UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Detección del punto de observación de un usuario sobre una pantalla mediante una webcam

Máster Universitario en Ingeniería de Telecomunicaciones

Autor: Camacho Raya, Emma

Tutor: Bescós Cano, Jesús
Departamento de VPULab

FECHA: Julio, 2018

Agradecimientos

En primer lugar, quiero agradecer a mi tutor Jesús Bescós Cano, por darme la posibilidad de llevar a cabo este proyecto y por toda la ayuda prestada, a su disponibilidad y diligencia en todo momento para cualquier duda que apareciera.

En segundo lugar, agradecer a todos los profesores que me han aportado los conocimientos y con los que he aprendido para poder abordar este proyecto.

Gracias también a mi familia, mis padres que siempre me han apoyado en todo y a mi hermana por estar siempre ahí.

A mis amigos por todos los años juntos y los buenos momentos.

Finalmente, agradecer a David por sus ánimos y consejos y por ayudarme en todo lo que ha podido.

Emma Camacho Raya

Junio, 2018

Resumen

La motivación principal de este trabajo ha sido investigar e implementar un método para estimar el punto de observación de un usuario en una pantalla con la mayor precisión posible, usando sólo la webcam que disponga el ordenador o *tablet* del usuario.

Recientes estudios han investigado la interacción del usuario con las máquinas usando la mirada, surgiendo así soluciones para métodos de comunicación de personas discapacitadas, nuevas formas de controlar el ratón y el teclado, o incluso nuevos métodos de aprendizaje, como, por ejemplo, podría seguirse la mirada mientras un niño realiza sus primeras lecturas, o conocer que estímulos llaman más la atención en juegos educativos.

Los métodos actuales de seguimiento de la mirada utilizan iluminación infrarroja, cámaras de alta calidad de vídeo, y requieren una posición relativa estable entre el ojo del usuario y la cámara. Pero estas cámaras son caras y no son viables para los dispositivos cotidianos. Éstos deben lidiar con recursos computacionales limitados, baja resolución de imágenes usando la webcam frontal integrada, condiciones adversas de luz, tamaños pequeños de pantalla donde mapear la mirada o movimientos de la cabeza del usuario.

Después de la investigación del estado del arte, se ha propuesto e implementado una posible solución con la que se ha tenido que tener en cuentas una serie de aspectos previos. Estos aspectos previos han sido encontrar una serie de puntos característicos en la imagen del usuario, que serán necesarios para estimar el punto de observación mediante la solución propuesta.

La evaluación del funcionamiento y del cumplimiento de unos requisitos previos se realizará mediante voluntarios a quienes se les ha sometido a una serie de pruebas. Se ha tenido en cuenta la precisión a la hora de estimar el punto de observación tanto del eje horizontal como del vertical, implementándose dos métodos para evaluar éste último, entre los cuales se realizará una comparativa. Con todo lo anteriormente expuesto, se han extraído una serie de conclusiones.

Palabras clave

Gaze-tracking, detección de la mirada, OpenCV, detección del iris, landmarks.

Abstract

The main motivation behind this work has been to investigate and implement a method of estimating the gaze of a user with the possible precision. The camera have to be the computer webcam.

There are many current studies on the interaction between machines and the user's gaze, for example, communication methods of disabled people, new ways of controlling the mouse and the keyboard or even new learning methods: follow the look of a child learning to read or know what stimuli call more attention in educational games.

Current methods for gaze tracking use infrared illumination, high-quality video cameras, and require a stable relative position between the user's eye and the camera. However, in normal devices there are many limitations.

In this work, a possible solution has been proposed and implemented. This solution need to find a series of characteristic points in the user's image, this will be the first step.

The evaluation of the correct work of some prerequisites will be through volunteers who will undergo a series of tests. These tests will calculate the accuracy in the gaze localization in the horizontal and vertical axis. There are two methods to evaluate the vertical axis and a comparison will be made between both, and the appropriate conclusions have been drawn.

Key words

Gaze-tracking, eye-gaze detection, OpenCV, iris detection, landmarks.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

Hoy en día existe una gran expansión en el mercado de *tablets*, teléfonos móviles y ordenadores personales, que acompañan a millones de personas en su día a día. La manera de interactuar con estas tecnologías es, principalmente, mediante la pantalla táctil o con el ratón y el teclado.

Recientes estudios están investigando un nuevo método de interacción: la mirada del usuario. Esto haría posible que personas físicamente discapacitadas puedan comunicarse mediante la detección de la dirección de la mirada y el parpadeo de los ojos [1], incluso podría llegar a incorporarse en los dispositivos una nueva interfaz con la que cualquier usuario pueda realizar la función de un ratón y un teclado con la mirada [2] [3].

Pero no queda ahí el uso del *eye-gaze detection* o localización del punto de observación de un usuario. Así, otro campo de aplicación es la detección de fatiga de un conductor, comprobando si cierra los ojos durante mucho tiempo, pudiendo así prevenir accidentes [4]. También puede ser útil para conocer a que parte de la pantalla mira más un usuario, cuando navega por internet, para estudios de publicidad o diseño u optimización de páginas web.

Otro ámbito muy interesante donde tendría mucha utilidad es en el del aprendizaje: podría seguirse la mirada mientras un niño está leyendo, o conocer que estímulos llaman más la atención en juegos educativos. Esta información sería útil para conocer cómo el procesamiento cognitivo y léxico afecta a la lectura y al aprendizaje.

Los métodos actuales para el seguimiento de la mirada utilizan iluminación infrarroja, cámaras de alta calidad de video, y requieren una posición relativa estable entre el ojo del usuario y la cámara. En estos métodos, también conocidos como *eye-tracker*, el ojo está iluminado por una luz de diodo infrarrojo que es invisible al usuario y no interrumpe su interacción con el ordenador. Las fuentes IR instaladas correctamente en la cámara, producen reflexiones únicas en el ojo del usuario (llamadas "destellos"). Usan el contraste para localizar el centro de la pupila y crear un reflejo en la córnea a través de luz infrarroja (ver Figura 1-1), para así rastrear entre otros factores, el movimiento de los globos oculares, la dilatación de la pupila y el parpadeo del sujeto. [5].

Pero estas cámaras son caras y no son viables para los dispositivos cotidianos. Éstos deben lidiar con recursos computacionales limitados, baja resolución de imágenes usando la webcam frontal integrada, condiciones adversas de luz, tamaños pequeños de pantalla donde mapear la mirada o movimientos de la cabeza del usuario.

En este proyecto se intenta realizar un programa que pueda detectar la mirada de un usuario sobre una pantalla con la mayor precisión posible usando sólo la webcam que disponga el ordenador o *tablet* del usuario.

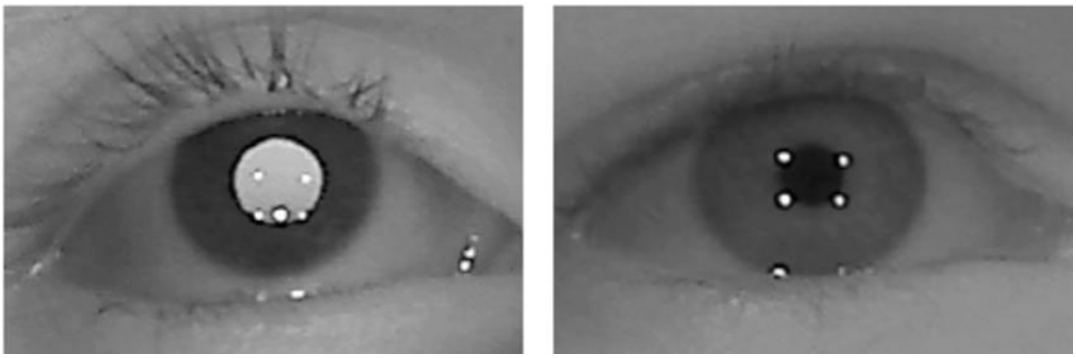

Figura 1-1: Ojo iluminado por luz infrarroja. Fuente: [5]

1.2 Objetivos

Este Trabajo Fin de Máster busca realizar el seguimiento de la mirada de un usuario mientras utiliza una pantalla con los únicos recursos de una webcam y la librería de OpenCV [6].

El estudio incluye los siguientes objetivos parciales:

- Estudio previo de artículos realizados acerca del *gaze-tracking* y *pupil and iris detection*.
- Elección de las metodologías de detección del centro de la pupila y la mirada.
- Análisis de la librería de OpenCV.
- Desarrollo de un programa que detecte, con la mayor precisión posible, la pupila, y a partir de una calibración previa, realizar el seguimiento de la mirada.
- Evaluación y diseño de escenarios de pruebas.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1:** Motivación, objetivos del proyecto y estructura de la memoria.
- **Capítulo 2:** Estado del arte de algoritmos de seguimiento de la mirada y detección del centro de la pupila. Explicación de los procedimientos elegidos. Descripción de los programas y las librerías utilizadas para el desarrollo.
- **Capítulo 3:** Diseño y desarrollo del código y métodos utilizados.
- **Capítulo 4:** Experimentos y resultados obtenidos en la evaluación del seguimiento de la mirada.
- **Capítulo 5:** Conclusiones obtenidas tras el análisis de resultados del trabajo y posibles vías para el trabajo futuro.

2 Estado del arte

2.1 Introducción y aproximación inicial

En los últimos años se ha experimentado un gran avance en el estudio del seguimiento de la mirada sin la necesidad de usar luces infrarrojas, existiendo multitud de artículos al respecto. En estos estudios se siguen distintas técnicas para intentar lograr la máxima precisión y subsanar carencias que tengan otros trabajos o técnicas, pero muchos de ellos tienen ciertas limitaciones respecto a la calidad de la cámara, la iluminación, o movimientos del usuario.

Los distintos métodos para el seguimiento del punto de observación, basados en la clasificación realizada en [7], suelen clasificarse de la siguiente manera:

1. *Appearance-Based:* Basados en la apariencia, en las que se utilizan las intensidades de los píxeles de la imagen del ojo. Un conjunto de muestras se utiliza para distinguir distintas posturas del usuario y miradas, y así mapearlas a la pantalla. Dentro de este grupo podemos encontrar técnicas como *Neural Network* y *Local Linear Interpolation* (LLI). La principal desventaja es el coste computacional.

2. *Feature-Based;* Basados en características, donde también se realiza un mapeo para calcular la mirada, pero la diferencia es que utiliza vectores de características. Busca romper la conexión directa entre las intensidades de píxel y el vector de entrada final, en un intento de aumentar la robustez a los cambios de iluminación con respecto a los basados en apariencia. Algunas de las características usadas son las siguientes:
    a. Posición del pixel de un punto clave, como son el centro del iris, las esquinas de los ojos y el párpado.
    b. Las posiciones relativas de los puntos anteriores, vectores que conecten dos píxeles.
    c. Características de visión artificial como histogramas orientados a gradientes (HOG) o patrones binarios locales (LBPs).
    d. Características de grupos y sumas de intensidades de los píxeles.

    En este método podemos encontrar trabajos acerca de la relación entre el centro del iris y las esquinas de los ojos, conocido como vector *Pupil Center-Eye Corner (PC-EC).*

3. *Model-Based:* Basados en modelos, en los que se trata de modelar los ojos y tal vez incluso la cara. La mirada se calcula geométricamente usando los parámetros del modelo. Requiere de parámetros para modelar el ojo en 2D o 3D. En el caso de 2D pueden ser los puntos que definen el borde del iris para aproximarlo a una elipse, en caso del 3D es más complejo, necesita incluir posiciones del centro del globo ocular u otros puntos faciales. El primer inconveniente es que necesita de imágenes con una alta resolución.

2.2 Técnicas específicas relacionadas con la aproximación planteada

A continuación, se expondrán las distintas técnicas y algoritmos encontrados y usados para las distintas etapas del trabajo. Lo primero es tomar la decisión acerca de que método usar para la interpolación y seguidamente decidir el camino para llegar a este método. En el capítulo 3 se expondrán más detalladamente el porqué de las elecciones tomadas. La última técnica que se comentará se usará para compensar pequeños desplazamientos en la posición de la cabeza mediante homografías.

2.2.1 Algoritmos de interpolación de la mirada a la pantalla.

Como se comentó en el apartado 2.1, existen varias maneras de resolver esta cuestión. Existen ejemplos concretos de las distintas clasificaciones que se comentan a continuación:

- Ejemplos de *Appearance-Based* como en [2], utilizan la función Haar Cascade de OpenCV para entrenar al programa para que detecte la pupila del ojo. Según el tamaño de píxeles de la pupila, se estima en qué dirección, y cuánto se ha movido, desde un punto inicial. También se realizan pruebas en personas con gafas. Otro ejemplo de este método está publicado en [8], donde se maneja la descomposición de la imagen del ojo, que tiene en cuenta las variaciones causadas por la dirección de la mirada, la apariencia del ojo y los cambios en el recorte de la imagen. Usando esta descomposición, encuentra las 3 muestras de entrenamiento más similares y usa LLI para calcular la dirección de la mirada. En [9] se utilizan dos métodos para estimar la mirada, primero utilizan la imagen binaria del ojo, y según el valor de los pixeles de la imagen, donde los pixeles tengan un valor más alto, es decir donde la imagen sea más oscura, será el iris. Con este método es fácil conocer si un usuario mira hacia la derecha o izquierda, pero no hacia arriba o hacia abajo. El segundo método que proponen es utilizar la función de *motion template* de OpenCV, la cual calcula cuanto y en qué dirección se mueve un objeto y la aplican al ojo del usuario, realizando una calibración previa, con el inconveniente de que es necesario volver a calibrar cada vez que se detecta un movimiento de cabeza.

- Aplicaciones de *Feature-Based* como [10], proponen utilizar el vector que une el centro del iris con la esquina interior del ojo tras una previa calibración. Se describen dos métodos para el mapeo: mediante regresión polinómica, en el que se propone un polinomio de segundo orden y mediante un Kernel de función de base radial (RBF). En [11] se utilizan las dos comisuras de los ojos, la interna y la externa, normalizando el vector PC-EC, dividiéndolo entre la distancia Euclidea entre las dos esquinas. En [12] se proponen métodos para detectar el centro del iris y la comisura del ojo con precisión subpíxel. Usan un mapeo lineal 2D para estimar las posiciones de las miradas desde los vectores de características.

- De *Model-Based*, en [13] se usa el contorno del iris detectado para calcular el centro del globo ocular y después una calibración para obtener el eje óptico, para ello se utilizan dos cámaras. En [14] se detectan los bordes del iris para aproximarlo a una elipse con el método de RANSAC, calculando la orientación y la posición de cada ojo en 3D. En todos los ejemplos basados en modelos, se necesitan imágenes de muy buena calidad.

La mayoría de los métodos comentados necesitan cámaras con una buena resolución, o un gran coste computacional, por lo que se optó por utilizar métodos *feature-based*, los más usados en imágenes de peor calidad, como las capturadas con *webcams*. Estos métodos, ya explicados, se basan en características que permiten aumentar la robustez, siendo las características más utilizadas el centro del iris y las comisuras de los ojos. Como punto de partida se plantea usar las dos comisuras o esquinas del ojo, y se extraerán de los dos ojos, creando una relación entre los 6 puntos obtenidos para la interpolación a la pantalla, calculando los vectores PC-EC provocando así que no se dependa en exceso de si la cabeza se mueve o no. Es necesario explicar ya la decisión tomada para poder entender el porqué de los siguientes pasos, pero esta decisión será comentada en detalle en el capítulo 3.

Se propone un sistema polinómico con el que relacionar los distintos vectores con el punto de observación en la pantalla, basado en [11].

$$\begin{pmatrix} \boldsymbol{POx} \\ \boldsymbol{POy} \end{pmatrix} = \boldsymbol{C} \begin{pmatrix} \mathbf{1} \\ \boldsymbol{v_{Nx}} \\ \boldsymbol{v_{Ny}} \\ \boldsymbol{v_{Nx} v_{Ny}} \\ \boldsymbol{v_{Nx}^2} \\ \boldsymbol{v_{Ny}^2} \end{pmatrix}, \qquad (1)$$

Donde ***POx*** y ***POy*** son las coordenadas en la pantalla de dicho punto de observación, $\boldsymbol{v_N}$ es el vector PC-EC normalizado y ***C*** es una matriz de coeficientes desconocida en principio. Para obtenerla será necesario una calibración previa donde el usuario deberá mirar en una serie de puntos indicados, con los cuales se resolverá la matriz.
Para calcular el vector $\boldsymbol{v_N}$ se calcula primero los vectores normalizados entre el centro de la pupila y cada una de las dos esquinas de cada ojo.

$$\boldsymbol{v_{N0_i}^{ojo}} = \frac{\boldsymbol{PC^{ojo} - EC_i^{ojo}}}{\boldsymbol{ECD^{ojo}}}, \qquad (2)$$

$$\boldsymbol{v_{N1}^{ojo}} = \frac{\mathbf{1}}{\mathbf{2}} \left(\boldsymbol{v_{N0_{interior}}^{ojo}} + \boldsymbol{v_{N0_{exterior}}^{ojo}} \right), \qquad (3)$$

Siendo $\boldsymbol{v_{N0_i}^{ojo}}$ los vectores normalizados, con i = {interior, exterior} y ojo = {izquierda, derecha}, para los extremos del ojo interior y exterior correspondientemente. $\boldsymbol{PC^{ojo}}$son las coordenadas del centro de la pupila, $\boldsymbol{EC_i^{ojo}}$coordenadas de cada extremo y $\boldsymbol{ECD^{ojo}}$ la distancia Euclidea entre los dos extremos. Así se obtendrán $\boldsymbol{v_{N0_{interior}}^{ojo}}$ y $\boldsymbol{v_{N0_{exterior}}^{ojo}}$; calculando su media se tendrá un $\boldsymbol{v_{N1}^{ojo}}$ para cada ojo. El $\boldsymbol{v_N}$ final vuelve a ser la media entre los $\boldsymbol{v_{N1}^{ojo}}$ de cada ojo.

$$\boldsymbol{v_N} = \frac{\mathbf{1}}{\mathbf{2}} \left(\boldsymbol{v_{N1}^{izquierda}} + \boldsymbol{v_{N1}^{derecha}} \right), \qquad (4)$$

Para resolver este método es necesario obtener los vectores $\boldsymbol{v_N}$, para ello lo primero es detectar la región de interés, en este caso el ojo, y conocer que características del ojo se necesitan para la búsqueda de técnicas de extracción de estas características, las cuales son el centro del iris y las comisuras de los ojos (ver Figura 2-1). A continuación, se procederá a explicar cada uno de los métodos existentes utilizados, tanto para la extracción de la región de interés, como para la detección de las características necesarias, concluyendo el capítulo con la explicación de cómo se utilizarán estas características para aproximar el punto de observación del usuario.

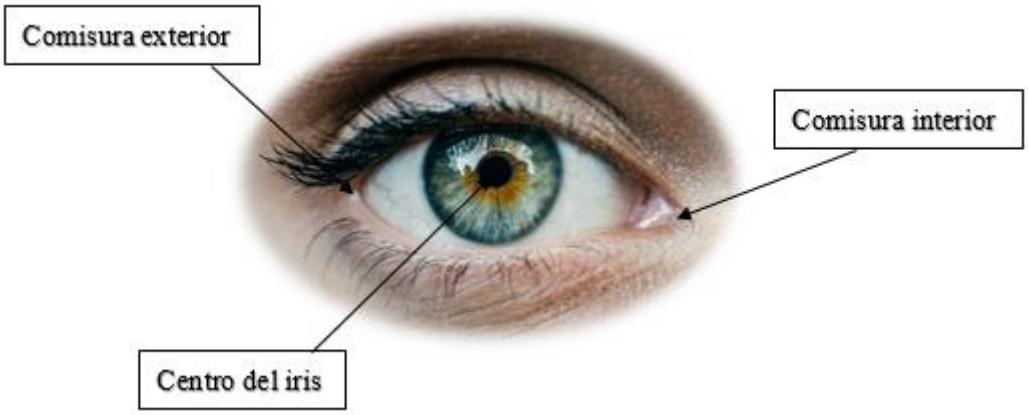

Figura 2-1: Características de interés del ojo

2.2.2 Algoritmo de detección de la cara y el ojo

Como se ha explicado en la sección 2.2.1, lo principal para cada *frame* obtenido es detectar la región de interés, los ojos. Previamente, es necesario acotar la zona de búsqueda, en este caso detectar la cara, así será mucho más fácil encontrar los ojos, ya que siempre están en el mismo lugar de la cara.

