3-distcmin2);
distcmin5=0;
if distcmin4<=499 && distcmin4>=450;
  distcmin5=distcmin1+1;
else distcmin4<450 && distcmin4>499;
    distcmin5=distcmin1;
end

%obtención del numero decimal para distrmax
 distfmax2=1000*distfmax;
distfmax3=1000*distfmax1;
```

```
distfmax4=abs(distfmax3-distfmax2);
distfmax5=0;
if distfmax4<=499 && distfmax4>=450;
    distfmax5=distfmax1+1;
else distfmax4<450 && distfmax4>499;
    distfmax5=distfmax1;
end

%obtencion del numero decimal para distfmin
distfmin2=1000*distfmin;
distfmin3=1000*distfmin1;
distfmin4=abs(distfmin3-distfmin2);
distfmin5=0;
if distfmin4<=499 && distfmin4>=450;
    distfmin5=distfmin1+1;
else distfmin4<450 && distfmin4>499;
    distfmin5=distfmin1;
end

distcmax5;
distcmin5;
distfmax5;
distfmin5;

if ((distfmax5==distfmin5 ||
distfmax5==(distfmin5+1) || (distfmax5-1)==distfmin5 ||
(distfmax5+1)==distfmin5 || distfmax5==(distfmin5-1)) &&
(distfmax5>=12 && distfmin5>=12) && (distcmin5==distcmax5 ||
(distcmax5+1)==distcmin5 || (distcmax5-1)==distcmin5 ||
(distcmax5+1)==distcmin5 || distcmax5==(distcmin5-1)) &&
(distcmax5>=14 && distcmin5>=14) && (x5==respc || x5==(respc-
1)) && (y5==respf || respf==(y5+1) || respf==(y5-1)))
    disp('es circulo')
```

```
            k_circulo=k;
            break
            else
            disp('no es circulo')
            end
    end

    %CENTROIDE

        c=stats(k_circulo).Centroid;
        xc = c(:, 1);
        yc = c(:, 2);

    t0 = clock;
            vector = int32([xc, yc])
            vector1=(vector+1);
            vector1=(vector-1);
            vector1=vector;

          dlmwrite('dato_vector_tiempo.txt', vector1, '-
          append')

            mouse.mouseMove(xc, yc);

%ayuda a ver la localizacion del mouse
 mouse_loc = java.awt.MouseInfo.getPointerInfo.getLocation
            pos_cuadro = [x y]
            pause (0.001)
            tiempo = etime(clock,t0);
            tiempo1=int32(100*(tiempo))
           dlmwrite('dato_vector_tiempo.txt', tiempo1, '-
          append',...
           'newline', 'pc');
```

```
          %muestra el cuadro de click derecho y click
izquierdo
             if tiempo1>=10
             cuadro=figure('Position',[x 14*y 260 70],...
             'Menu','None',
           'Name','HeadMouse','NumberTitle','off');
              pos=get(cuadro,'Position');

             izq=uicontrol('ButtonDownFcn', 'pushbutton',...
                 'String', 'Izquierdo',...
                 'Position', [10 10 100 50],...
                 'BackgroundColor',[.2 .8 0],...
                 'FontWeight','bold',...
                 'TooltipString','Izquierdo');
                     disp('aplasto el izquierdo')

                                  der=uicontrol('ButtonDownFcn',
                'pushbutton',...
                 'String', 'Derecho',...
                 'Position', [150 10 100 50],...
                'BackgroundColor',[.2 .8 0],...
                 'FontWeight','bold',...
                 'TooltipString','Derecho');
                     disp('aplasto el derecho')
                    pause(3)
                 delete (cuadro)
             end
          end