La técnica más usada por ser rápida y eficaz es *Haar Cascade Classifiers* [15]. Los clasificadores tipo Haar se basan en unas características predefinidas, con el mismo nombre, que sirven para reconocer formas en una imagen, como se puede observar en la Figura 2-2. Estas características usan los cambios de contraste entre rectángulos de pixeles adyacentes para determinar áreas claras y oscuras. Dos o tres rectángulos con diferente contraste forman una única característica Haar. Estas características se pueden ampliar y reducir fácilmente, aumentando o disminuyendo el tamaño del grupo de píxeles por lo que permite que las características Haar detecten objetos de varios tamaños con relativa facilidad.

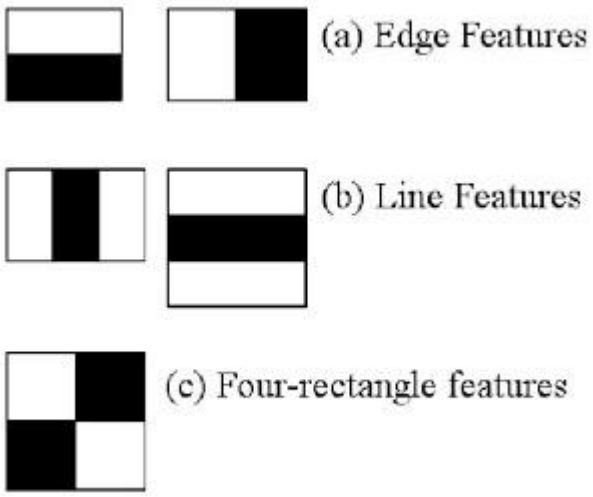

Figura 2-2: Características Haar. Fuente: [15]

Los rectángulos se calculan utilizando una imagen intermedia llamada imagen integral. Esta imagen integral contiene la suma de las intensidades de todos los píxeles a la izquierda y por encima del pixel actual. Usando las diferencias entre dos o tres rectángulos se sacan las características de la imagen. Una ventaja de este sistema es que calcular características de varios tamaños requiere el mismo esfuerzo que de dos o tres pixeles.

El principal objetivo es crear un clasificador de un objeto para entrenarlo, usando un conjunto de ejemplos positivos, con imágenes que contengan el objeto, y de ejemplos negativos, con imágenes que no lo contengan. Una vez entrenado el clasificador puede ser usado para detectar dicho objeto (ver Figura 2-3).

OpenCV incluye una implementación de *Haar Cascade Classifiers* y bases de datos con ejemplos positivos y negativos tanto de caras como de ojos, que serán las usadas en este proyecto para detectarlos [15].

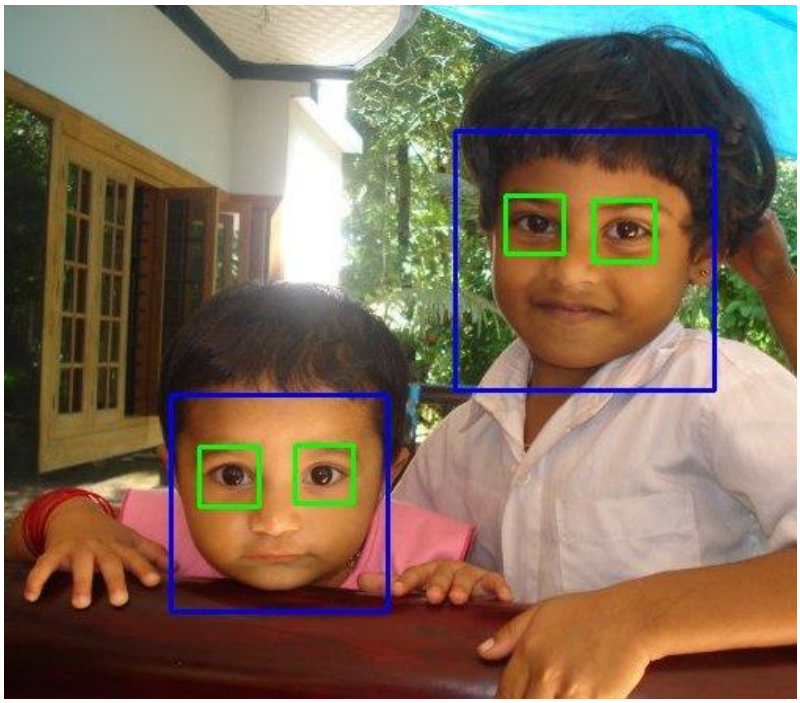

Figura 2-3: Ejemplo de detección de caras y ojos de Haar Cascade Classifiers. Fuente: [15]

2.2.3 Algoritmos de detección del iris y la pupila

Una óptima localización del iris y la pupila es el paso más importante para un buen *gaze-traking.* En muchos trabajos se busca solucionar esto con detección de círculos, ya que el iris podría aproximarse a uno [11][16][17]. Utilizan Circular Hough Transform (CHT), método que necesita de una imagen binaria con el filtro Canny para obtener el contorno. El principal inconveniente es que se necesita de imágenes con cierta calidad. Por este procedimiento y con una resolución como la disponible en este trabajo se tienen falsos negativos sobre todo cuando el iris no está centrado (está desplazado a la derecha o izquierda) o cuando la luminosidad no es muy adecuada, además de que la imagen del ojo es pequeña, lo que provoca que el número de puntos del contorno del iris sea limitado.

Estudios que intentan resolver estos problemas proponen tratar el iris como una elipse [12]. El algoritmo de ajuste de la elipse supone que los puntos de entrada del contorno del iris, pertenecen a una elipse, y utiliza estos puntos para estimar los parámetros de la elipse. También se combina este algoritmo de ajuste de la elipse con la técnica RANdom SAmple Consensus (RANSAC) [14][18], un proceso iterativo que selecciona muchos subconjuntos pequeños y aleatorios de los datos de entrada, utilizando cada subconjunto para ajustarlo a un modelo, y encuentra el modelo que mejor se ajusta al conjunto de datos de entrada. Estos algoritmos mejoran los resultados del CHT, pero se siguen usando cámaras con buena resolución.

Otros estudios se basan en un algoritmo para localizar primero la región del ojo con filtros de Gabor, y luego localizar la pupila con una medida de simetría radial. Sin embargo, la precisión del método disminuye cuando el iris se mueve hacia las esquinas [19]. También se ha propuesto un método basado en curvas “isophotes” para lograr invariancia a cambios lineales de la iluminación, a rotaciones, y para mantener costes computacionales bajos

[20][21]. Las 'isophotes' de una imagen son curvas que conectan puntos con una misma intensidad Una imagen puede ser totalmente descrita por sus 'isophotes'. Además, sus formas son independientes a cambios lineales de iluminación y rotación. Debido a estas propiedades, las 'isophotes' han sido exitosamente utilizadas como características en detección de objetos y segmentación de imágenes. Sin embargo, la exactitud del método falla cuando el iris se mueve hacia los extremos del ojo, detectando como centro del iris las cejas o los extremos del ojo.

Por último, la técnica elegida está basada en gradientes [22]. Este algoritmo puede encontrar el centro del iris a partir de imágenes de baja resolución, de forma eficiente comparado con otros métodos, como se verá en el capítulo 3. Dado **c** como un posible centro y $\boldsymbol{g_i}$ como el vector gradiente en la posición $\boldsymbol{x_i}$**.** Entonces el vector de desplazamiento normalizado $\boldsymbol{d_i}$ debe tener la misma orientación (excepto por el signo) que el gradiente $\boldsymbol{g_i}$ como podemos observar en la Figura 2-4.

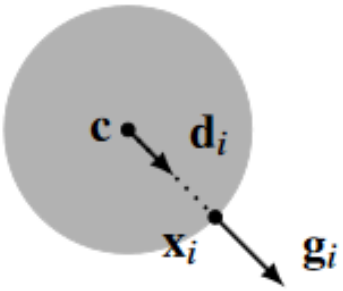

Figura 2-4: Ejemplo de circulo oscuro en fondo blanco con misma orientación de los vectores desplazamiento y gradiente. Fuente: [22]

El centro óptimo $\boldsymbol{c}^*$ será el máximo de cada posible centro, calculado como la suma del producto escalar al cuadrado de los vectores $\boldsymbol{d_i}$ y $\boldsymbol{g_i}$ de cada pixel, Ecuación (5). El vector desplazamiento $\boldsymbol{d_i}$ se calcula restando el vector de la posición de cada pixel $\boldsymbol{x_i}$ menos el vector de las coordenadas de un posible centro $\boldsymbol{c}$ y se normaliza para todas las posiciones de los píxeles, Ecuación (6).

$$\boldsymbol{c}^* = \underset{\boldsymbol{c}}{\boldsymbol{argmax}} \left\{ \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\boldsymbol{d_i^T g_i})^{\boldsymbol{2}} \right\} , \qquad (5)$$

$$\boldsymbol{d_i} = \frac{\boldsymbol{x_i - c}}{\|\boldsymbol{x_i - c}\|_{\boldsymbol{2}}} , \forall \boldsymbol{i} : \ \|\boldsymbol{g_i}\|_{\boldsymbol{2}} = \boldsymbol{1} , \qquad (6)$$

Para calcular el gradiente sirve cualquier método, eligiéndose el operador Sobel. Además, solo se tendrán en consideración los vectores del gradiente con una magnitud significante, Ecuación (7), ignorando las regiones homogéneas, para disminuir la complejidad computacional.

$$\boldsymbol{magnitud_i} = \sqrt{\boldsymbol{g_{ix}^2 + g_{iy}^2}} , \qquad (7)$$

Bajo algunas condiciones el máximo no está bien definido o existen falsos positivos donde no debería, como en las cejas o las gafas. Por esto, se propone incorporar un conocimiento previo sobre el ojo para incrementar la eficiencia. Como la pupila es bastante oscura comparada con la esclerótica y la piel, se aplica un peso $\boldsymbol{w_c}$ para cada posible centro **c** de forma, ya que los centros oscuros son más probables que los brillantes. Se añade de la siguiente forma:

$$\underset{\boldsymbol{c}}{\boldsymbol{argmax}}\ \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{w_c}\left(\boldsymbol{d_i^T g_i}\right)^{\mathbf{2}} \ , \qquad (8)$$

donde $\boldsymbol{w_c}$ es la imagen invertida con un filtro Gaussiano, para evitar problemas de elementos brillantes como el cristal de las gafas.
Además, se propone un post procesado para evitar los fallos con las cejas y el pelo, con un umbral basado en el valor máximo y eliminando los valores que estén en los bordes de la imagen, se vuelve a calcular el máximo y se estima como centro.

2.2.4 Algoritmos de detección de los bordes del ojo.

El siguiente paso consiste en detectar las comisuras de los ojos. En trabajos como [23] utilizan la varianza del brillo de los píxeles en ambas esquinas del ojo para detectar el punto límite exterior del ojo donde la variación es más baja, ya que los píxeles de la piel son más o menos iguales. Pero, aunque tengan buenos resultados en la esquina exterior, la interior es más complicada de detectar. Otro método usado es el explicado en [24], donde se utiliza el método Harris Corner Detector para extraer las comisuras de una imagen binaria. Se necesita una imagen de bastante calidad para poder detectarlos bien, sobre todo el del interior como en el caso anterior, además emplean un sistema con la cámara montada en la cabeza.

Por todo lo indicado y debido a la baja calidad de las imágenes disponibles, como se verá en el capítulo 3, se decidió utilizar una librería llamada DLIB [25], la cual localiza 68 *landmarks* o puntos de referencia de la cara. Estos puntos están relacionados con los ojos, las cejas, la nariz, la boca y el contorno de la cara. Los 68 puntos se pueden observar en la Figura 2-5.

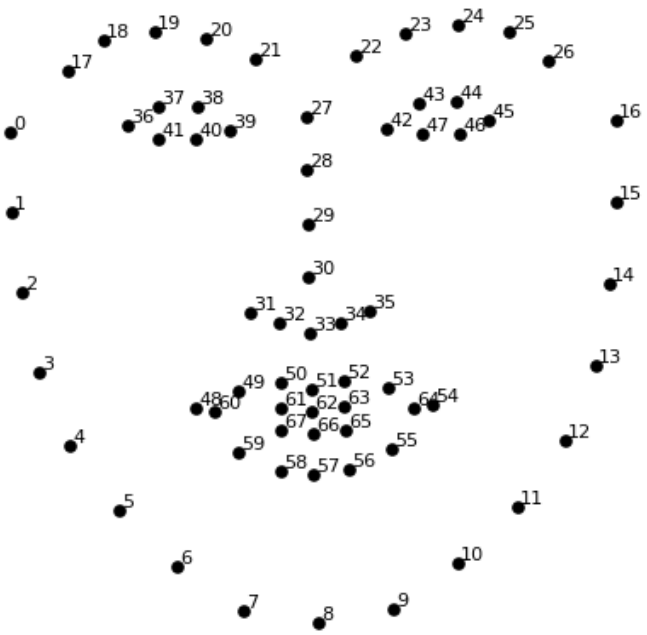

Figura 2-5: Los 68 landmarks de Dlib. Fuente: [25]

Como se puede comprobar los puntos 36, 39, 42 y 45 pertenecen a las 4 esquinas de los dos ojos buscadas. Este proceso es algo lento, pero es bastante preciso ya que tiene en cuenta toda la cara para localizar los puntos de referencia.

2.2.5 Aproximación por mínimos cuadrados.

Para resolver el polinómico expuesto en el capítulo 2.2.1 en la Ecuación (1) se usarán más puntos de los necesarios y así disminuir el error, para esto se hará uso de una aproximación por mínimos cuadrados, una técnica de análisis numérico, en la que, dados un conjunto de datos, se intenta encontrar la función continua que mejor se aproxime a los datos (un "mejor ajuste"), de acuerdo con el criterio del mínimo error cuadrático [26].

Sea $\{(x_k, y_k)\}_{k=1}^{n}$ un conjunto de *n* datos, y sea $\{f_j(x)\}_{j=1}^{m}$ un conjunto de *m* funciones linealmente independientes, se desea encontrar una función $f(x)$ de dicho espacio, es decir, una combinación lineal del conjunto de funciones, de la siguiente forma:

$$\boldsymbol{f(x) = c_1 f_1(x) + c_2 f_2(x) + \cdots + c_m f_m(x) = \sum_{j=1}^{m} c_j f_j(x),} \qquad (9)$$

Quedaría un sistema de la forma:

$$\begin{bmatrix} (f_1, f_1)_d & (f_1, f_2)_d & \cdots & (f_1, f_m)_d \\ (f_2, f_1)_d & (f_2, f_2)_d & & (f_2, f_m)_d \\ \vdots & \vdots & \ddots & \vdots \\ (f_m, f_1)_d & (f_m, f_1)_d & \cdots & (f_m, f_m)_d \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} (f_1, y)_d \\ (f_2, y)_d \\ \vdots \\ (f_m, y)_d \end{bmatrix} \qquad (10)$$

siendo$(h(x), g(x))_d$ el producto escalar discreto:

$$(\boldsymbol{h(x), g(x))_d} = \sum_{\boldsymbol{k=1}}^{\boldsymbol{n}} \boldsymbol{h(x_k) g(x_k)}, \qquad (11)$$

Este método intenta disminuir el error cuadrático medio (ECM), pero siempre existe un error que hay que tener en cuenta. El error cuadrático medio viene dado por:

$$\boldsymbol{Ecm} = \sqrt{\frac{\boldsymbol{1}}{\boldsymbol{n}} \sum_{\boldsymbol{k=1}}^{\boldsymbol{n}} \boldsymbol{(y_k - f(x_k))^2}}, \qquad (12)$$

2.2.6 Homografía

En visión artificial, dos imágenes de la misma superficie plana en el espacio están relacionadas por una homografía [27]. Se determina la matriz de homografía la cual relaciona una imagen con otra rotada o trasladada (ver Figura 2-6). Esta información puede ser usada para encontrar un objeto en una escena, rectificar una imagen, etc.

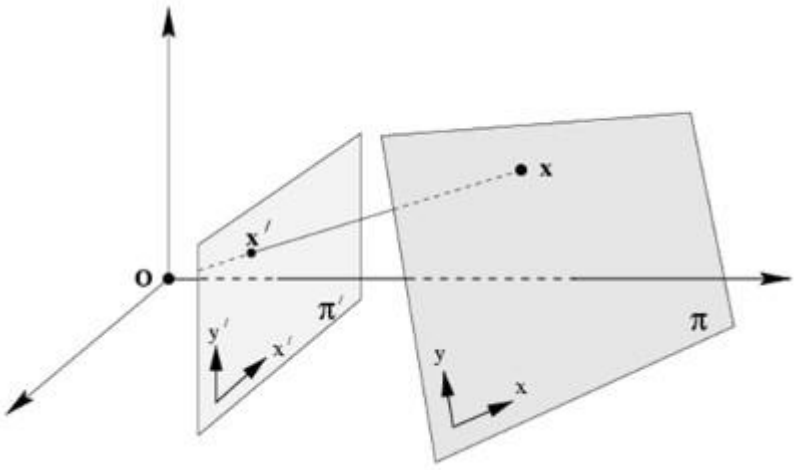

Figura 2-6: Punto de una imagen plana relaciona con otra trasladada. Fuente: [27].

Para usar esta técnica es necesario extraer, previamente, buenas características para localizar un objeto, puntos clave únicos y descriptivos de una imagen, *Features Detection*. Para esto existen varios métodos, SIFT (Scale-invariant feature transform) [28] y SURF (Speeded-Up Robust Features) [29], basado en los mismos principios y pasos que SIFT, pero utilizando un esquema diferente que provee mejores resultados y más rapidez.

La extracción de características es necesaria en las dos imágenes y el siguiente paso sería realizar un *matching* entre ellas (ver Figura 2-7). OpenCV implementa dos estrategias de *matching, Brute-Force Matcher* and *FLANN Matcher* [30].

Brute-Force Matcher es simple, parte de una característica de la primera imagen y busca coincidencias con todas las características de la segunda imagen utilizando algunos cálculos de distancia. Se devuelve el más parecido.

FLANN *(Fast Library for Approximate Nearest Neighbors*) contiene una colección de algoritmos optimizados para buscar rápidamente vecinos cercanos en un conjunto de datos largo y con un número alto de características. Trabaja más rápido que *Brute-Force* cuando hay muchos datos.

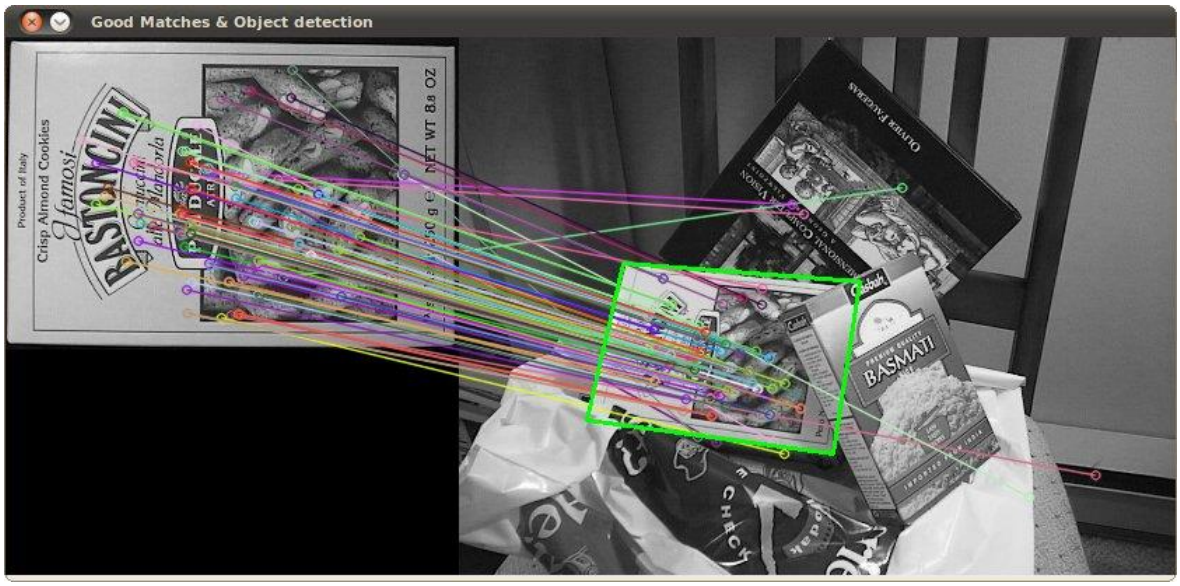

Figura 2-7: Ejemplo de *matching* entre dos imágenes. Fuente: [31].