  stop(vid)
closepreview(vid);
```

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones del proyecto y se aporta con un resumen de las contribuciones científicas y tecnológicas de este trabajo y se plantean posibles trabajos futuros que brinda este proyecto.

5.1 CONCLUSIONES

- El programa está constituido por dos etapas que son seguimiento del objeto reflector desarrollado por algoritmos en Matlab y detección del clic efectuado por librerías utilizado para el desarrollo del proyecto.

- Para el presente proyecto no hubo la necesidad de la utilización de cámaras sofisticadas o con alta resolución, aunque esta característica tiene gran influencia en este proyecto.

- Para el procesamiento de las imágenes es necesario realizar operaciones morfológicas de relleno, filtrado y eliminación de ruido, con el propósito de poder optimizar la detección de nuestro objeto base.

- Es necesario en el proceso de detección de imágenes obtener las propiedades de cada elemento con el fin de poder seleccionar el elemento deseado que se encuentran en el mismo plano de la imagen.

- En el reconocimiento del dispositivo el cual se utiliza como mouse, en el primer algoritmo realizado dependía mucho de la cantidad de iluminación, es decir si había mucha iluminación reconocía el objeto mouse y otro elementos con geometría cerrada o no, si había cantidad de luz promedio reconocía adecuadamente, pero si en el recuadro de la imagen había un objeto blanco el algoritmo descrito presentaba errores, y muchos más cuando la cantidad de iluminación bajaba presentaba mayores errores, por lo tanto se hizo varios planteamientos de algoritmos.

- Se debió plantear un nuevo algoritmo que permita el reconocimiento de un objeto redondeado, porque no cumple con las características de

circunferencia perfecta, es decir permitir un radio no constante, esto se notaba aún más en el movimiento de cabeza, entonces se permite un porcentaje de error, y con ello, la posibilidad máxima del reconocimiento del mouse reflector.

- Finalmente se hizo un análisis del puntero del mouse respecto al cuadro de selección de clic derecho y clic izquierdo, para lograr una concordancia de funcionamiento entre el cuadro de selección y el puntero.

- El uso de lentes o color de la piel no influyó en este proyecto, como se podrá haber notados en algunos frames con diferentes resultados, ya que el reconocimiento del objeto reflector es independiente si el usuario usa lentes o no, y el nivel de la iluminación sobre el usuario solo influye en el objeto reflector.

- Hay que reconocer que con la aparición de las computadoras digitales, el mouse (general) pasa a segundo plano ya que se tiene una interacción directa con la pantalla, pero aún este medio es muy costoso.

5.2 APORTACIONES PRINCIPALES

Este proyecto sirve de gran aportación para acercar al discapacitado al uso de la manipulación del puntero del mouse, ya que como se explicó en el capítulo I, las personas que sufren diferentes tipos de discapacidades físicas sufren un aislamiento de la sociedad.

El valor económico para la inversión del uso de este proyecto es aceptable, ya que los dispositivos a utilizar se encuentran accesibles en el mercado.

Este proyecto sobretodo aporta a los jóvenes discapacitados ya que al poder incursionar al uso de diferentes tecnologías, facilita su desarrollo en la sociedad

como entes normales, y así ayudar con la confianza del discapacitado frente a cualquier tipo de tecnología.

5.3 TRABAJOS FUTUROS

En realidad en este campo de tecnología adaptativa o tecnología para personas con diferentes discapacidades físicas, es muy amplio. Como ya se pudo ver en la exposición de diferentes proyectos similares, con sus respectivas ventajas y desventajas.

5.3.1 RECOMENDACIONES

- Este proyecto sería mejorado si la cámara web sería de mejor resolución, aunque esta consideración aumentaría el precio considerablemente. Ya que este tipo de mouse ante los diferentes tipos de luz mostrará menos sensibilidad.

- Es aconsejable revisar las propiedades de la cámara web como es la resolución que esta ofrece y si fuera necesario ajustar las propiedades de brillo, el contraste, los balances de colores, etc.

- Otra alternativa sería utilizar cámaras infrarrojas para que los errores de los resultados en niveles de luz baja u oscuridad sean mínimas.

- Tener más acceso a personas que tengan discapacidades motrices para diferentes tipos de pruebas y así analizar mejores alternativas tanto en software como en hardware.

BIBLIOGRAFÍA

- José Luis Alba -Universidad de VigoJesús Cid -Universidad Carlos III de MadridInmaculada Mora -Universidad Rey Juan Carlos, “Métodos de análisis de imágenes”. Extracción de características, marzo de 2006.

- José Luis Alba -Universidad de Vigo. Jesús Cid -Universidad Carlos III de Madrid. “Reconocimiento de Patrones”, mayo de 2006.

- José Luis Alba, Fernando Martín -Universidad de VigoJesús Cid - Universidad Carlos III de MadridInmaculada Mora -Universidad Rey Juan Carlos. “Morfología matemática”. Aplicación a procesado de imágenes binarias y monocromáticas, abril de 2006.

- Prof. Erick Norman Guevara Corona, “Aplicación de las Matrices en el Procesamiento de Imágenes”.

- Stephen J Chapman. “Matlab. Programing for Engineering”. Segunda edición.

- Doctor Jurado de los Santos, del grupo CIFO miembro del departamento de Pedagogía Aplicada de la Universidad Autónoma de Barcelona (UAB). “La Transición hacia el mundo del trabajo para las personas con discapacidad. El papel de las tecnologías”.

- Agencia europea para el desarrollo de la Educación Especial en el 2002. “Informe “Transición de la escuela al empleo””. www.european-agency.org.

- Documentación del Toolbox de Image Processing de Matlab www.mathworks.com

REFERENCIAS BIBLIOGRAFICAS

[1]http://www.bc.edu/schools/csom/eagleeyes/cameramouse/about.html#q1

[2]http://www.bc.edu/schools/csom/eagleeyes/cameramouse/gallery/page11.html

[3]http://www.canal-ar.com.ar/noticias/noticiamuestra.asp?Id=4496#comentario

[4]http://robotica.udl.cat/headmouse/headmouse1/headmouse1.html#CONTROL

[5]http://www.naturalpoint.com/smartnav/products/about.html

http://tetramouse.com/manual.html#connect

[6]http://www.naturalpoint.com/smartnav/store/accessories.html

[7]http://www.naturalpoint.com/smartnav/products/about.html

[8]http://tetramouse.com/manual.html#connect

ANEXO

CÓDIGO FUENTE

A. PROGRAMA COMPLETO DEL MOUSE PARA PERSONAS COM DISCAPACIDAD MOTRIZ

A.2 PROGRAMA COMPLETO EN TIEMPO DIFERIDO