Cuando se tiene el *matching* entre las dos imágenes podemos obtener la matriz de homografía y encontrar un objeto en una escena o rectificar una imagen.

3 Diseño e Implementación

3.1 Introducción

Como se comentó en el capítulo 1, en este proyecto se intenta encontrar, con la mayor precisión posible, la mirada del usuario sobre la pantalla, teniendo en cuenta las limitaciones existentes. Estas limitaciones son el coste computacional y la calidad de las imágenes, al utilizar la cámara integrada en el dispositivo utilizado, lo que provoca una menor precisión a la hora de detectar las características y por lo tanto de encontrar el punto de observación. Una vez descritos, en el capítulo anterior, los métodos que se han encontrado más adecuados para el proyecto, se procede a desarrollar la implementación, así como los requisitos previos necesarios a cumplir.

3.2 Requisitos previos

El trabajo resultante de este proyecto debe adaptarse a ciertos requisitos necesarios, algunos obligatorios y otros deseables.

- Requisitos computacionales:
  - El sistema operativo usado será Windows
  - El programa debe poder ejecutarse en equipos con un rendimiento no muy alto como en *tablets* y ordenadores portátiles.
  - La cámara usada para grabar las imágenes del usuario será de baja calidad.
- Requisitos funcionales:
  - Funcionar a tiempo real.
  - El programa debe saber localizar la cara y los ojos del usuario para obtener de ellos los puntos característicos.
  - Debe funcionar con distintos usuarios.
  - El usuario tiene una cierta libertad para mover la cabeza, sin movimientos bruscos ni exagerados.
  - Tiene que saber si un usuario está mirando dentro de la pantalla o fuera.
  - Tiene que saber a qué cuadrante de la pantalla está mirando un usuario.
  - Debe detectar en que novena parte de la pantalla se encuentra el punto de observación del usuario.

3.3 Implementación

El desarrollo del programa se llevó a cabo en el entorno Visual Studio con el lenguaje de programación C++ y el uso de las librerías OpenCV y DLIB. El programa se compone de diferentes etapas que consisten en: una calibración, una fase de pruebas y finalmente se utiliza la calibración para obtener el punto de observación del usuario, tanto de la fase de pruebas como de los puntos de la calibración (ver Figura 3-1).

Figura 3-1: Diagrama de las etapas del programa

Dentro de las etapas de calibración y pruebas se sigue un procedimiento parecido, en ambos capturamos un *frame* del cual se obtienen la cara y los ojos del usuario, y seguidamente de los puntos característicos se calculan los vectores (ver Figura 3-2). A continuación, se explicarán en detalle cada procedimiento.

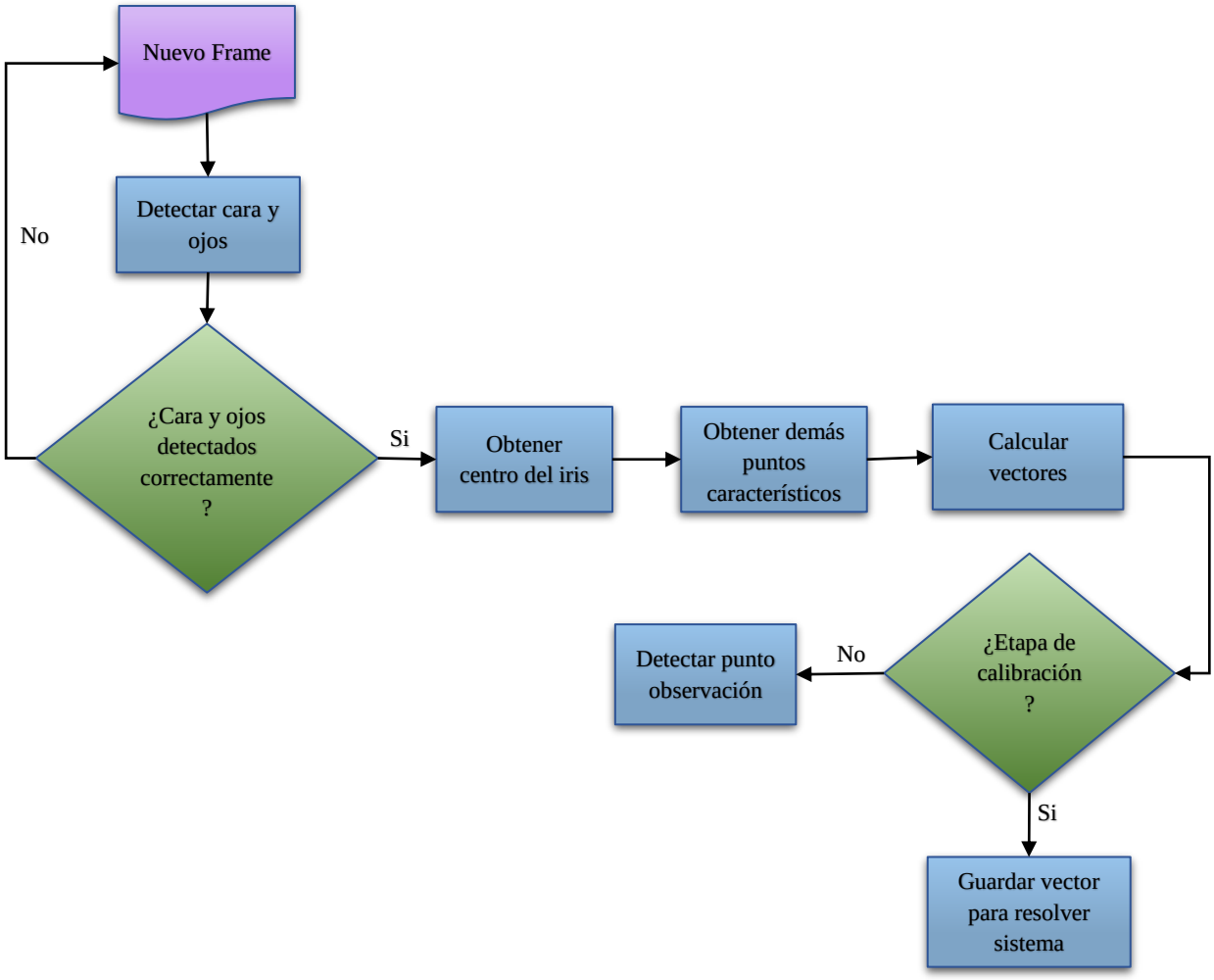

Figura 3-2: Diagrama de las etapas calibración y pruebas

3.3.1 Interpolación de la mirada a la pantalla

De entre todos los métodos expuestos en el apartado 2.2.1, la mayoría necesitan cámaras con una buena resolución, o un gran coste computacional, por lo que se optó por utilizar métodos *feature-based*, los más usados en imágenes de peor calidad, como las capturadas con webcams. Estos métodos, se basan en características que permiten aumentar la robustez, siendo las características más utilizadas el centro del iris y las comisuras de los ojos.

Dentro de los ejemplos comentados, se explicó en detalle en [11], la cual utiliza las dos comisuras de los ojos además de los dos ojos, lo que no ocurre en otros trabajos de este mismo tipo, como [10] ó [12]. Este método proporciona una menor probabilidad de error, por lo que fue el método elegido. Por ejemplo, en [10], utilizan un sistema muy parecido al empleado en esta memoria, si bien, para cada ojo por separado y exclusivamente con la comisura interior.

Como novedad se añadirá una nueva relación para evaluar la distancia entre los párpados superior e inferior de cada ojo y así poder aumentar la precisión para conocer si el usuario mira más arriba o abajo de la pantalla. Se añadirá un nuevo sistema polinómico para evaluar la distancia entre los parpados superior e inferior de cada ojo y conocer la altura de la pantalla a la que mira el usuario. El sistema será como el anterior, ecuación (1), cambiando el vector utilizado. Además, solo servirá para calcular la coordenada ***POy*** del punto de observación sobre la pantalla.

$$(\boldsymbol{POy}) = \boldsymbol{C'} \begin{pmatrix} \mathbf{1} \\ \boldsymbol{v_{Px}} \\ \boldsymbol{v_{Py}} \\ \boldsymbol{v_{Px} v_{Py}} \\ \boldsymbol{v_{Px}^2} \\ \boldsymbol{v_{Py}^2} \end{pmatrix}, \qquad \mathbf{(13)}$$

Estos nuevos vectores $\boldsymbol{v_P}$ son la media entre los cuatro vectores que unen los puntos del párpado superior y los del párpado inferior de ambos ojos (ver Figura 3-3).

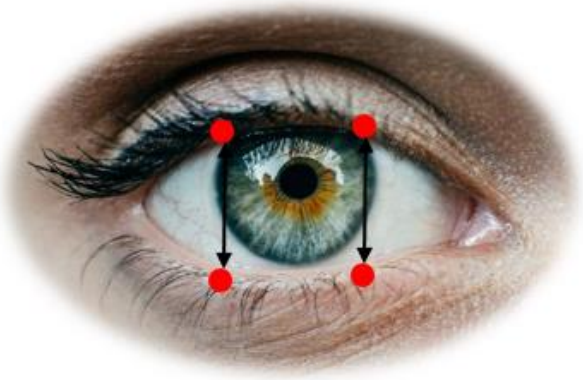

Figura 3-3: Vectores entre los puntos obtenidos de ambos parpados

3.3.2 Detección de la cara y los ojos

En este procedimiento se llevará a cabo el algoritmo explicado en el apartado 2.2.1. OpenCV contiene algunos clasificadores pre-entrenados para caras, ojos, sonrisas, etc, en ficheros XML los cuales se cargan para poder detectar estos elementos. En el trabajo se usarán *haarcascade_frontalface_alt.xml* primero, y tras obtener la región de la cara se usará *haarcascade_eye_tree_eyeglasses.xml* para los ojos. Al detectar los ojos no distingue el derecho del izquierdo por lo que hay que comprobar cuál es el que está más a la izquierda y derecha de la imagen.

En la Figura 3-4 se encuentra un ejemplo de la detección de la cara y los ojos con este método.

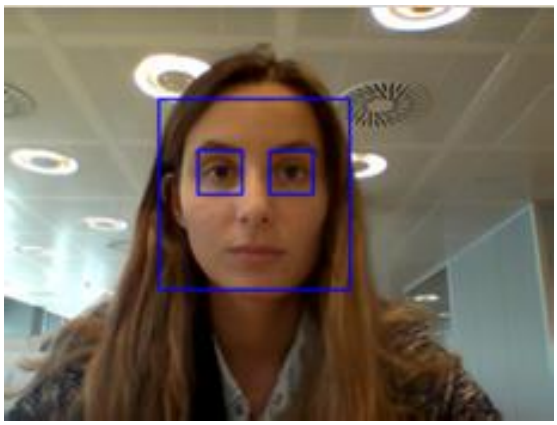

Figura 3-4: Ejemplo de detección de cara y ojos

3.3.3 Detección del centro del iris

Para la detección del centro del iris se probó además otro método de los mencionados en la sección 2.2.2. Optando por el que mejor resultados daba, que es el explicado al final del apartado 2.2.2.

La primera prueba fue con Circular Hough Transform; la imagen binaria del ojo, con ecualización del histograma (para aumentar el contraste), pasa por un filtro *Canny* para localizar los contornos y entonces se utiliza la función HoughCircles para detectar círculos (ver Figura 3-5).

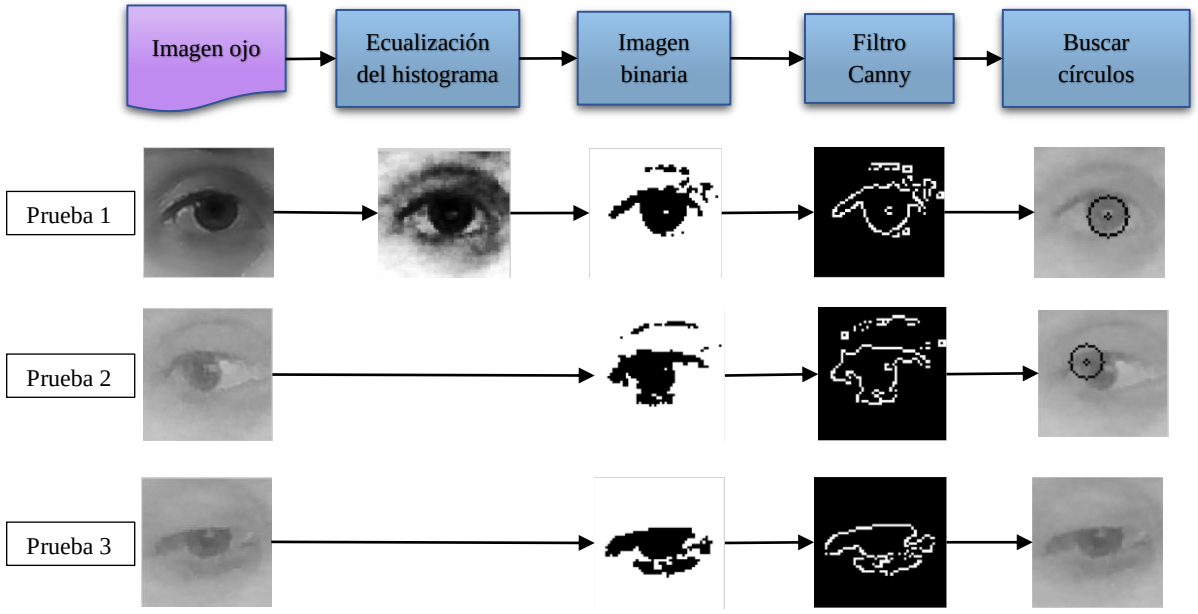

Figura 3-5: Diagrama y pruebas del método Circular Hough Transform

Como se observa en la Figura 3-5, este método funciona bien cuando el iris se encuentra en el centro del ojo, pero falla si se desplaza por la poca calidad de las imágenes. Cuando mira a la izquierda localiza un círculo un poco más arriba del iris y cuando mira hacia abajo ni siquiera detecta círculos.

Por esto se probó llevar a cabo la técnica basada en gradientes, anteriormente explicada en la sección 2.2.2, la cual detecta eficientemente el centro del iris en imágenes de baja resolución. La implementación de esta técnica consiste en primero redimensionar la imagen del ojo para disminuir el tamaño de píxeles y así ahorrar en tiempo computacional, seguido del cálculo de los gradientes en los ejes $\boldsymbol{x}$ e $\boldsymbol{y}$ junto a sus magnitudes usando el filtro *Sobel*, eliminando los píxeles de los gradientes donde la magnitud sea mayor que un umbral calculado a partir de su media y desviación típica. La imagen $\boldsymbol{w_c}$ se calcula invirtiendo la imagen con un filtro, se probó tanto con un filtro *GaussianBlur* como con una ecualización del histograma, siendo el resultado el mismo en ambos casos. Finalmente se lleva a cabo la matriz de suma de posibles centros de la Ecuación (4), siendo el mayor el centro buscado (Figura 3-6).

En la Figura 3-7 se pueden observar ejemplos del cálculo del centro del iris con este método, mirando a distintos sitios y con distinta luz. Ya que esta técnica funciona adecuadamente para las circunstancias de este trabajo, se decidió el uso de ella.

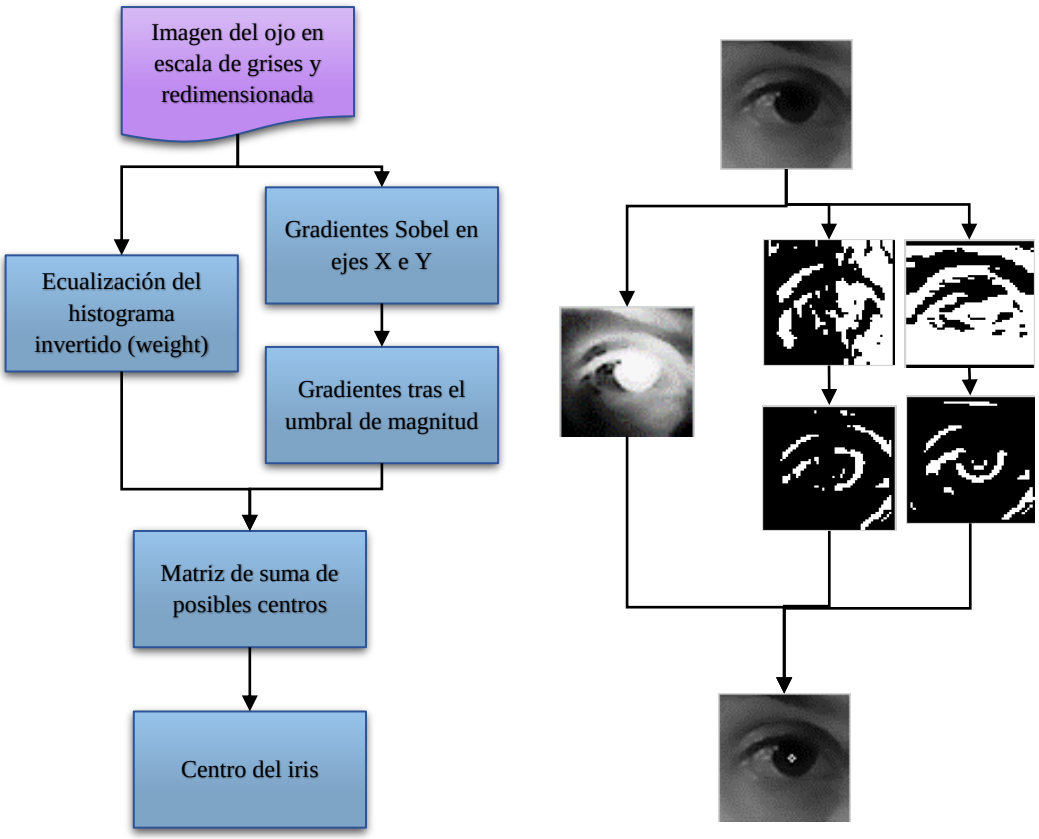

Figura 3-6: Diagrama del método del gradiente

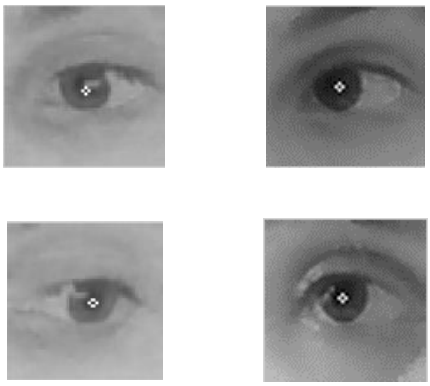

Figura 3-7: Ejemplos del funcionamiento del método del gradiente

3.3.4 Detección de los puntos característicos del contorno del ojo

La detección de los extremos del ojo con precisión es algo complicado, al igual que con el centro del iris, antes de dar con el algoritmo utilizado y explicado en el apartado 2.2.3, se probaron distintas técnicas. La primera fue *Harris Corner Detector*, para implementar esta técnica se utiliza la imagen binaria normalizada y el filtro *Canny* para obtener los bordes, entonces se usa la función *cornerHarris*, la cual devuelve todas las esquinas que encuentre. De estas esquinas hay que buscar la que se encuentre en la zona de interés, es decir, las comisuras de los ojos. Un inconveniente de este método es que varía mucho según la luz y las sombras, y puede encontrar muchas más esquinas de las deseadas cerca de la zona o incluso no encontrar ninguna. En la Figura 3-8 se puede observar distintas pruebas con los dos ojos, la primera, tercera y cuarta con el izquierdo y la segunda con el derecho, con variación de luz y de sombras. Las últimas imágenes de cada prueba muestran todas las esquinas encontradas por la función *cornerHarris*, se puede observar en las pruebas 1 y 2 como no llega a detectar ninguna esquina en la comisura interna, mientras que detecta demasiadas esquinas en la comisura externa.

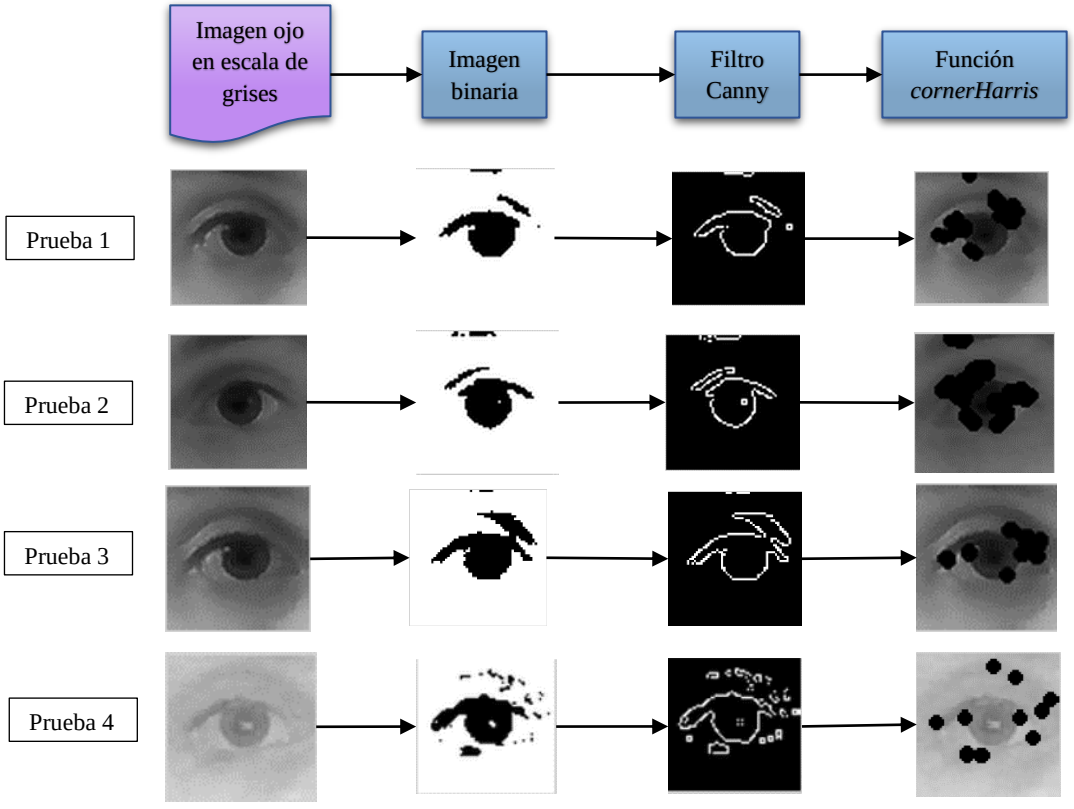

Figura 3-8: Diagrama y pruebas de Harris Corner Detector

El segundo método fue el de *Shi-Tomasi corner detector*, versión mejorada de *Harris Corner Detector*, el cual da mejores resultados. La función que lo implementa se llama *goodFeaturesToTrack*, se le pasa como parámetros la imagen en escala de grises, número de esquinas para detectar, el nivel de calidad y la mínima distancia entre esquinas. Como se puede comprobar en la Figura 3-9, si se le envía la imagen del ojo entero con un máximo de

10 esquinas a detectar, entre ellas se encuentran las comisuras buscadas. Para acotar la búsqueda, se le envían las regiones dónde posiblemente se encuentren las comisuras de la imagen del ojo y un máximo de esquinas a detectar, de esta manera suele tener bastantes aciertos, pero aún queda algún caso donde da prioridad a otra parte de la imagen antes de la comisura buscada, como vemos en la prueba 3 de la Figura 3-9.

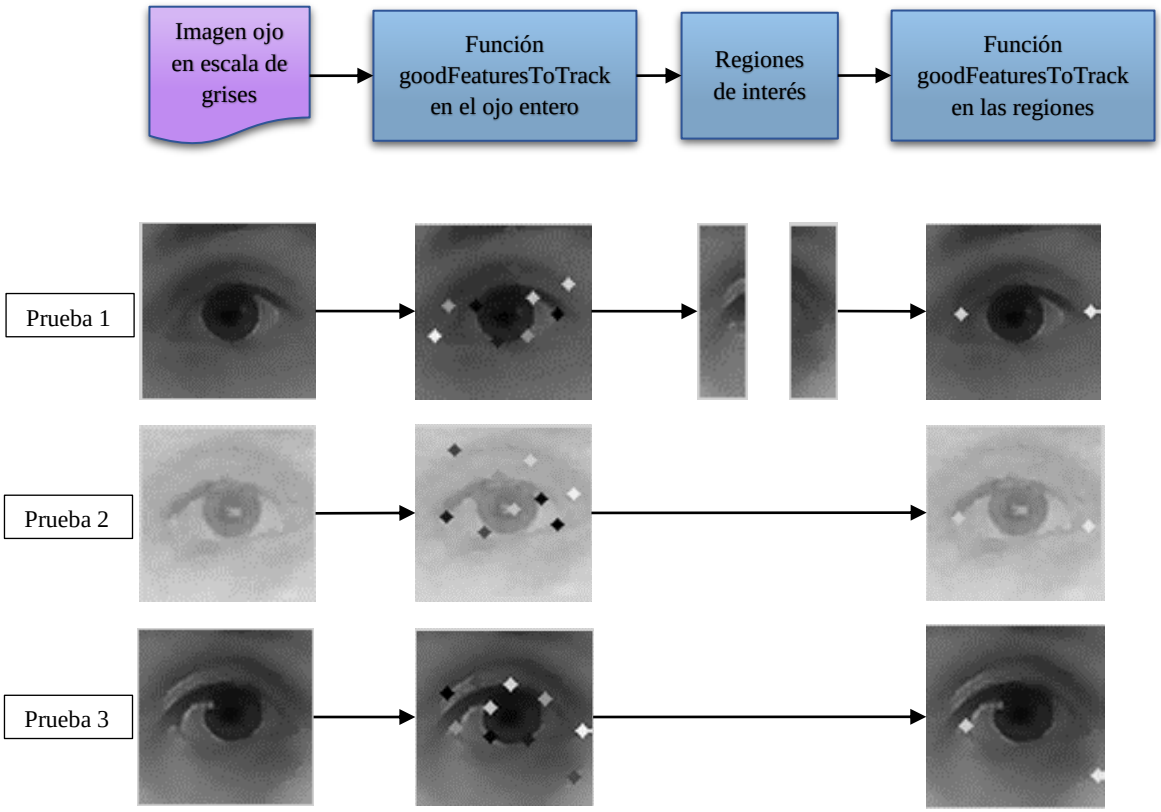

Figura 3-9: Diagrama y pruebas de Shi-Tomasi corner detector

Como conclusión se optó por un método bastante más robusto, la librería DLIB ya explicada, la cual no solo detecta las comisuras de los ojos con precisión, sino que además localiza el contorno del ojo. Esta librería hace uso de un modelo entrenado para esta tarea, el modelo usado será "*shape_predictor_68_face_landmarks.dat* ", el cual puede ser encontrado en la página de DLIB. Una vez entrenado es necesario pasarle la cara en la que tiene que detectar los *landmarks*, y el número del *landmarks* deseado, los cuáles se comentaron en el apartado 2.2.2. En la Figura 3-10 se observan los resultados que se obtienen con este método, bastante preciso.

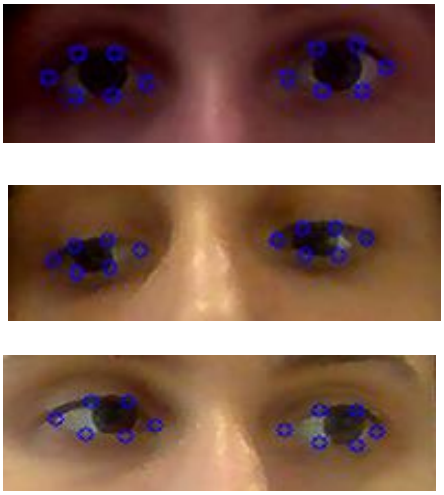

Figura 3-10: Ejemplo del funcionamiento de la librería DLIB

3.3.5 Calcular vectores v_N y v_P

Una vez obtenidos todos los puntos hacen uso de las Ecuaciones (2), (3) y (4) donde se relacionan el centro del iris con ambas comisuras de cada ojo, obteniendo así el vector $\boldsymbol{v_N}$. De igual manera se usan los dos puntos del párpado superior y los dos del inferior para crear dos vectores en cada ojo, siendo la media $\boldsymbol{v_P}$. En la etapa de calibración se guardarán estos vectores junto con el punto en la pantalla al que hacen referencia, para posteriormente resolver las Ecuaciones (1) y (12).

3.3.6 Resolver el sistema

Una vez acabada la etapa de calibración se usarán 13 vectores $\boldsymbol{v_N}$ y 13 vectores $\boldsymbol{v_P}$ para resolver ambos sistemas. Es necesario separar la Ecuación (1) en dos subsistemas, uno para ***POx*** (El punto de observación en el eje horizontal) y otro para ***POy*** (El punto de observación en el eje vertical), quedando las siguientes ecuaciones, del mismo tipo que la Ecuación (9):

$$\boldsymbol{POx = a_1 + a_2 * v_{Nx} + a_3 * v_{Ny} + a_4 * v_{Nx} * v_{Ny} + a_5 * v_{Nx}^2 + a_6 * v_{Ny}^2,} \quad (14)$$

$$\boldsymbol{POy = b_1 + b_2 * v_{Nx} + b_3 * v_{Ny} + b_4 * v_{Nx} * v_{Ny} + b_5 * v_{Nx}^2 + b_6 * v_{Ny}^2,} \quad (15)$$

siendo los coeficientes $\boldsymbol{a_i}$ y $\boldsymbol{b_i}$ los pertenecientes a la matriz ***C*** y que se deben obtener.
A partir de la técnica de los mínimos cuadrados y las Ecuaciones (10) y (11) quedarían los siguientes sistemas a resolver:

$$\begin{bmatrix} \sum \mathbf{1} & \sum v_{Nx} & \sum v_{Ny} & \sum v_{Nx}v_{Ny} & \sum v_{Nx}^2 & \sum v_{Ny}^2 \\ \sum v_{Nx} & \sum v_{Nx}^2 & \sum v_{Nx}v_{Ny} & \sum v_{Nx}^2 v_{Ny} & \sum v_{Nx}^3 & \sum v_{Nx}v_{Ny}^2 \\ \sum v_{Ny} & \sum v_{Nx}v_{Ny} & \sum v_{Ny}^2 & \sum v_{Nx}v_{Ny}^2 & \sum v_{Nx}^2 v_{Ny} & \sum v_{Ny}^3 \\ \sum v_{Nx}v_{Ny} & \sum v_{Nx}^2 v_{Ny} & \sum v_{Nx}v_{Ny}^2 & \sum v_{Nx}^2 v_{Ny}^2 & \sum v_{Nx}^3 v_{Ny} & \sum v_{Nx}v_{Ny}^3 \\ \sum v_{Nx}^2 & \sum v_{Nx}^3 & \sum v_{Nx}^2 v_{Ny} & \sum v_{Nx}^3 v_{Ny} & \sum v_{Nx}^4 & \sum v_{Nx}^2 v_{Ny}^2 \\ \sum v_{Ny}^2 & \sum v_{Nx}v_{Ny}^2 & \sum v_{Ny}^3 & \sum v_{Nx}v_{Ny}^3 & \sum v_{Nx}^2 v_{Ny}^2 & \sum v_{Ny}^4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} \sum \boldsymbol{POx} \\ \sum \boldsymbol{POx}\, v_{Nx} \\ \sum \boldsymbol{POx}\, v_{Ny} \\ \sum \boldsymbol{POx}\, v_{Ny}v_{Ny} \\ \sum \boldsymbol{POx}\, v_{Nx}^2 \\ \sum \boldsymbol{POx}\ v_{Ny}^2 \end{bmatrix} \quad (16)$$

$$\begin{bmatrix} \sum \mathbf{1} & \sum v_{Nx} & \sum v_{Ny} & \sum v_{Nx}v_{Ny} & \sum v_{Nx}^2 & \sum v_{Ny}^2 \\ \sum v_{Nx} & \sum v_{Nx}^2 & \sum v_{Nx}v_{Ny} & \sum v_{Nx}^2 v_{Ny} & \sum v_{Nx}^3 & \sum v_{Nx}v_{Ny}^2 \\ \sum v_{Ny} & \sum v_{Nx}v_{Ny} & \sum v_{Ny}^2 & \sum v_{Nx}v_{Ny}^2 & \sum v_{Nx}^2 v_{Ny} & \sum v_{Ny}^3 \\ \sum v_{Nx}v_{Ny} & \sum v_{Nx}^2 v_{Ny} & \sum v_{Nx}v_{Ny}^2 & \sum v_{Nx}^2 v_{Ny}^2 & \sum v_{Nx}^3 v_{Ny} & \sum v_{Nx}v_{Ny}^3 \\ \sum v_{Nx}^2 & \sum v_{Nx}^3 & \sum v_{Nx}^2 v_{Ny} & \sum v_{Nx}^3 v_{Ny} & \sum v_{Nx}^4 & \sum v_{Nx}^2 v_{Ny}^2 \\ \sum v_{Ny}^2 & \sum v_{Nx}v_{Ny}^2 & \sum v_{Ny}^3 & \sum v_{Nx}v_{Ny}^3 & \sum v_{Nx}^2 v_{Ny}^2 & \sum v_{Ny}^4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} = \begin{bmatrix} \sum \boldsymbol{POy} \\ \sum \boldsymbol{POy}\, v_{Nx} \\ \sum \boldsymbol{POy}\, v_{Ny} \\ \sum \boldsymbol{POy}\, v_{Ny}v_{Ny} \\ \sum \boldsymbol{POy}\, v_{Nx}^2 \\ \sum \boldsymbol{POy}\ v_{Ny}^2 \end{bmatrix} \quad (17)$$

De la misma forma se resolverá la Ecuación (13), obteniendo dos conjuntos de coeficientes, $\boldsymbol{b_i}$ y $\boldsymbol{c_i}$ para resolver el punto de observación en el eje vertical.

$$\boldsymbol{POy'} = \boldsymbol{c_1} + \boldsymbol{c_2} * \boldsymbol{v_{Px}} + \boldsymbol{c_3} * \boldsymbol{v_{Py}} + \boldsymbol{c_4} * \boldsymbol{v_{Px}} * \boldsymbol{v_{Py}} + \boldsymbol{c_5} * \boldsymbol{v_{Px}^2} + \boldsymbol{c_6} * \boldsymbol{v_{Py}^2}, \quad (18)$$

$$\begin{bmatrix} \sum \mathbf{1} & \sum v_{Px} & \sum v_{Py} & \sum v_{Px}v_{Py} & \sum v_{Px}^2 & \sum v_{Py}^2 \\ \sum v_{Px} & \sum v_{Px}^2 & \sum v_{Px}v_{Py} & \sum v_{Px}^2 v_{Py} & \sum v_{Px}^3 & \sum v_{Px}v_{Py}^2 \\ \sum v_{Py} & \sum v_{Px}v_{Py} & \sum v_{Py}^2 & \sum v_{Px}v_{Py}^2 & \sum v_{Px}^2 v_{Py} & \sum v_{Py}^3 \\ \sum v_{Px}v_{Py} & \sum v_{Px}^2 v_{Py} & \sum v_{Px}v_{Py}^2 & \sum v_{Px}^2 v_{Py}^2 & \sum v_{Px}^3 v_{Py} & \sum v_{Px}v_{Py}^3 \\ \sum v_{Px}^2 & \sum v_{Px}^3 & \sum v_{Px}^2 v_{Py} & \sum v_{Px}^3 v_{Py} & \sum v_{Px}^4 & \sum v_{Px}^2 v_{Py}^2 \\ \sum v_{Py}^2 & \sum v_{Px}v_{Py}^2 & \sum v_{Py}^3 & \sum v_{Px}v_{Py}^3 & \sum v_{Px}^2 v_{Py}^2 & \sum v_{Py}^4 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} \sum \boldsymbol{POy'} \\ \sum \boldsymbol{POy'}\, v_{Px} \\ \sum \boldsymbol{POy}\, v_{Py} \\ \sum \boldsymbol{POy'}\, v_{Py}v_{Py} \\ \sum \boldsymbol{POy'}\, v_{Px}^2 \\ \sum \boldsymbol{POy'}\ v_{Py}^2 \end{bmatrix} \quad (19)$$

En la calibración, ***POy*** será igual a ***POy'*** ya que serán el punto de observación del eje vertical donde deberá mirar el usuario. A la hora de tener que detectar el punto de observación después de la calibración será cuando varíen y se comprobará cuál de los dos es más preciso.

3.3.7 Calibración

En esta primera etapa se obtendrán los vectores necesarios para resolver los sistemas. Se usarán 13 coordenadas de la pantalla, que serán los puntos de observación ***POx*** y ***POy.*** El usuario deberá mirarlos uno por uno y se calcularán los vectores de características $\boldsymbol{v_N}$ y $\boldsymbol{v_P}$ correspondientes.
Estos trece puntos serán las 4 esquinas de la pantalla, los puntos medios de cada extremo de la pantalla, el centro y 4 puntos en las diagonales (ver Figura 3-11).

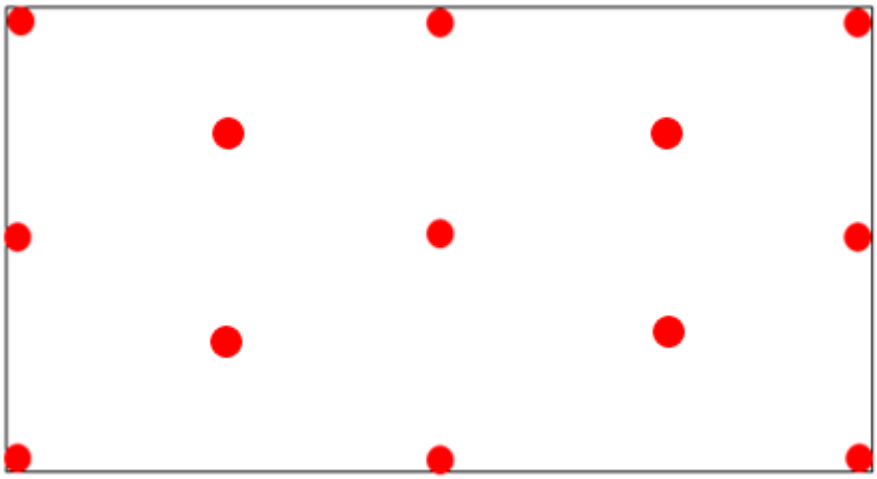

Figura 3-11: Los trece puntos de la calibración

Una vez que el usuario haya mirado a todas las coordenadas y se obtengan todos los vectores se resolverán los sistemas (15), (16) y (18). Para esto se crearán las matrices necesarias sumando los datos de los 13 casos. Por último, habrá que resolver los sistemas, para lo que se usará el método de la matriz invertida:

$$\boldsymbol{A} * \boldsymbol{x} = \boldsymbol{b} \quad (20)$$

$$\boldsymbol{x} = \boldsymbol{A^{-1}} * \boldsymbol{b} \quad (21)$$

Obtenido $\boldsymbol{x}$ de los distintos sistemas ya se tienen los coeficientes $\boldsymbol{a_i}$ $\boldsymbol{b_i}$ y $\boldsymbol{c_i}$, usados para la siguiente etapa.

3.3.8 Pruebas

En esta etapa se pide al usuario que mire a distintos puntos de la pantalla, y se usarán los coeficientes de antes junto con las ecuaciones para estimar los puntos de observación.
Se comprobará que método es más preciso para el eje vertical y en qué medida se cumplen los requisitos previos del apartado 3.2.

3.3.9 Resultados

Para representar los resultados se pintarán en la pantalla los lugares estimados como puntos de observación y se calcularán los errores de exactitud cometidos.

3.4 Compensar el movimiento de la cabeza

En un principio se planteó implementar una técnica que compensara el movimiento de la cabeza del usuario para que este pudiera moverse con libertad y naturalidad. Aunque finalmente no se ha incluido en el sistema final por problemas que surgieron con el planteamiento del método, se describe en este apartado la problemática y una posible solución.

Como ya se detalló en el capítulo 2, OpenCV dispone de una función para el cálculo de la matriz necesitada, *findHomography*, la cual necesita como parámetros unos puntos de referencia de las dos imágenes, la del objeto a buscar y la de la escena donde buscar el objeto. Para obtener los mejores puntos de referencia de un objeto al que buscar, se puede hacer uso de las técnicas SURF o SIFT y de FLANN o *Brute-Force Matcher*, siendo los elegidos SURF y FLANN (ver capítulo 2.2.6). En la Figura 3-12 se muestra una prueba del funcionamiento.

Figura 3-12: Puntos de referencia mediante SURF y matching mediante FLANN (izquierda). Imagen transformada con homografía (derecha).

En un principio se utilizó solo la homografía entre la imagen de referencia y la nueva imagen con el usuario desplazado, pero tras realizar diferentes pruebas, se llegó a la conclusión que había algo que no cuadraba.