```
clear all
close all
clc

I=imread('retrato.JPG');
figure, imshow(I)
title('visualizacion del retrato')

       I2=rgb2gray(I);
       I3 = im2double(I2); % CAMBIO A DOUBLE
       figure, imshow(I3)
title('imagen doble')
        %Detección de bordes:
    BW1=edge(I3,'prewitt');
    figure, imshow(BW1)
title('técnica de prewitt')
        % Eliminación de pixeles
     BW2 = bwareaopen(BW1,80);
     figure, imshow(BW2)
     title('imagen sin ruido')
%      impixelregion

     %conectividad de objetos
    [M,NUM]=bwlabel(BW2,8);



     % Operaciones morfológicas:
     se = strel('disk',1);
```

```
     BW11 = imclose(BW2,se);
     BW12 = imfill(BW11,'holes');
     figure, imshow(BW12)
title('relleno de objetos cerrados')

%ubicacion de los valores uno en la Matriz M
% [F,C] = find (M==NUM);

%PROPIEDADES DE LA IMAGEN
stats = regionprops(M,'all'); %todas las propiedades

for k=1:1:NUM
 a = stats(k).Area;
p=stats(k).Perimeter;
end

 %RECONCIMIENTO DE CÍRCULO
   for  k=1:1:NUM
         [R,S] = find (M==k); %matriz de objetos
       for i=1:length(S);
           punts=S(i);
           puntr=R(i);
       end

          %LONGITUDES DE LAS FILAS
          a=1;
          for w=min(R):max(R);
          numeror=find(R==w);
          longitudr(a)=length(numeror);
          a=a+1;
          end
          longitudr;
         %LONGITUDES DE LAS COLUMNAS
```

```
b=1;
for x=min(S):max(S);
numeros=find(S==x);
longituds(b)=length(numeros);
b=b+1;
end
longituds;

% CENTROIDE
  c=stats(k).Centroid;
  x = c(:, 1);  %centroide componente x
  y = c(:, 2);    %centroide componente y

  %RECONOCIMIENTO DEL MOUSE
  x1=int32(x);  %número entero de x
  y1=int32(y);  %número entero de y

  %obtención del número decimal
  x2=1000*x;
  x3=1000*x1;
  x4=abs(x3-x2);
  x5=0;
  if x4<=499 && x4>=450;
      x5=x1+1;
  else x4<450 && x4>499;
      x5=x1;
  end

  y2=1000*y;
  y3=1000*y1;
  y4=abs(y3-y2);
  y5=0;
  if y4<=499 && y4>=450;
```

```
    y5=y1+1;
else y4<450 && y4>499;
    y5=y1;
end
x5;
y5;

 respr=int32(max(R)+min(R))/2;
 resps=int32((max(S)+min(S))/2);

 distsmax=abs(x5-max(S));
 distsmax1=int32(distsmax);
 distsmin=abs(x5-min(S));
 distsmin1=int32(distsmin);
 distrmax=abs(y5-max(R));
 distrmax1=int32(distrmax);
 distrmin=abs(y5-min(R));
 distrmin1=int32(distrmin);

 %obtención del número decimal para distsmax
distsmax2=1000*distsmax;
distsmax3=1000*distsmax1;
distsmax4=abs(distsmax3-distsmax2);
distsmax5=0;
if distsmax4<=499 && distsmax4>=450;
    distsmax5=distsmax1+1;
else distsmax4<450 && distsmax4>499;
    distsmax5=distsmax1;
end

 %obtención del número decimal para distsmin
distsmin2=1000*distsmin;
distsmin3=1000*distsmin1;
```

```
distsmin4=abs(distsmin3-distsmin2);
distsmin5=0;
if distsmin4<=499 && distsmin4>=450;
    distsmin5=distsmin1+1;
else distsmin4<450 && distsmin4>499;
    distsmin5=distsmin1;
end

%obtencion del número decimal para distrmax
 distrmax2=1000*distrmax;
distrmax3=1000*distrmax1;
distrmax4=abs(distrmax3-distrmax2);
distrmax5=0;
if distrmax4<=499 && distrmax4>=450;
    distrmax5=distrmax1+1;
else distrmax4<450 && distrmax4>499;
    distrmax5=distrmax1;
end

%obtención del número decimal para distrmin
distrmin2=1000*distrmin;
distrmin3=1000*distrmin1;
distrmin4=abs(distrmin3-distrmin2);
distrmin5=0;
if distrmin4<=499 && distrmin4>=450;
    distrmin5=distrmin1+1;
else distrmin4<450 && distrmin4>499;
    distrmin5=distrmin1;
end

distsmax5;
distsmin5;
distrmax5;
```

```
            distrmin5;

           if (distrmax5==distrmin5 ||
distrmax5==(distrmin5+1) || (distrmax5)==distrmin5-1) &&
(distsmin5==distsmax5 || (distsmin5)==(distsmax5+1) ||
distsmin5==(distsmax5-1)) && distrmax5>=12 && distrmin5>=12
&& distsmax5>=14 && distsmin5>=14
           disp('es circulo')
           k_circulo=k;
           break
           else
           disp('no es circulo')
           end
   end

    D = plot(S,R,'*r');
    grid on
    hold on

     % CENTROIDE
        c=stats(k_circulo).Centroid;
        xc = c(:, 1);
        yc = c(:, 2);
        plot(x, y, '*b');
        axis ij;
        axis equal;
        hold on
        xlabel('x');
        ylabel('y');
title('Objeto Localizado');

vector = [xc , yc]
```

A.3 PROGRAMA COMPLETO EN TIEMPO REAL

```
 clear all
close all
clc

import java.awt.Robot;
import java.awt.event.*;
mouse = Robot;
%creacion del video, vid= video de entrada
vid = videoinput('winvideo',1,'RGB24_352x288');