Si suponemos un usuario mirando a un punto *x*, cuando el usuario se desplaza, pero sigue mirando al mismo punto, el iris se desplaza también (ver Figura 3-13). Si realizamos la homografía entre estas dos imágenes y calculamos el punto de observación en esta nueva posición, el punto *x'* resultante se encontrará en una pantalla desplazada a la original, ver Figura 3-14.

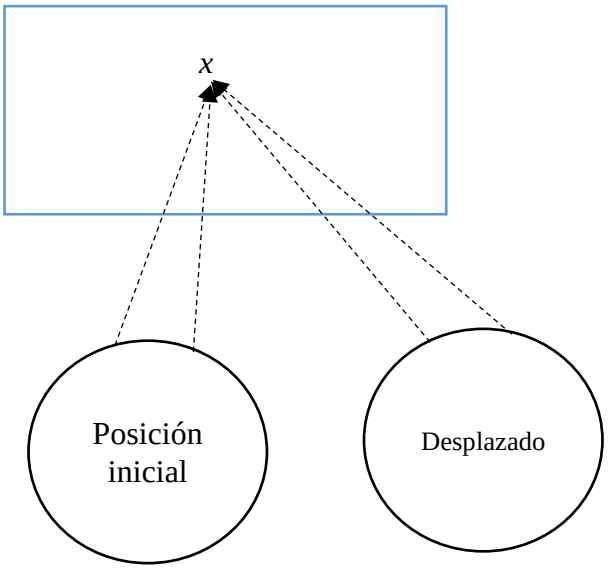

Figura 3-13: Posición inicial y posición desplazada de un usuario mirando a un mismo punto en una pantalla.

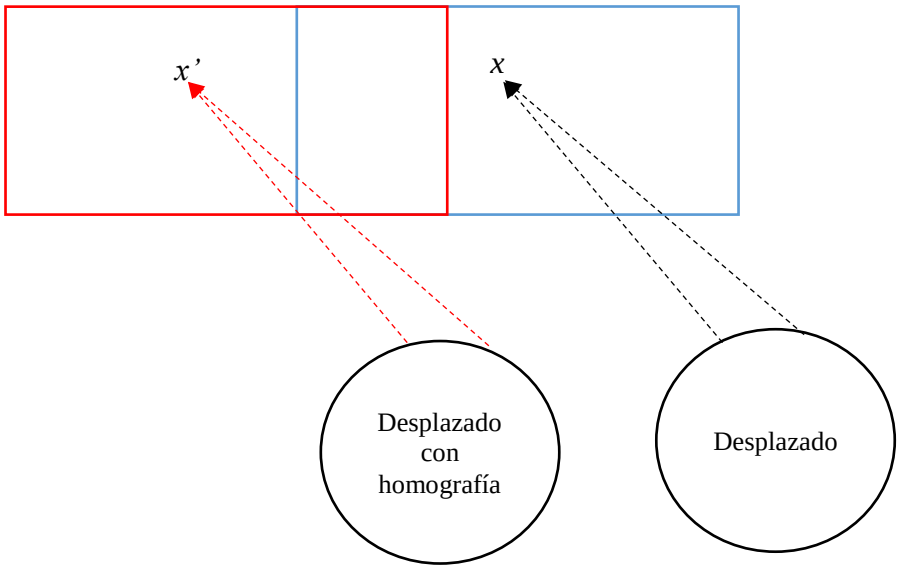

Figura 3-14: Comparación entre el punto donde mira un usuario desplazado y donde se estimaría mediante la homografía.

Debido a esto será necesario otra homografía para relacionar la pantalla con la posición inicial del usuario y cuando éste se desplace, calcular el punto estimado haciendo uso de ambas homografías. Esta nueva implementación no se ha podido llevar a cabo debido a la limitación de tiempo de este trabajo, pero se propondrá como trabajo futuro para evaluar su efectividad.

4 Pruebas y resultados

4.1 Introducción

En este apartado se comentarán las pruebas realizadas para el funcionamiento del programa, así como comprobar en qué situaciones se obtienen mejores resultados y en cuales no funciona, si cumple los requisitos necesarios y cuál es la precisión que se puede obtener.

Para la obtención de los resultados se ha utilizado un equipo con un procesador Intel® Core™ i7-4510U con una CPU a 2.0 GHz y 2GB de RAM. El sistema operativo es Windows 10 de 64 bits y la resolución de la pantalla es de 1536x864 pixeles. El lenguaje utilizado es C++ con las librerías OpenCV versión 3.4.1 y DLIB versión 19.10, implementado en Visual Studio.

4.2 Descripción del dataset

Primero se describirán los datos usados para la evaluación, aunque no se hará uso de una base de datos, sino de imágenes de voluntarios que ejecutarán el programa en tiempo real. Realizarán primero la etapa de calibración y seguidamente la etapa de pruebas.

Para la realización de las pruebas planteadas a continuación se ha hecho uso de 5 voluntarios distintos, los cuales pondrán a prueba el programa. El programa deberá funcionar correctamente para los 5 voluntarios cumpliendo así el requisito que dice que debe funcionar con distintos usuarios.

4.3 Calibración

Para cada uno de los voluntarios lo primero será que pasen la etapa de calibración, se pedirá que en esta etapa dejen la cabeza lo más quieta posible y miren a los puntos que vayan apareciendo. Como ya se comentó habrá un total de 13 coordenadas de la pantalla a la que tendrán que mirar. El programa pedirá al usuario que tras la aparición de un punto lo mire y entonces pulse la tecla *enter* y espere unos segundos, y así sucesivamente con cada punto.

4.4 Evaluación

Se realizarán 3 tipos de pruebas, las cuales consistirán en mirar a distintas coordenadas de la pantalla o fuera de ella y que el programa estime el punto de observación del usuario a partir de los datos de calibración. Cada evaluación completa contando con la explicación al voluntario ha tenido una duración estimada de 20 minutos. Además, se evaluarán los dos métodos usados para el cálculo de la coordenada en el eje vertical. Comprobando qué método devuelve mejores resultados.

4.4.1 Primera prueba

En esta primera prueba se evaluará el requisito de que el programa debe conocer si un usuario mira a la pantalla o fuera de ella. Para ello se utilizan las coordenadas del punto de observación, si están fuera de las dimensiones de la pantalla, considerando un intervalo de error, se indicará como que el usuario no está mirando a la pantalla.

Se pedirá en 4 ocasiones al usuario que mire afuera de la pantalla o a cualquier lugar dentro de la pantalla, el programa debe indicar en cada ocasión que hizo el usuario. Se realizará el cálculo del eje vertical de la coordenada del punto de observación por los dos métodos propuestos en este trabajo.

- Primer voluntario: El proceso que siguió el primer voluntario fue mirar fuera, dentro, dentro y fuera de la pantalla. Se puede observar en la Figura 4-1 la detección de los 2 puntos dentro de la pantalla, siendo el negro el primer método y el azul el segundo, mientras que los otros no aparecen por estar fuera de la pantalla. Los resultados del programa son los de la Tabla 4-1, coincidiendo con la realización del usuario.

Figura 4-1: Estimación de los puntos de observación en la prueba 1 del primer voluntario.

Tabla 4-1: Resultados del proceso del primer voluntario (en verde si es correcto y en rojo si es incorrecto).

	Primer caso	Segundo caso	Tercer caso	Cuarto caso
Método 1	FUERA	DENTRO	DENTRO	FUERA
Método 2	FUERA	DENTRO	DENTRO	FUERA

- Segundo voluntario: En este segundo caso, las pautas del usuario fueron mirar dentro, fuera, dentro y fuera de la pantalla. En la Figura 4-2 y la Tabla 4-2 se encuentran los resultados del programa, en este caso se produce un error de estimación con el primer método que provoca que se estime como dentro, aunque al límite, un punto que debería estar fuera.

Figura 4-2: Estimación de los puntos de observación en la prueba 1 del segundo voluntario.

Tabla 4-2: Resultados del proceso del segundo voluntario (en verde si es correcto y en rojo si es incorrecto).

	Primer caso	Segundo caso	Tercer caso	Cuarto caso
Método 1	DENTRO	DENTRO	DENTRO	FUERA
Método 2	DENTRO	FUERA	DENTRO	FUERA

- Tercer voluntario: El procedimiento del tercer voluntario fue mirar fuera, dentro, fuera y dentro de la pantalla. También coincide con los resultados del programa como se observa en la Figura 4-3 y la Tabla 4-3.

Figura 4-3: Estimación de los puntos de observación en la prueba 1 del tercer voluntario.

Tabla 4-3: Resultados del proceso del tercer voluntario (en verde si es correcto y en rojo si es incorrecto).

	Primer caso	Segundo caso	Tercer caso	Cuarto caso
Método 1	FUERA	DENTRO	FUERA	DENTRO
Método 2	FUERA	DENTRO	FUERA	DENTRO

- Cuarto voluntario: El procedimiento del cuarto voluntario fue mirar dentro, dentro, fuera y fuera de la pantalla. En este caso el primer método falla en la estimación de los puntos externos, ver Figura 4-4 y Tabla 4-4.

Figura 4-4: Estimación de los puntos de observación en la prueba 1 del cuarto voluntario.

Tabla 4-4: Resultados del proceso del cuarto voluntario (en verde si es correcto y en rojo si es incorrecto).

	Primer caso	Segundo caso	Tercer caso	Cuarto caso
Método 1	DENTRO	DENTRO	DENTRO	DENTRO
Método 2	DENTRO	DENTRO	FUERA	FUERA

- Quinto voluntario: El procedimiento del quinto voluntario fue mirar fuera, fuera, fuera y dentro de la pantalla. En este caso se cumplen las estimaciones, ver Figura 4-5 y Tabla 4-5.

Figura 4-5: Estimación de los puntos de observación en la prueba 1 del quinto voluntario.

Tabla 4-5: Resultados del proceso del quinto voluntario (en verde si es correcto y en rojo si es incorrecto).

	Primer caso	Segundo caso	Tercer caso	Cuarto caso
Método 1	FUERA	FUERA	FUERA	DENTRO
Método 2	FUERA	FUERA	FUERA	DENTRO

En esta primera prueba se evaluaba el requisito en el que el programa pudiera distinguir cuando un usuario está mirando o no a la pantalla, siendo un requisito que se cumple en el 85% de los casos probados con el primer método y un 100% con el segundo. El fallo se produce cuando se mira al límite de la pantalla o fuera de ella, debido al error de precisión.

4.4.2 Segunda prueba

El siguiente requisito a evaluarse en esta segunda prueba será conocer en qué cuadrante de la pantalla se encuentra el punto de observación del usuario, ver Figura 4-6.

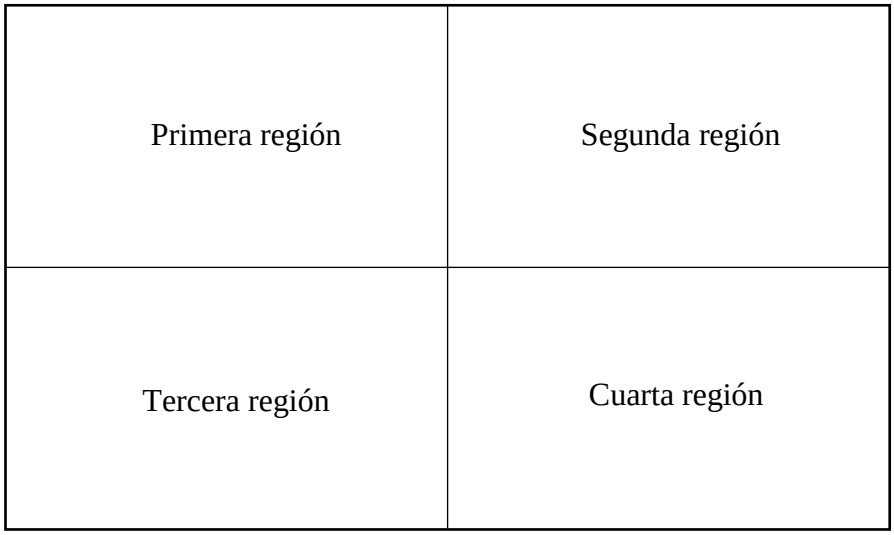

Figura 4-6: División de la pantalla en la segunda prueba.

La prueba consistirá en la aparición de distintos puntos en la pantalla a los que el usuario deberá mirar uno por uno al igual que en la calibración. Cuando finalice se comprobará cuantos son estimados correctamente en la región a la que pertenecían y cuantos no. Aparecerán 16 puntos, 4 por región, siendo el orden de aparición, primero horizontal y luego vertical, es decir, primero los 4 de arriba, siendo el primero el más a la izquierda.

- Primer voluntario: Los resultados del primer usuario utilizando ambos métodos para la coordenada del eje vertical son los siguientes:

Tabla 4-6 : Estimación correcta o incorrecta de cada punto en el cuartil correspondiente del primer voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	%
1	✓	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	87.5
2	✓	✓	✓	✓	×	✓	×	×	×	×	✓	✓	✓	✓	✓	✓	68.75

En las Figura 4-7 y Figura 4-8 encontramos los puntos rojos que son a donde debe mirar el usuario y los puntos negros o azules que son los estimados por el programa.

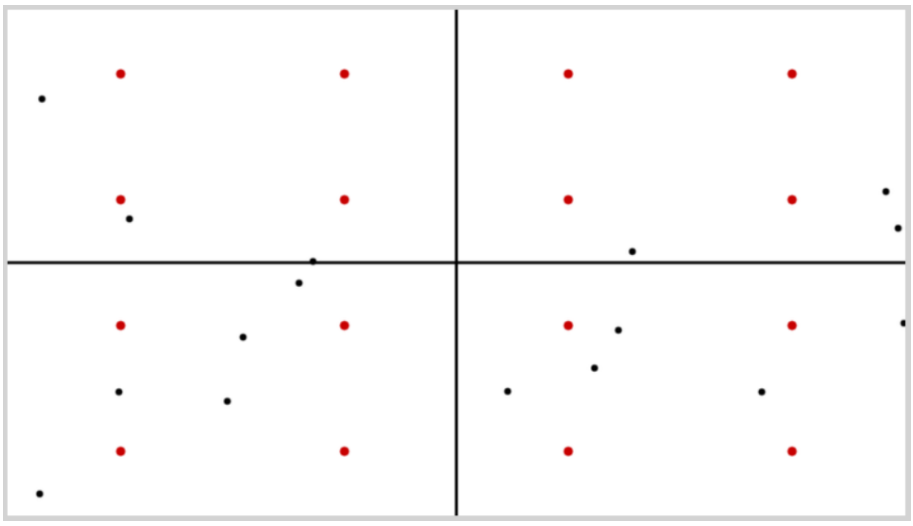

Figura 4-7: Estimación de los puntos de observación mediante el primer método en la prueba 2 del primer voluntario.

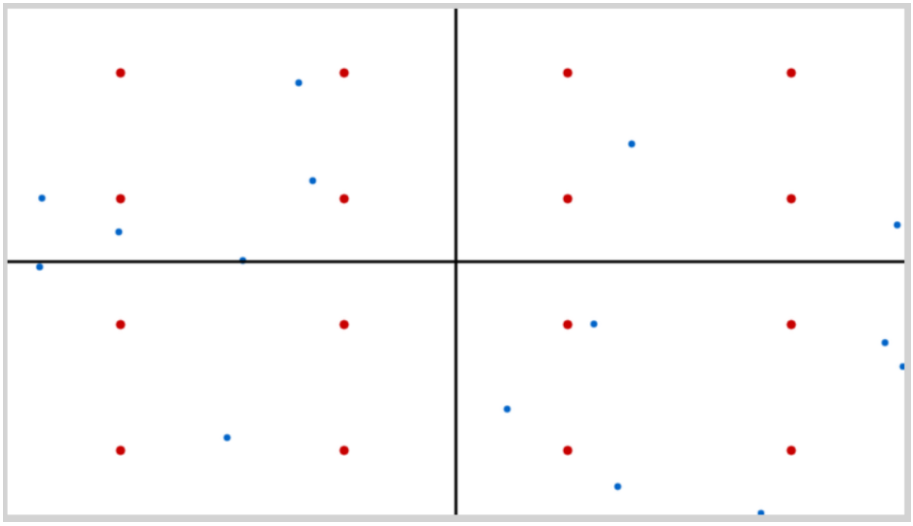

Figura 4-8: Estimación de los puntos de observación mediante el segundo método en la prueba 2 del primer voluntario.

- Segundo voluntario: Los resultados del segundo caso se encuentras en la Tabla 4-7, Figura 4-9 y Figura 4-10.

Tabla 4-7: Estimación correcta o incorrecta de cada punto en el cuartil correspondiente del segundo voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	%
1	✓	✓	✓	×	×	✓	×	×	✓	✓	✓	×	✓	✓	×	✓	62.50
2	×	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×	×	✓	37.5

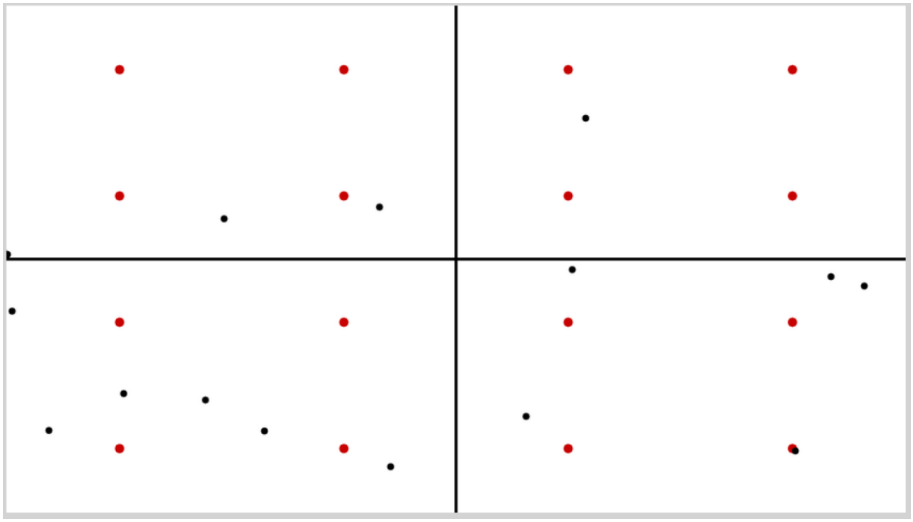

Figura 4-9: Estimación de los puntos de observación mediante el primer método en la prueba 2 del segundo voluntario.

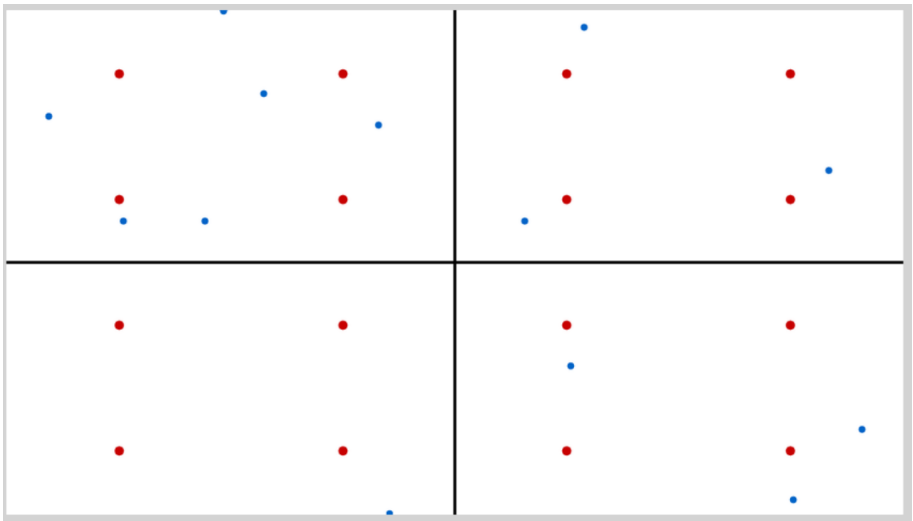

Figura 4-10: Estimación de los puntos de observación mediante el segundo método en la prueba 2 del segundo voluntario.

- Tercer voluntario: Los resultados del tercer caso se encuentras en la Tabla 4-8, Figura 4-11 y Figura 4-12.