%vista preliminar del video de entrada
preview(vid)

%-------------------------------------------------------
set(vid,'TriggerRepeat',Inf);
vid.FrameGrabInterval = 30;
start(vid)

while(vid.FramesAcquired<=30) % Stop after 20 frames; El
número de imagenes obtenidas a partir del flujo de vídeo.
%      I = getsnapshot(vid);
     data = getdata(vid,1);
      %imshow(data);
     I=data;

%---------------------------------------
%Obtencion de una imagen

%I = getsnapshot(vid);
    %escala de grises con delineacion doble
         I2=rgb2gray(I);
         I3 = im2double(I2); % CAMBIO A DOUBLE
```

```
    %Detección de bordes:
    BW1=edge(I3,'prewitt');

    %ELIMINAR DETALLES PEQUEÑOS EN LAS IMAGENES

     BW1 = bwareaopen(BW1,80);
    [M,NUM]=bwlabel(BW1,8);

     % Operaciones morfológicas:
     se = strel('disk',1);
     BW11 = imclose(BW1,se);
     BW12 = imfill(BW11,'holes');

%ubicacion de los valores uno en la Matriz M
[F,C] = find (M==NUM);

%PROPIEDADES DE LA IMAGEN
stats = regionprops(M,'all'); %todas las propiedades

for k=1:NUM
 a = stats(k).Area;
 p=stats(k).Perimeter;
end
k_circulo=1

    %DETERMINACION DE CIRCULO PARA CADA OBJETO
    for  k=1:1:NUM
         [F,C] = find (M==k);
        for i=1:length(C)
            puntf=F(i);
            puntc=C(i);
        end
```

```
    %LONGITUDES DE LAS FILAS
    z=1;
for j=min(C):max(C);
    numeroc=find(C==j);
    longitudc(z)=length(numeroc);
    z=z+1;
end
    longitudc;

 %LONGITUDES DE LAS COLUMNAS
    y=1;
for m=min(F):max(F);
    numerof=find(F==m);
    longitudf(y)=length(numerof);
    y=y+1;
end
     longitudf;
      % CENTROIDE
    c=stats(k).Centroid;
    x = c(:, 1);  %centroide componente x
    y = c(:, 2);    %centroide componente y

    %RECONOCIMIENTO DEL MOUSE
    x1=int32(x);  %número entero de x
    y1=int32(y);  %número entero de y

    %obtención del número decimal
    x2=1000*x;
    x3=1000*x1;
    x4=abs(x3-x2);
    x5=0;
    if x4<=499 && x4>=450;
        x5=x1+1;
```

```
else x4<450 && x4>499;
    x5=x1;
end

y2=1000*y;
y3=1000*y1;
y4=abs(y3-y2);
y5=0;
if y4<=499 && y4>=450;
    y5=y1+1;
else y4<450 && y4>499;
    y5=y1;
end
x5;
y5;

 respf=int32(max(F)+min(F))/2;
 respc=int32((max(C)+min(C))/2);

 distcmax=abs(x5-max(C));
 distcmax1=int32(distcmax);
 distcmin=abs(x5-min(C));
 distcmin1=int32(distcmin);
 distfmax=abs(y5-max(F));
 distfmax1=int32(distfmax);
 distfmin=abs(y5-min(F));
 distfmin1=int32(distfmin);

 %obtención del número decimal para distsmax
distcmax2=1000*distcmax;
distcmax3=1000*distcmax1;
distcmax4=abs(distcmax3-distcmax2);
distcmax5=0;
```

```
if distcmax4<=499 && distcmax4>=450; %467
    distcmax5=distcmax1+1;
else distcmax4<450 && distcmax4>499;
    distcmax5=distcmax1;
end

 %obtención del número decimal para distcmin
distcmin2=1000*distcmin;
distcmin3=1000*distcmin1;
distcmin4=abs(distcmin3-distcmin2);
distcmin5=0;
if distcmin4<=499 && distcmin4>=450;
    distcmin5=distcmin1+1;
else distcmin4<450 && distcmin4>499;
    distcmin5=distcmin1;
end

%obtención del número decimal para distrmax
 distfmax2=1000*distfmax;
distfmax3=1000*distfmax1;
distfmax4=abs(distfmax3-distfmax2);
distfmax5=0;
if distfmax4<=499 && distfmax4>=450;
    distfmax5=distfmax1+1;
else distfmax4<450 && distfmax4>499;
    distfmax5=distfmax1;
end

%obtención del número decimal para distfmin
distfmin2=1000*distfmin;
distfmin3=1000*distfmin1;
distfmin4=abs(distfmin3-distfmin2);
distfmin5=0;
```

```
            if distfmin4<=499 && distfmin4>=450;
                distfmin5=distfmin1+1;
            else distfmin4<450 && distfmin4>499;
                distfmin5=distfmin1;
            end

            distcmax5;
            distcmin5;
            distfmax5;
            distfmin5;

          if ((distfmax5==distfmin5 ||
distfmax5==(distfmin5+1) || (distfmax5-1)==distfmin5 ||
(distfmax5+1)==distfmin5 || distfmax5==(distfmin5-1)) &&
(distfmax5>=12 && distfmin5>=12) && (distcmin5==distcmax5 ||
(distcmax5+1)==distcmin5 || (distcmax5-1)==distcmin5 ||
(distcmax5+1)==distcmin5 || distcmax5==(distcmin5-1)) &&
(distcmax5>=14 && distcmin5>=14) && (x5==respc || x5==(respc-
1)) && (y5==respf || respf==(y5+1) || respf==(y5-1)))
           disp('es circulo')
           k_circulo=k;
           break
           else
           disp('no es circulo')
           end
    end

         %valor del centroide
        c=stats(k_circulo).Centroid;
        xc = c(:, 1);
        yc = c(:, 2);

    t0 = clock;
```

```
            vector = int32([xc, yc])
            vector1=(vector+1);
            vector1=(vector-1);
            vector1=vector;

          dlmwrite('dato_vector_tiempo.txt', vector1, '-
          append')

          mouse.mouseMove(xc, yc);
        % localización del mouse
     mouse_loc =
java.awt.MouseInfo.getPointerInfo.getLocation
            pos_cuadro = [xc yc]
            pause (0.001)
            tiempo = etime(clock,t0);
            tiempo1=int32(100*(tiempo))
            dlmwrite('dato_vector_tiempo.txt', tiempo1, '-
          append',...
          'newline', 'pc');

         %muestra el cuadro de click derecho y click
izquierdo
            if tiempo1>=10
                cuadro=figure('Position',[x 14*y 260 70],...
               'Menu','None',
               'Name','HeadMouse','NumberTitle','off');
                pos=get(cuadro,'Position');

                izq=uicontrol('ButtonDownFcn',
'pushbutton',...
                'String', 'Izquierdo',...
                'Position', [10 10 100 50],...
                'BackgroundColor',[.2 .8 0],...
```

```
                'FontWeight','bold',...
                'TooltipString','Izquierdo');
                    disp('aplasto el izquierdo')

                der=uicontrol('ButtonDownFcn',
'pushbutton',...
                'String', 'Derecho',...
                'Position', [150 10 100 50],...
               'BackgroundColor',[.2 .8 0],...
                'FontWeight','bold',...
                'TooltipString','Derecho');
                    disp('aplasto el derecho')
                 %clic izquierdo
                  mouse.mouseMove((xc+50), (yc+100));
                  pause(2)

                  %clic derecho
                  mouse.mouseMove((xc+200), (yc+100));
                  pause(2)
                          delete (cuadro)
          end

end


 stop(vid)
closepreview(vid);
```