Tabla 4-8: Estimación correcta o incorrecta de cada punto en el cuartil correspondiente del tercer voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	%
1	✓	✓	✓	×	×	×	×	✓	✓	×	✓	✓	×	✓	×	✓	56.25
2	×	✓	✓	✓	✓	×	×	×	✓	×	✓	✓	✓	✓	✓	✓	68.75

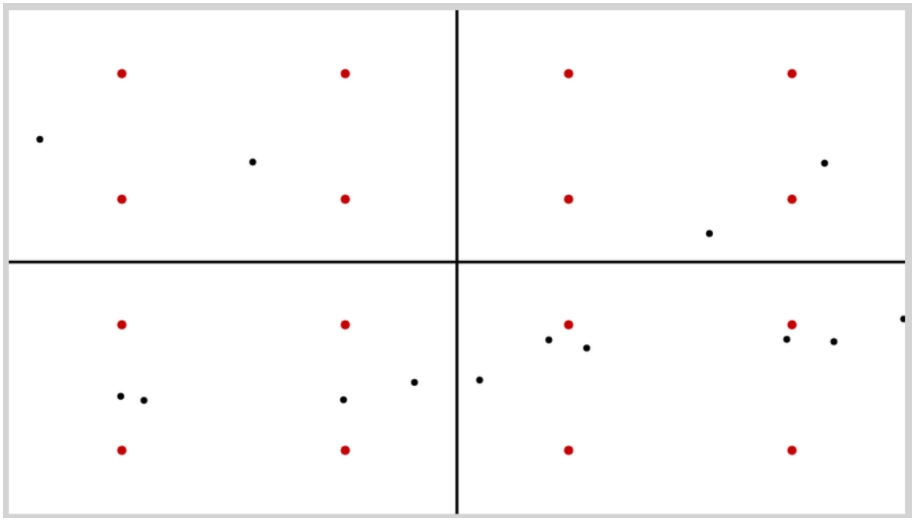

Figura 4-11: Estimación de los puntos de observación mediante el primer método en la prueba 2 del tercer voluntario.

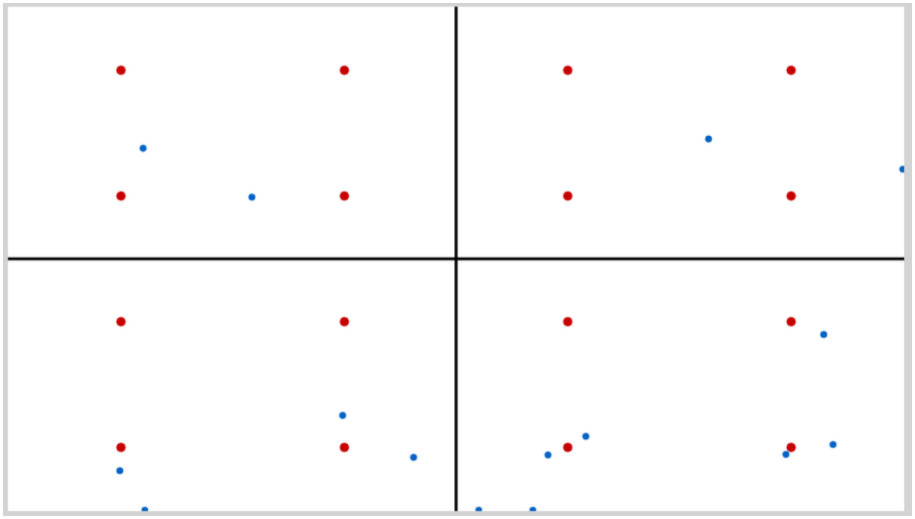

Figura 4-12: Estimación de los puntos de observación mediante el segundo método en la prueba 2 del tercer voluntario.

- Cuarto voluntario: Los resultados del cuarto caso se encuentras en la Tabla 4-9, Figura 4-13 y Figura 4-14.

Tabla 4-9: Estimación correcta o incorrecta de cada punto en el cuartil correspondiente del cuarto voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	%
1	×	✓	×	✓	×	×	✓	×	✓	✓	✓	✓	×	×	✓	✓	62.5
2	×	✓	✓	✓	×	×	×	×	✓	✓	✓	✓	×	✓	✓	✓	68.75

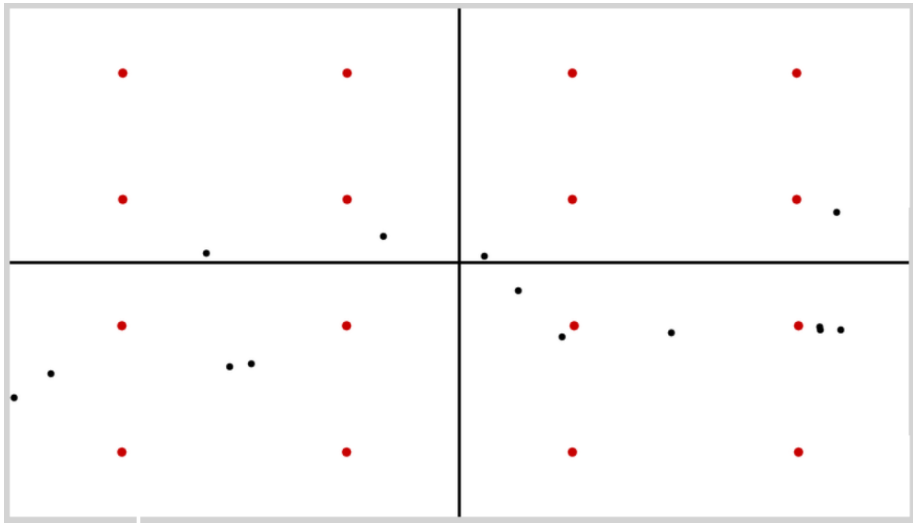

Figura 4-13: Estimación de los puntos de observación mediante el primer método en la prueba 2 del cuarto voluntario.

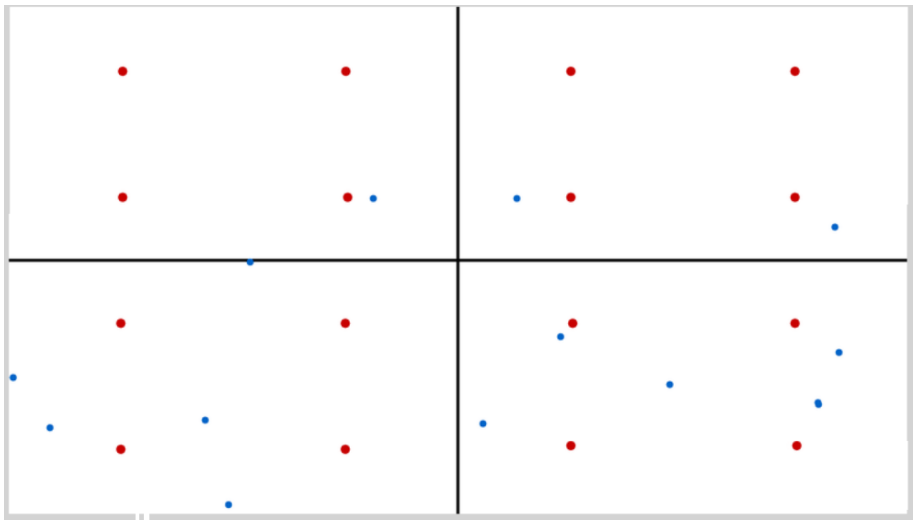

Figura 4-14: Estimación de los puntos de observación mediante el segundo método en la prueba 2 del cuarto voluntario.

- Quinto voluntario: Los resultados del último caso se encuentras en la Tabla 4-10, Figura 4-15 y Figura 4-16.

Tabla 4-10: Estimación correcta o incorrecta de cada punto en el cuartil correspondiente del cuarto voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	%
1	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	93.75
2	×	✓	✓	✓	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓	81.25

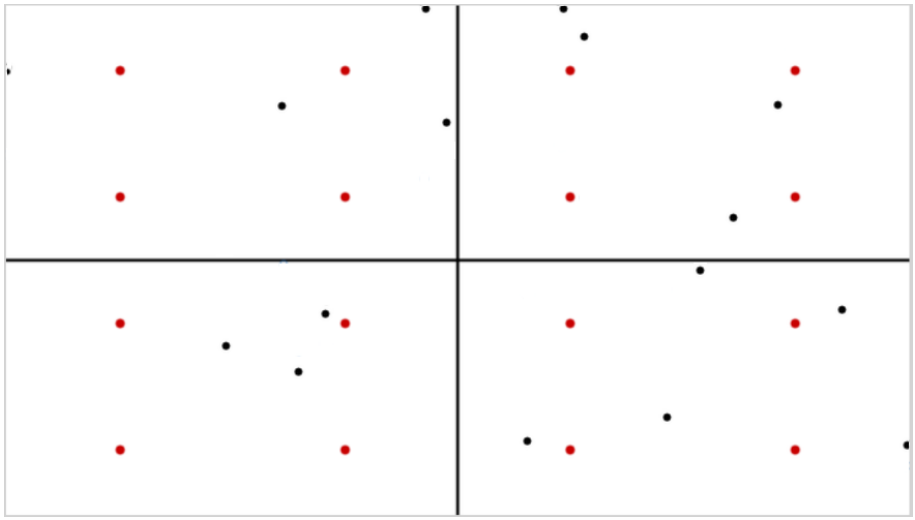

Figura 4-15: Estimación de los puntos de observación mediante el primer método en la prueba 2 del quinto voluntario.

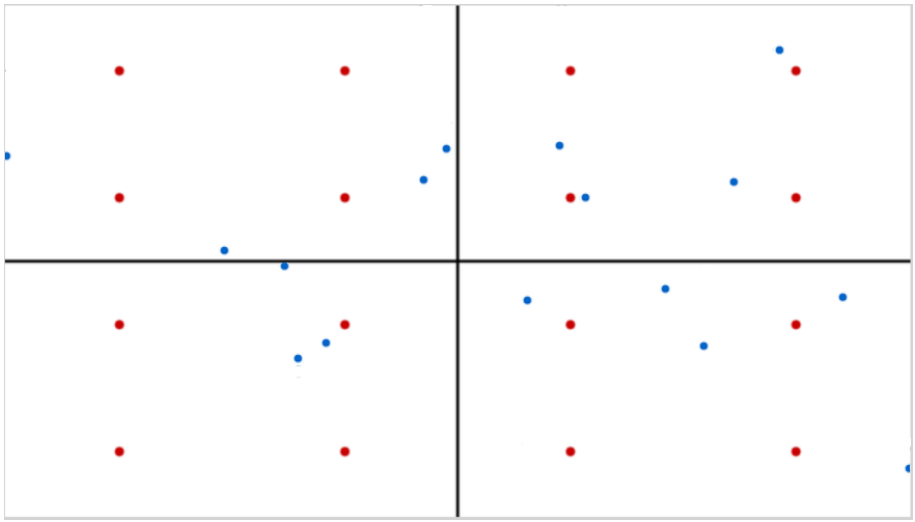

Figura 4-16: Estimación de los puntos de observación mediante el segundo método en la prueba 2 del quinto voluntario.

En esta segunda prueba se evaluaba el requisito de poder distinguir a que cuadrante de la pantalla mira un usuario se hace más distinción entre los dos métodos, en general, juntando los 5 voluntarios, se tiene un 71.25% de aciertos en la estimación del punto de observación con el primer método y un 65% con el segundo método. Las Tabla 4-11 y Tabla 4-12 son resúmenes de los aciertos de cada voluntario con los dos métodos.

Tabla 4-11: Número de aciertos de cada voluntario en cada región con el primer método (sobre 4)

	Primera región	Segunda región	Tercera región	Cuarta región	Total
Voluntario 1	3	3	4	4	14
Voluntario 2	3	1	4	2	10
Voluntario 3	2	2	2	3	9
Voluntario 4	1	2	3	4	9
Voluntario 5	4	4	2	4	15
Total					57
Media	2,6	2,4	3	3,4	
Desviación	1,14	1,14	1	0,89	

Tabla 4-12: Número de aciertos de cada voluntario en cada región con el segundo método (sobre 4)

	Primera región	Segunda región	Tercera región	Cuarta región	Total
Voluntario 1	3	2	2	4	11
Voluntario 2	3	2	0	1	6
Voluntario 3	2	2	3	4	11
Voluntario 4	1	2	3	4	10
Voluntario 5	3	4	2	4	13
Total					51
Media	2,4	2,4	2	3,4	
Desviación	0,89	0,89	1,22	1,34	

En la Figura 4-17 se encuentra representado la media y variación de aciertos en cada cuarto, se observa que las regiones con menos aciertos para el primer método son los dos de arriba, el primero y el segundo, mientras que para el segundo método son los tres primeros, principalmente el tercero.

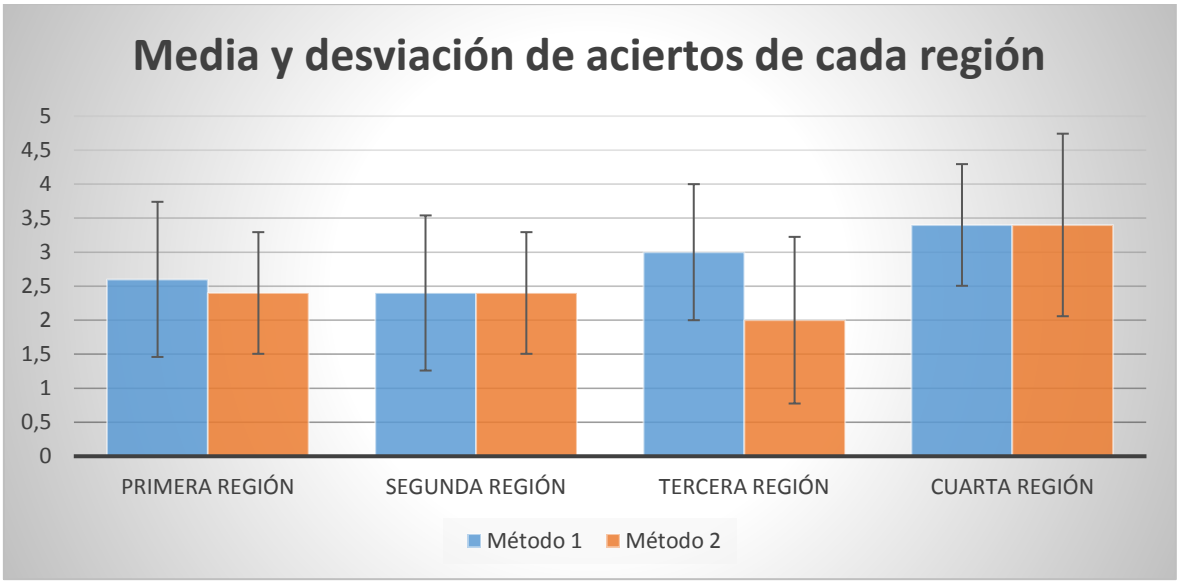

Figura 4-17: Media y desviación de aciertos en cada región

Las Tabla 4-13 y Tabla 4-14 contienen los porcentajes de en qué cuadrante son estimados los puntos de cada cuadrante. En estas representaciones se observan que la mayoría de las veces el error se comete de forma vertical, entre el primer y tercer cuadrante y entre el segundo y el cuarto.

Con el primer método se suelen estimar por debajo, es por eso por lo que los errores del primer cuadrante se encuentran en el tercero, los del segundo en el cuarto y los del tercero y cuarto fuera, aunque también hay casos entre el cuarto y tercer cuadrante de un error horizontal.

Con el segundo método parece no haber una relación tan clara como con el primero, en el primer cuadrante encontramos errores verticales tanto hacia arriba, los de fuera, como hacia abajo, los del tercer cuadrante. En el segundo cuadrante son hacia el cuarto, hacia abajo. En el tercero van principalmente hacia arriba, el primer cuadrante. Y en el cuarto hacia el segundo y fuera.

Tabla 4-13: Porcentajes de las estimaciones en cada cuadrante con el primer método.

Real / Estimación	Primera región	Segunda región	Tercera región	Cuarta región
Primera región	65	0	5	0
Segunda región	0	60	0	0
Tercera región	30	0	75	5
Cuarta región	0	40	10	85
Fuera	5	0	10	10
	100	100	100	100

Tabla 4-14: Porcentajes de las estimaciones en cada cuadrante con el segundo método.

Real / Estimación	Primera región	Segunda región	Tercera región	Cuarta región
Primera región	60	0	35	0
Segunda región	0	60	0	5
Tercera región	20	0	50	5
Cuarta región	0	40	10	85
Fuera	20	0	5	5
	100	100	100	100

4.4.2.1 Conclusiones

Al observar los distintos datos de esta prueba, queda claro que el eje horizontal cumple con el requisito, habiendo tenido lugar solo 3 errores en la estimación de la región, entre la tercera y cuarta, de todas las pruebas. Mientras que el eje vertical no lo cumple de la misma manera, aunque si con más de un 60% de probabilidad. Con el primer método se suelen estimar hacia abajo y con el segundo para ambos sentidos, aunque hay más casos hacia arriba.

4.4.3 Tercera prueba

En esta última prueba se comprobará en qué medida se cumple el último requisito: detectar en que parte de la pantalla se encuentra el punto de observación del usuario, dividendo la pantalla en 9 regiones (ver Figura 4-18).

Primera región	Segunda región	Tercera región
Cuarta región	Quinta región	Sexta región
Séptima región	Octava región	Novena región

Figura 4-18: División de la pantalla en la tercera prueba.

La prueba consistirá en la aparición de distintos puntos en la pantalla a los que el usuario deberá mirar uno por uno al igual que en la calibración y la prueba anterior, cuando acabe se comprobará cuantos son estimados correctamente en la región a la que pertenecían y cuantos no. Aparecerán 9 puntos, 1 por región, siendo el orden de aparición, primero horizontal y luego vertical, es decir, primero los 4 de arriba, siendo el primero el más a la izquierda. El punto aparecerá en el centro de la región.

- Primer voluntario: Los resultados del primer voluntario para los dos métodos utilizados pueden verse en la Tabla 4-15, Figura 4-19 y Figura 4-20.

Tabla 4-15: Estimación correcta o incorrecta de cada punto en la novena parte correspondiente del primer voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	%
1	✗	✗	✗	✓	✗	✓	✗	✓	✗	33.33
2	✗	✓	✓	✗	✗	✓	✗	✓	✓	55.56

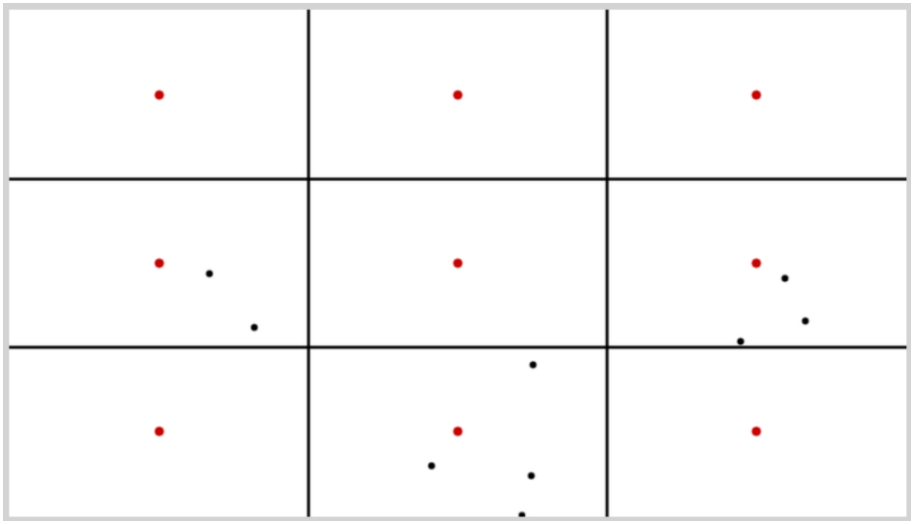

Figura 4-19: Estimación de los puntos de observación mediante el primer método en la prueba 3 del primer voluntario.

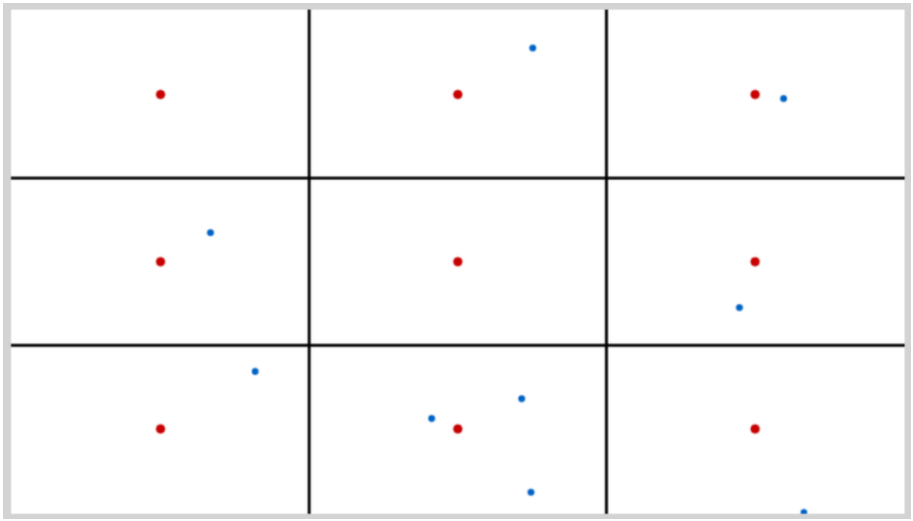

Figura 4-20: Estimación de los puntos de observación mediante el segundo método en la prueba 3 del primer voluntario.

- Segundo voluntario: En la Tabla 4-16, Figura 4-21 y Figura 4-22 se observan los resultados para ambos métodos del segundo caso.

Tabla 4-16: Estimación correcta o incorrecta de cada punto en la novena parte correspondiente del segundo voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	%
1	×	✓	✓	✓	×	×	×	✓	✓	55.55
2	✓	✓	×	✓	✓	✓	×	✓	✓	77.78

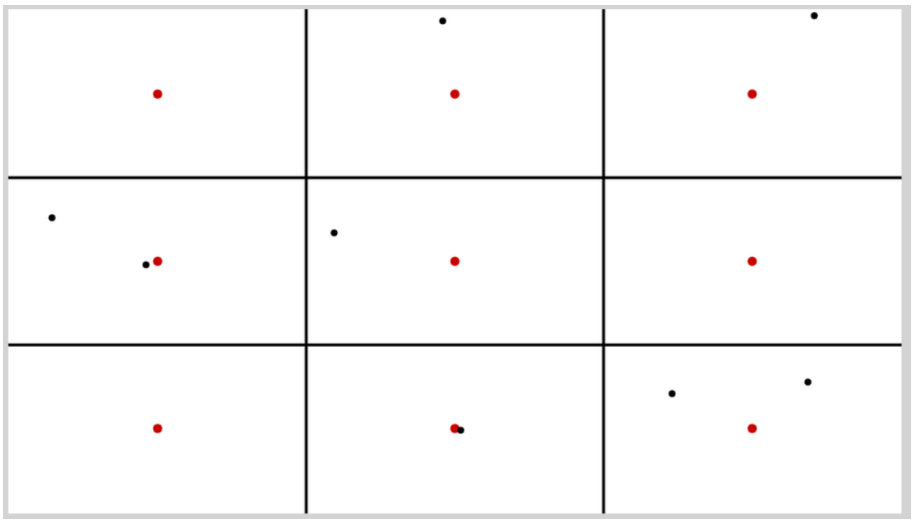

Figura 4-21: Estimación de los puntos de observación mediante el primer método en la prueba 3 del segundo voluntario.

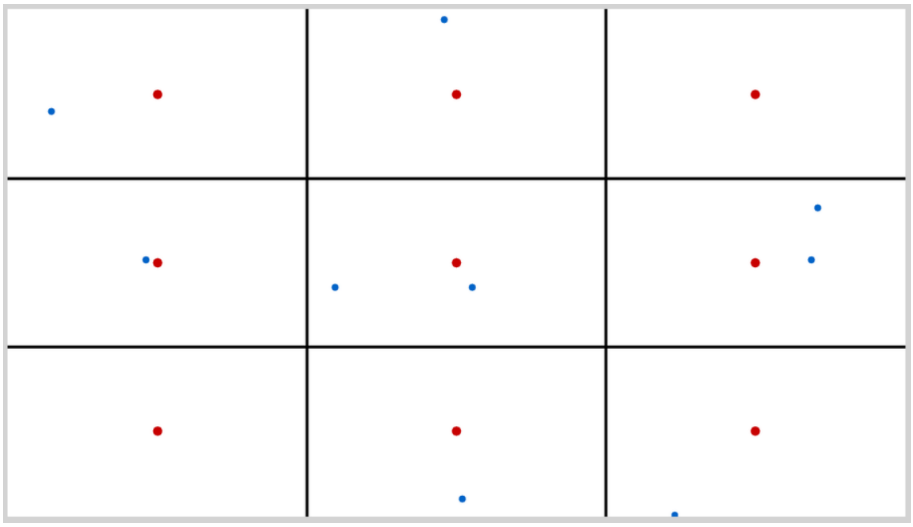

Figura 4-22: Estimación de los puntos de observación mediante el segundo método en la prueba 3 del segundo voluntario.

- Tercer voluntario: Los resultados del tercer voluntario se encuentran en la Tabla 4-17, Figura 4-23 y Figura 4-24:

Tabla 4-17: Estimación correcta o incorrecta de cada punto en la novena parte correspondiente del tercer voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	%
1	×	×	✓	✓	×	✓	✓	✓	×	55.55
2	×	✓	✓	×	✓	×	✓	×	✓	55.55

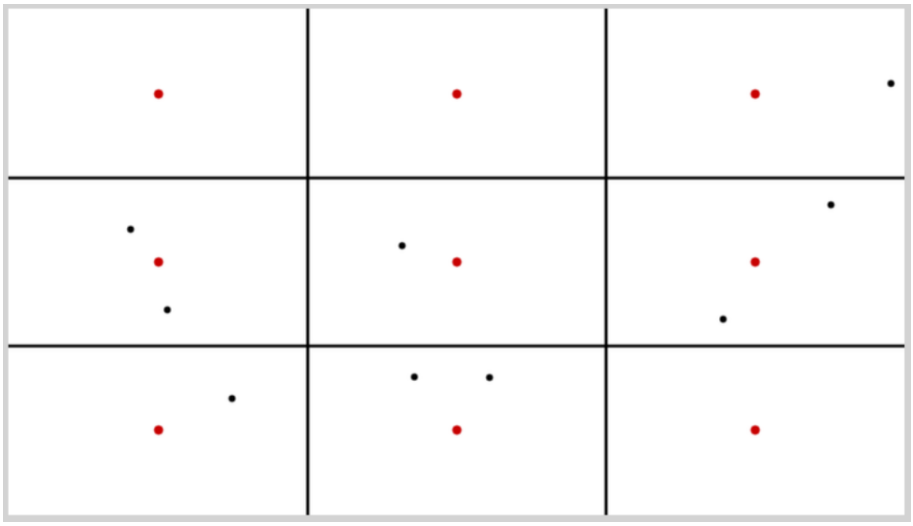

Figura 4-23: Estimación de los puntos de observación mediante el primer método en la prueba 3 del tercer voluntario.

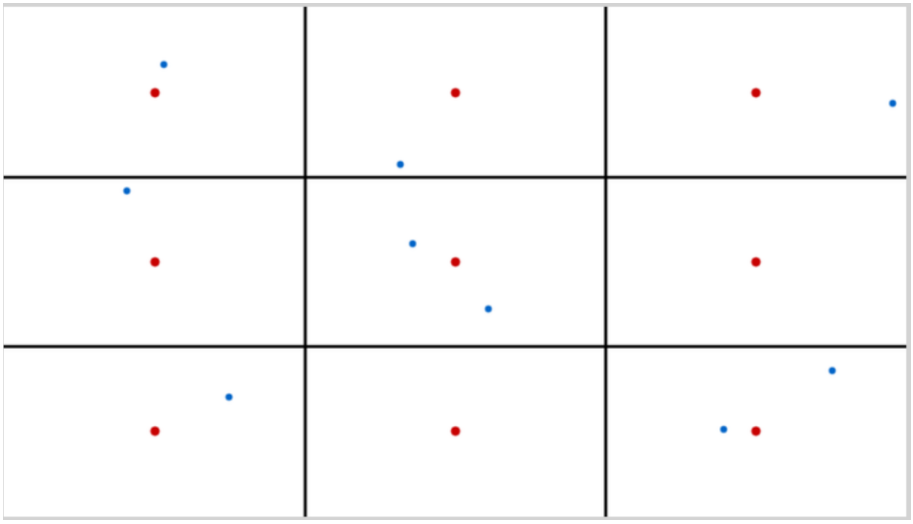

Figura 4-24: Estimación de los puntos de observación mediante el segundo método en la prueba 3 del tercer voluntario.

- Cuarto voluntario: Los resultados del cuarto voluntario se encuentran en la Tabla 4-18, Figura 4-25 y Figura 4-26.

Tabla 4-18: Estimación correcta o incorrecta de cada punto en la novena parte correspondiente del cuarto voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	%
1	×	×	×	×	×	✓	✓	×	×	22.22
2	×	×	×	×	×	×	✓	✓	✓	33.33

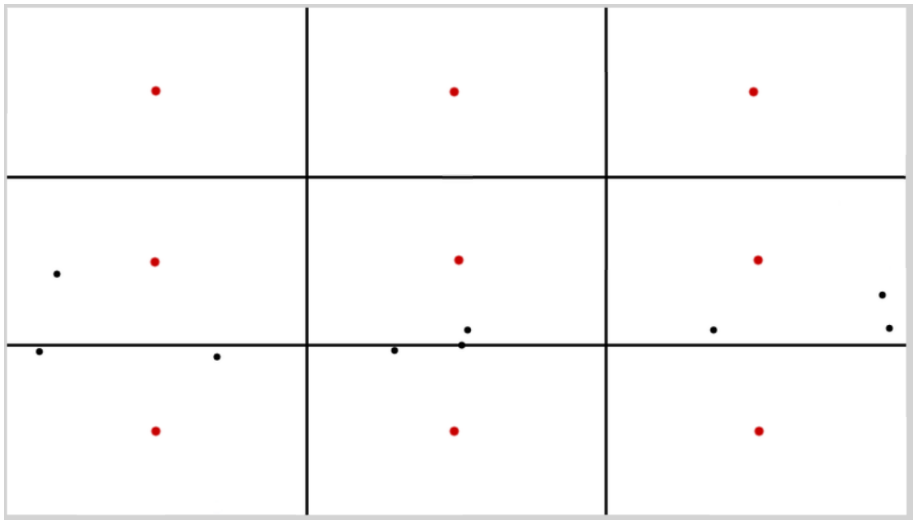

Figura 4-25: Estimación de los puntos de observación mediante el primer método en la prueba 3 del cuarto voluntario.

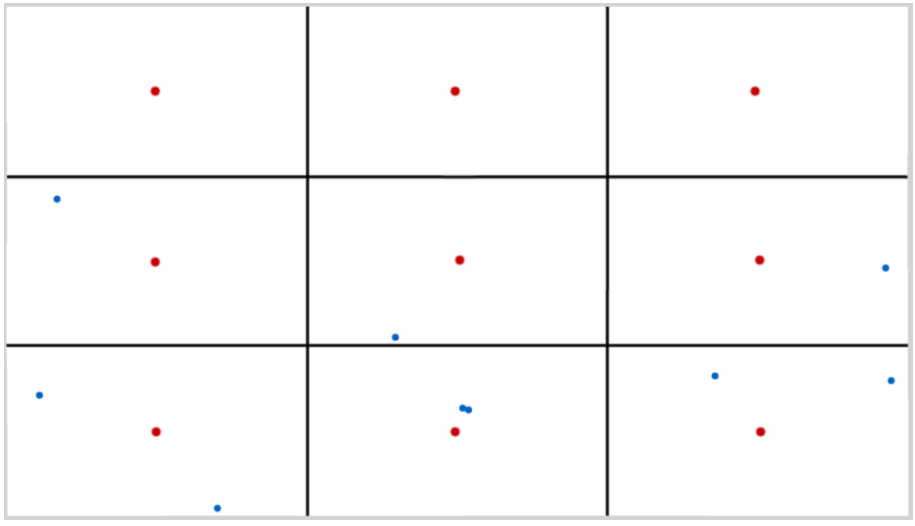

Figura 4-26: Estimación de los puntos de observación mediante el segundo método en la prueba 3 del cuarto voluntario.

- Quinto voluntario: Los resultados del cuarto voluntario se encuentran en la Tabla 4-19, Figura 4-27 y Figura 4-28:

Tabla 4-19: Estimación correcta o incorrecta de cada punto en la novena parte correspondiente del cuarto voluntario. La última columna es el porcentaje de estimaciones correctas.

Puntos / Método	1	2	3	4	5	6	7	8	9	%
1	✓	×	✓	✓	×	✓	✓	✓	✓	77.78
2	✓	✓	✓	✓	×	✓	✓	✓	✓	88.89

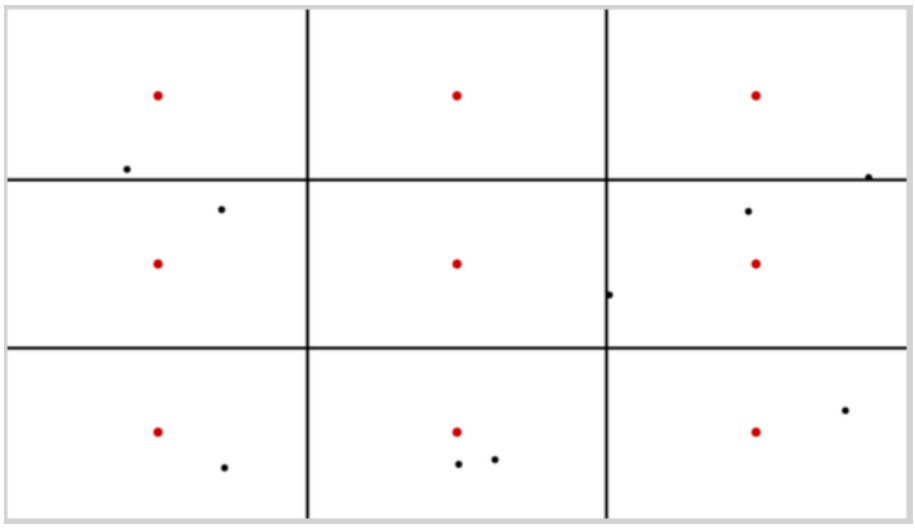

Figura 4-27: Estimación de los puntos de observación mediante el primer método en la prueba 3 del quinto voluntario.

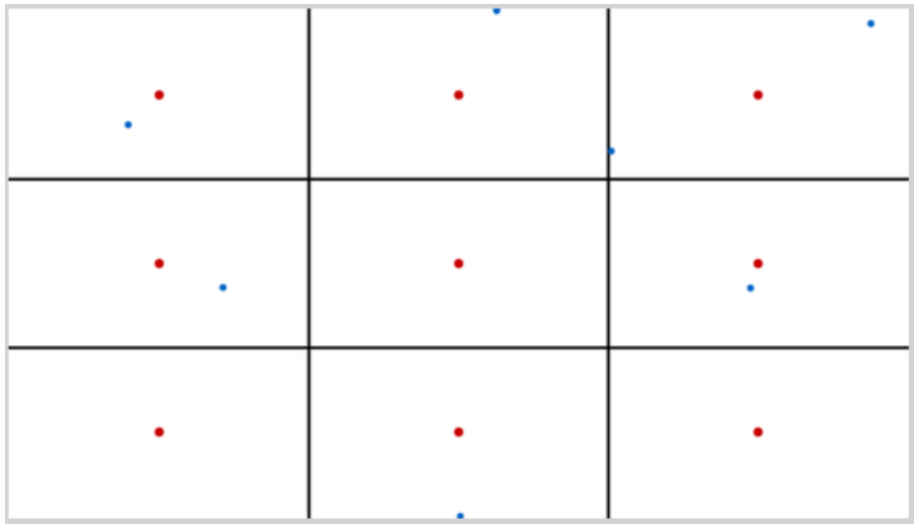

Figura 4-28: Estimación de los puntos de observación mediante el segundo método en la prueba 3 del quinto voluntario.

En la última prueba se evaluaba el requisito de poder distinguir a que novena parte de la pantalla mira un usuario, a diferencia de la anterior, tenemos menos puntos y se encuentran en el centro de cada región para que no se pierda tanta precisión por el pequeño espacio de las regiones.

En total, se tiene un 48.88% de aciertos en la estimación del punto de observación con el primer método y un 62.22% con el segundo método. En la Tabla 4-20 y Tabla 4-21 se encuentran resúmenes de los aciertos de cada voluntario con cada método.

Tabla 4-20: Aciertos con el primer método (sobre 1).

Región	1	2	3	4	5	6	7	8	9	Total
Voluntario 1	0	0	0	1	0	1	0	1	0	3
Voluntario 2	0	1	1	1	0	0	0	1	1	5
Voluntario 3	0	0	1	1	0	1	1	1	0	5
Voluntario 4	0	0	0	0	0	1	1	0	0	2
Voluntario 5	1	0	1	1	0	1	1	1	1	7
Total										22
Medía	0,2	0,2	0,6	0,8	0	0,8	0,6	0,8	0,4	
Desviación	0,45	0,45	0,55	0,45	0	0,45	0,55	0,45	0,58	

Tabla 4-21: Aciertos con el segundo método (sobre 1).

Región	1	2	3	4	5	6	7	8	9	Total
Voluntario 1	0	1	1	0	0	1	0	1	1	5
Voluntario 2	1	1	0	1	1	1	0	1	1	7
Voluntario 3	0	1	1	0	1	0	1	0	1	5
Voluntario 4	0	0	0	0	0	0	1	1	1	3
Voluntario 5	1	1	1	1	0	1	1	1	1	8
Total										28
Medía	0,4	0,8	0,6	0,4	0,4	0,6	0,6	0,8	1	
Desviación	0,55	0,45	0,55	0,55	0,55	0,55	0,55	0,45	0	

En las siguientes tablas se tiene el porcentaje de en qué regiones son estimados los puntos de cada región. Se vuelve a comprobar, con el primer método, que los errores de las regiones de arriba se cometen de forma vertical en los de abajo, los de la primera región en la cuarta, los de la segunda en la quinta y octava, los de la tercera en la sexta, los de la cuarta en la séptima, los de la quinta en la octava y fuera, y los de la sexta en la novena. Los errores de las tres regiones de abajo se cometen arriba. Además, se dan un par de casos de error horizontal. Las regiones con mayor error son las de más arriba y principalmente la del centro.

Con el segundo método se producen menos errores en las regiones de arriba, siendo los que hay hacia abajo, además mejora la región del centro y la novena.

Tabla 4-22: Porcentajes de las estimaciones en cada novena parte con el primer método.

Región Real / Región Estimación	1	2	3	4	5	6	7	8	9
1	20	0	0	0	0	0	0	0	0
2	0	20	0	0	0	0	0	0	0
3	0	0	60	0	0	0	0	0	0
4	80	0	0	80	0	0	0	0	0
5	0	20	0	0	0	0	20	20	0
6	0	0	40	0	20	80	0	0	60
7	0	0	0	20	0	0	60	0	0
8	0	60	0	0	60	0	20	80	0
9	0	0	0	0	0	20	0	0	40
Fuera	0	0	0	0	20	0	0	0	0
	100	100	100	100	100	100	100	100	100

Tabla 4-23: Porcentajes de las estimaciones en cada novena parte con el segundo método.

Región Real / Región Estimación	1	2	3	4	5	6	7	8	9
1	40	0	0	20	0	0	0	0	0
2	0	80	0	0	0	0	0	0	0
3	0	0	60	0	20	0	0	0	0
4	60	0	0	40	0	0	0	0	0
5	0	20	0	0	40	0	20	20	0
6	0	0	40	0	0	60	0	0	0
7	0	0	0	40	0	0	60	0	0
8	0	0	0	0	40	0	20	80	0
9	0	0	0	0	0	40	0	0	100
Fuera	0	0	0	0	0	0	0	0	0
	100	100	100	100	100	100	100	100	100

4.4.3.1 Conclusiones

Como conclusiones de esta tercera prueba se tiene que en el eje horizontal vuelve a cumplirse con el requisito salvo en dos casos. En el eje vertical se vuelven a dar más casos de error, pero en esta prueba tiene mejor porcentaje de acierto el segundo método que si supera el 50% a diferencia del primero. También hay que tener en cuenta que en esta prueba el punto estaba en el centro de la región y las regiones eran más pequeñas.

A continuación, se valorará el error entre el punto y la estimación para un análisis más preciso.

4.4.4 Análisis cuantitativo global

Además de valorar el cumplimiento de los requisitos para el distinto grado de exactitud de estimación del punto de observación, se calcularán los errores cometidos entre el punto exacto donde mira un usuario y el punto que estima el programa en pixeles. Los errores son calculados para cada coordenada del punto, eje *x* e *y*, habiendo dos estimaciones para el eje vertical con cada uno de los métodos.
Entre todos los datos se encuentra uno en el eje horizontal y el eje vertical del primero método que discrepa bastante de los demás, se encuentra en rojo en Tabla A-1 y Tabla A-2, y no han sido usados para los cálculos de las medias y desviaciones típicas.

Primero se analizará la media de los errores totales junto con la media de los errores en absoluto en cada eje, en la Figura 4-29 se pueden observar las representaciones de ambos. Con esta gráfica se concluye que el eje horizontal comete errores hacia ambos lados casi por igual, aunque un poco más hacia la derecha, siendo éste el eje que menos errores comete a nivel absoluto. Con respecto al eje vertical, con ambos métodos se cometen más errores hacia abajo, sobre todo con el primer método, pero a nivel absoluto el segundo método es el que más errores comete, aunque con poca diferencia.

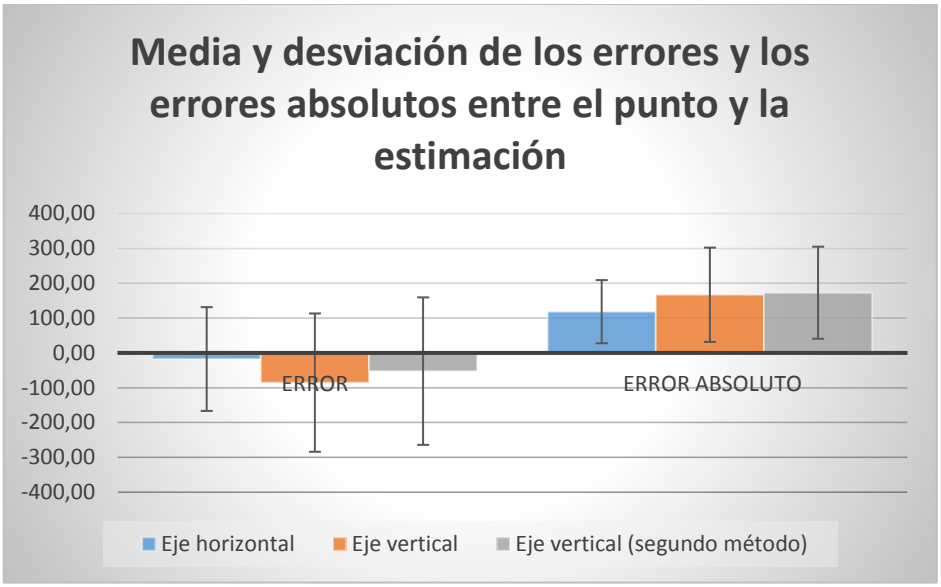

Figura 4-29: Media y desviación de los errores y los errores absolutos entre el punto y la estimación.

Se analizarán también las medias de los errores positivos y negativos por separado, quedando claro que en el eje horizontal es donde menos errores se cometen y casi en la misma medida hacia los dos lados, y que en el eje vertical se producen más errores hacia arriba con el segundo método y hacia abajo con el primero.

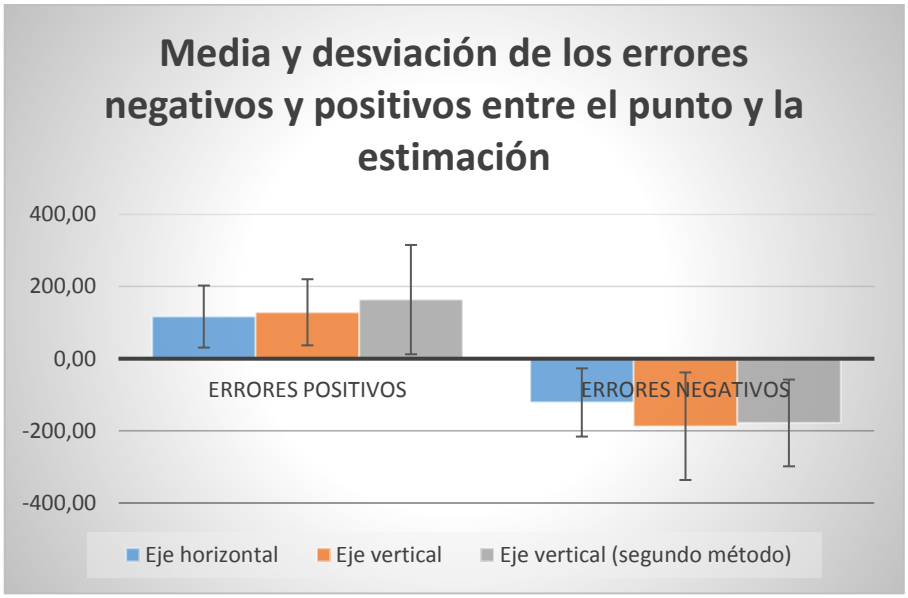

Figura 4-30: Media y desviación de los errores negativos y positivos por separado entre el punto y la estimación.

4.4.4.1 Error cuadrático medio

A continuación, se calculará el error cuadrático medio usando la fórmula comentada en el capítulo 2, ecuación (12). Se usará esta medida como un criterio para seleccionar un estimador apropiado, ver Tabla 4-24.

Tabla 4-24: Errores cuadráticos medios en cada eje.

	Error cuadrático medio
Eje horizontal	149 pixeles
Eje vertical primer método	215 pixeles
Eje vertical segundo método	217 pixeles

Con esta medida se vuelve a llegar a la misma conclusión, el eje horizontal es el más preciso. Con respecto al eje vertical apenas hay diferencia entre los dos métodos, pero el error del primer método es dos píxeles menor que el segundo, por lo que no se ha logrado mejorar de forma global utilizando este segundo método, aunque si en algunos casos comentados.

5 Conclusiones y trabajo futuro

5.1 Introducción

Tras detallar todo lo que se ha implementado en este trabajo y realizar la evaluación de dicha implementación, en este capítulo, se procede a exponer las conclusiones y proponer futuras vías de trabajo.

5.2 Conclusiones

El objetivo de este Trabajo de Fin de Máster era investigar e implementar un método para estimar el punto de observación de un usuario en una pantalla. Siendo un estudio en auge, existen muchos trabajos al respecto con los que se intenta dar solución a varios ámbitos, como la comunicación de la persona con las máquinas mediante la mirada o el aprendizaje de la lectura.

Los métodos actuales para el seguimiento de la mirada utilizan iluminación infrarroja, cámaras de alta calidad de video, y requieren una posición relativa estable entre el ojo del usuario y la cámara. Pero estas cámaras son caras y no son viables para los dispositivos cotidianos. Por esto en este proyecto se centró en detectar la mirada de un usuario sobre una pantalla con la mayor precisión posible usando sólo la webcam del ordenador o tableta del usuario, lidiando con recursos computacionales limitados, baja resolución de imágenes, condiciones adversas de luz, tamaños pequeños de pantalla donde mapear la mirada y movimientos de cabeza del usuario.

Como solución se ha implementado un método *Feature-Based*, primero detecta unos puntos característicos del usuario y utiliza sus distancias para, mediante un sistema, interpolar al punto de observación en la pantalla. Este sistema utiliza los centros de los irises y las comisuras de los ojos para detectar los ejes horizontales y verticales de las coordenadas del punto de observación. Además, se implementó un segundo método para intentar mejorar la diferencia de precisión existente entre ambos ejes, teniendo más error el eje vertical. El segundo método utiliza como puntos característicos los párpados inferior y superior de los ojos para conocer a que altura, coordenada vertical, se encuentra el punto de observación.

Para evaluar los requisitos del programa se ha pedido a 5 voluntarios que realicen unas pruebas, siguiendo unos requisitos explícitos para cada prueba. También se está evaluando que el programa funcione con varias personas, y que detecte correctamente los puntos característicos de cada una. Aunque no se realizaron explícitamente pruebas para estos dos primeros requisitos, es algo indispensable para las demás pruebas, lo que se logró con muy buenos resultados.

Con respecto a los resultados de las pruebas que evalúan los requisitos de precisión en la estimación del punto de observación del usuario, comentar que no se ha logrado lo esperado al comenzar este TFM. En un comienzo se pensó, que sería más sencillo y rápido poder conocer la dirección de la mirada de una persona disponiendo solo de una imagen, el caso es que no es así, por lo que se sigue investigando para intentar conseguir un *gaze-tracking* sin la ayuda de infrarrojos y con imágenes de poca calidad. Los errores cuadráticos medios son de 149 píxeles en el eje horizontal y 215 píxeles en el eje vertical en una pantalla de 1536x864 píxeles, lo cual es bastante si se requiere una precisión alta. Por esto las pruebas

de este trabajo se centraron en otro tipo de requisitos. En primer lugar, si el usuario mira o no a la pantalla, seguido de a que parte de la pantalla mira, si dividimos está en distintas partes. Estos requisitos, aunque no se cumplen al 100%, si tienen más del 50% de aciertos. En referencia a los dos métodos implementados, aunque haya casos en los que el segundo método mejora al primero, el error sigue siendo menor con el primer método, por lo que no se logra mejorar la precisión en el eje vertical buscada.

El estudio de este TFM abarca un gran abanico de cuestiones, puede que la solución propuesta no haya sido la más acertada, pero fue la elegida en un primer momento y la que se llegó a implementar en el tiempo disponible para este trabajo. Una de las cuestiones a las que no se ha podido dar solución, aunque se comentó una posible idea en capítulos anteriores, es la de dar libertad de movimiento a la cabeza, ya que es algo que afecta bastante al resultado final.

En el comienzo de la implementación de este trabajo, se quiso hacer hincapié en la precisión a la hora de detectar el centro del iris y las demás características. Cuestión bastante importante, que llevó un tiempo considerable, y le resto tiempo a otras cuestiones, debido a la baja resolución de las imágenes capturadas. Y quizás debido a esto y a la limitación de tiempo de este trabajo no se han podido probar otros métodos que quizás hubieran funcionado mejor.

Por último, recalcar que, aunque no hayan sido los resultados más esperado, sí que cumplen en gran medida con los requisitos y principalmente se ha logrado detectar el centro del iris con bastante precisión, que fue una primera meta bastante difícil.

5.3 Trabajo futuro

A continuación, se expondrán algunas posibles líneas con las que continuar la investigación de este TFM:

- Debido a que no se ha llegado a mejorar la precisión en el eje vertical con el segundo método propuesto, se podría utilizar un tercer método que implemente de alguna forma todos los puntos característicos de ambos métodos, debido a que si mejoraba los resultados de la parte superior de la pantalla.
- Utilizar lo planteado para el movimiento de la cabeza con la homografía, creando así un programa para situaciones más reales, en las que además no sea necesario que le usuario tenga que calibrar cada vez que va a usarlo.
- También se podría intentar fusionar el método de estimar el punto de observación de la pantalla con algún otro, para ganar más precisión.
- Por último, utilizar esta estimación para poder mover el ratón o poder dibujar sobre la pantalla, para estos casos en los que las imágenes del usuario se capturaran de forma seguida, se podría llevar un seguimiento en el que se tuviera en cuenta también el *frame* anterior.

Glosario de acrónimos

- **CHT**: Circular Hought Transform
- **ECM**: Error Cuadrático Medio
- **FLANN**: Fast Library for Approximate Nearest Neighbors
- **HOG**: Histogramas Orientados a Gradient
- **IR**: Infrarrojos
- **LBPs:** Patrones Binarios Locales
- **LLI**: Local Linear Interpolation
- **PC-EC**: Pupil Center-Eye Corner
- **RANSAC**: RANdom SAmple Consensus
- **RBF**: Función de Base Radial
- **SIFT**: Scale-Invariant Feature Transform
- **SURF**: Speeded-Up Robust Features

Referencias

[1] Ippei Torii, Kaoruko Ohtani, Takahito Niwa, Naohiro Ishii, "System and method for detecting gaze direction" Lecture Notes in Computer Science, vol. 9739, pp. 237, 2016, ISSN 0302-9743, ISBN 978-3-319-40237-6.

[2] Chang-Zheng Li, Chung-Kyue Kim, Jong-Seung Park, "The Indirect Keyboard Control System by Using the Gaze Tracing Based on Haar Classifier in OpenCV" 2009 International Forum on Information Technology and Applications.

[3] Hyun-Cheol Kim Jihun Cha, Won Don Lee, "Eye Detection for Gaze Tracker with Near Infrared Illuminator" 2014 IEEE 17th International Conference on Computational Science and Engineering.

[4] Bindhu K Rajan, Anjali K U, et al, "Real-Time Nonintrusive Monitoring and Detection of Eye Blinking in view of Accident Prevention due to Drowsiness" 2016 International Conference on Circuit, Power and Computing Technologies [ICCPCT].

[5] B. Kunka, B. Kostek, M. Kulesza, P. Szczuko, A. Czyzewski. "Gaze-tracking based audio-visual correlation analysis employing Quality of Experience Methodology", Intelligent Decision Technologies Journal, Greece, 2009.

[6] OpenCV library, online: https://opencv.org/.

[7] Onur Ferhat, Fernando Vilariño, "Low Cost Eye Tracking: The Current Panorama". Hindawi Publishing Corporation Computational Intelligence and Neuroscience Volume 2016, Article ID 8680541, 14 pages, February 2016.

[8] Y. Ono, T. Okabe, and Y. Sato, "Gaze estimation from low resolution images," in Advances in Image and Video Technology, L.-W.ChangandW.-N.Lie,Eds.,vol.4319of Lecture Notes in Computer Science, pp. 178–188, Springer, 2006.

[9] Kazuki Fukushima y Naruki Shirahama, "Proposal of Eye-gaze Recognize Method for Input interface without Infra-red Ray Equipment", IEEE SNPD 2014.

[10] A. George and A. Routray. 2016. "Fast and Accurate Algorithm for Eye Localization for Gaze Tracking in Low Resolution Images". arXiv preprint arXiv:1605.05272 (2016).

[11] L. Sesma, A. Villanueva, and R. Cabeza, "Evaluation of pupil center-eye corner vector for gaze estimation using a web cam", in Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12),C.H.Morimoto,H.O.Istance,S.N. Spencer, J. B. Mulligan, and P. Qvarfordt, Eds., pp. 217–220, ACM, Santa Barbara, Calif, USA, March 2012.

[12] J. Zhu and J. Yang, "Subpixel eye gaze tracking," in Proceedings of the 5th IEEE International Conference on Automatic Face Gesture Recognition, pp. 131–136, IEEE, Washington, DC, USA, May 2002.

[13] M. Reale, T. Hung, and L. Yin, "Viewing direction estimation basedon3DeyeballconstructionforHRI,"in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition—Workshops (CVPRW '10), pp. 24–31, IEEE, San Francisco, Calif, USA, June 2010.

[14] E. Wood and A. Bulling, "EyeTab: model-based gaze estimation on unmodified tablet computers," in Proceedings of the 8th Symposium on Eye Tracking Research and Applications (ETRA'14), P. Qvarfordtand D. W. Hansen, Eds., pp. 207–210, ACM, Safety Harbor, Fla, USA, March 2014.

[15] OpenCV Face Detection using Haar Cascades, online: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html.

[16] K. Toennies, F. Behrens, M. Aurnhammer. “Feasibility of hough-transform-based iris localization for real-time-application”. In 16th International Conference on Pattern Recognition, 2002. Proceedings, vol. Two, 1053–1056, 2002.
[17] D. Young, H. Tunley, and R. Samuels, “Specialised hough transform and active contour methods for real-time eye tracking”. University of Sussex, Cognitive & Computing Science, 1995.
[18] Radu Gabriel Bozomitu, Alexandru Păsărică, et al, “Pupil Detection Algorithm Based on RANSAC Procedure”, In Signals, Circuits and Systems (ISSCS), 2017 International Symposium on (pp. 1-4). IEEE
[19] P. Yang, B. Du, S. Shan, and W. Gao, “A novel pupil localization method based on gabor eye model and radial symmetry operator,” in Image Processing, 2004. ICIP’04. 2004 International Conference on, vol. 1. IEEE, 2004, pp. 67–70.
[20] R. Valenti and T. Gevers, “Accurate eye center location through invariant isocentric patterns,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 34, no. 9, pp. 1785–1798, 2012.
[21] R. Valenti and T. Gevers,” Accurate Eye Center Location and Tracking Using Isophote Curvature”, in IEEE Conference on Computer Vision and Pattern Recognition, 2008.
[22] Timm and E. Barth, “Accurate eye centre localisation by means of gradients.” VISAPP, vol. 11, pp. 125–130, 2011.
[23] Kunka, Bartosz & Kostek, Bozena. (2009). “Non-intrusive infrared-free eye tracking method”. Signal Processing Algorithms, Architectures, Arrangements, and Applications Conference Proceedings (SPA). 105 – 109, 2009.
[24] N. H. Cuong, H. T. Hoang, "Eye Gaze Detection with a Single WebCAM Based on Geometry Features Extraction", 2010 11th International Conference on Control Automation Robotics and vision, pp. 2507-2512, December, 2010.
[25] Dlib C++ Library, online: http://dlib.net/intro.html.
[26] Wikipedia: Mínimos Cuadrados, online: https://es.wikipedia.org/wiki/M%C3%ADnimos_cuadrados.
[27] OpenCV tutorials: Basic concepts of the homography, online: https://docs.opencv.org/3.4/d9/dab/tutorial_homography.html
[28] OpenCV tutorials: Introduction to SIFT, online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro
[29] OpenCV tutorials: Introduction to SURF, online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html#surf
[30] OpenCV tutorials: Brute-Force Matcher y FLANN, online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html
[31] OpenCV tutorials: Features2D + Homograph, online: https://docs.opencv.org/3.0-beta/doc/tutorials/features2d/feature_homography/feature_homography.html#feature-homography

Anexos

A Datos de los errores de cada punto de las pruebas

Tabla A-1: Errores en el eje horizontal de los puntos de la prueba 2 y 3 para cada voluntario.

Voluntario 1	Voluntario 2	Voluntario 3	Voluntario 4	Voluntario 5
134	192	140	122	-279
53	-61	158	-47	-174
-110	-30	-242	90	14
-182	-66	-192	-71	30
-15	183	-38	370	192
77	204	2	164	-135
-45	-7	-31	148	-23
-161	-123	-56	-77	105
2	-7	1	185	-179
173	236	-231	200	-311
103	71	33	16	-225
-192	-458350	-72	-43	-78
138	120	-41	292	-353
200	135	-119	240	78
-86	303	59	-171	-163
51	-5	8	-41	-192
-86	181	47	172	52
-129	20	93	106	-65
-49	-107	-233	-217	-193
-163	19	-15	201	-109
-110	-27	-56	-9	-262
26	-96	-130	-229	12
-467	-304	-126	-103	-114
-126	-10	72	-20	-3
-84	137	54	71	-153

Tabla A-2: Errores en el eje vertical con el primer método de los puntos de la prueba 2 y 3 para cada voluntario.

Voluntario 1	Voluntario 2	Voluntario 3	Voluntario 4	Voluntario 5
-43	-316	-113	-514	-58
-322	-235	-152	-234	-88
-305	-83	-275	-373	108
-265	-354	-422	-238	-60
-33	-197	-346	-362	220
-143	-39	-345	-281	324
-289	-126	-256	-97	276
13	-154	61	-223	-36
-114	-122	-123	-123	-43
-20	-133	-95	-70	-203
-113	-161	-26	-19	93
3	138190	-29	-8	21
-73	30	-194	94	237
85	29	116	339	134
207	-31	-180	203	57
101	-4	190	214	9
-306	-213	-232	-311	-126
-462	125	-260	-441	-624
-314	134	17	-346	-141
-110	-6	-82	-156	92
-432	-761	-198	-144	-53
-134	-208	97	-115	89
-59	336	53	123	-61
-76	-3	90	169	-55
188	59	189	169	36

Tabla A-3: Errores en el eje vertical con el segundo método de los puntos de la prueba 2 y 3 para cada voluntario.

Voluntario 1	Voluntario 2	Voluntario 3	Voluntario 4	Voluntario 5
-215	359	189	-612	-330
-185	-88	-218	-218	-133
-122	79	-118	-218	-127
-261	-166	-170	-267	36
522	412	81	-431	72
198	324	-377	-112	30
-215	-286	-413	-388	-1
-247	-395	-238	-267	26
158	178	-256	-94	124
109	178	-324	-311	47
-145	178	-229	-23	-35
-72	-32	-211	-139	47
314	574	-108	37	182
21	613	-17	49	151
-62	-108	-108	110	273
-108	-84	-12	79	-27
-238	-29	-167	-182	-51
79	127	-122	-418	144
-7	-194	-18	-299	121
-189	4	335	-229	-41
-236	-42	-80	-252	191
-79	4	-185	-204	-43
17	245	57	-134	-297
-109,	-116	318	33	-144
-144	-144	2	91	-323