Actualización de hardware y software de topógrafo de córnea

David Marcelo Malpica Piñeros

Universidad Nacional de Colombia
Facultad de Ciencias, Departamento de Física
Bogotá, Colombia
2022

Actualización de hardware y software de topógrafo de córnea

David Marcelo Malpica Piñeros

Trabajo de grado presentado como requisito parcial para optar al título de:
Magister en Ciencias - Física

Director:
PhD. Yobani Mejía Barbosa.

Línea de Investigación:
Metrología Optica
Grupo de Investigación:
Grupo de óptica aplicada

Universidad Nacional de Colombia
Facultad de Ciencias, Departamento de Física
Bogotá, Colombia
2022

Dedicatoria

A mi mama, mi papa y mi hermano por infundirme valores, darme su apoyo y motivación a lo largo de mi desarrollo académico, desde los primeros niveles de educación, influyendo positivamente en este proceso, permitiendo en gran medida la culminación de este trabajo.

Agradecimientos

Expreso mis agradecimientos a las siguientes personas por contribuir en el desarrollo de este trabajo.

- Agradecimiento a mi director de trabajo de grado: Yobani Mejía Barbosa, por su acompañamiento, paciencia y apoyo en el desarrollo de este trabajo, en el que enriqueció mis conocimientos en temas de Óptica y de ciencia.
- Agradecimiento a los integrantes del grupo de Investigación de Óptica Aplicada, por sus aportes en mi formación académica que contribuyeron directamente en el desarrollo de este trabajo.
- Agradecimiento a mi familia, mi madre: Luz, mi padre: Clodobaldo y mi hermano: Sebastián, que por medio de su apoyo incondicional me brindaron una gran cantidad de enseñanzas y habilidades.
- Agradecimiento a Natalia, que a través de su cariño y apoyo me permitió inspirarme y motivarme, contribuyendo en mi formación académica.
- Agradecimiento a amigos y compañeros, entre los cueles resalto a Diego, por sus observaciones, sugerencias, amistad y confianza, a lo largo del desarrollo de este trabajo.

Resumen

Actualización de hardware y software de topógrafo de córnea

En el desarrollo de este trabajo, se realiza la actualización de un prototipo de topógrafo de córnea desarrollado para la Universidad hace más de una década, el cual se encontraba en desuso debido a que presentaba fallas en su funcionamiento. Esto implica la profundización en conceptos físicos de óptica tales como los fenómenos reflexión y refracción y aberraciones como la curvatura de campo y el astigmatismo. A partir del equipo actualizado se realiza la toma de la medida topográfica de superficies encargadas de dar forma a lentes de contacto, para de esta manera verificar el buen funcionamiento del topógrafo, sacar conclusiones respecto al mismo y complementar la documentación que se hará entrega al grupo de óptica aplicada de la Universidad sobre el topógrafo.

Palabras clave: Topografía, Córnea, Reflexión, Refracción, Aberraciones ópticas y Salud visual.

Abstract

Corneal Topography Hardware and Software Update

This document describes the upgrade of a corneal topographer prototype developed by the university more than a decade ago. However, the topographer is currently unused due to failures on its functioning. The enhancement of the device include the study of physical topics related with optics such as reflection and refraction phenomena or aberrations like field curvature and astigmatism. When the upgrade is completed the device makes topographic measures of the contact lens shaping surfaces in order to check if the topographer is working properly and offer conclusions about the results, also a document will be shared with the university applied optics group.

Keywords: Topography, Cornea, Reflection, Refraction, Optical aberration and Visual health.

Contenido

Lista de Figuras

Lista de Tablas

1. Introducción

El ojo es un sistema natural complejo que nos permite tener el sentido de la vista, esto lo hace principalmente porque tiene la capacidad de refractar la luz hacia un punto donde se encuentran los sensores encargados de procesar la información lumínica que les llega (conos y bastones). La refracción que sufren los rayos de luz al entrar al ojo es controlada en muchos aspectos por la córnea, que se comporta como un lente que separa dos medios diferentes, (interior y exterior del ojo) [Mejía, 2021]. Por lo tanto, es posible hacer una aproximación de la córnea y el interior del ojo exceptuando los conos y bastones, como si se tratara de una lente.

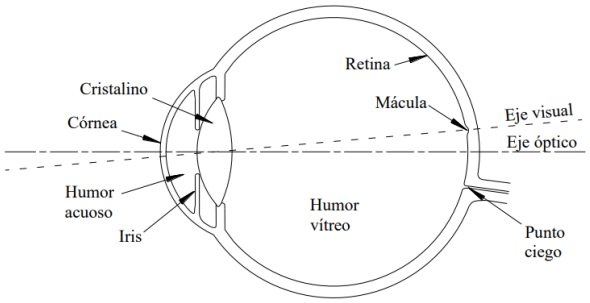


Figura 1-1.: Esquema del ojo [Mejía, 2021].

A partir de esto y teniendo en cuenta que la córnea es la interfaz por la que la luz atraviesa hacia el interior del ojo, como se puede observar en la Figura 1-1, se intuye lo valioso que resulta contar con una córnea en buen estado y ésta, al tener características similares a una lente, hace que resulte pertinente hablar de los fenómenos que presenta la luz al llegar a la córnea, para entender su función y fenómenos tales como la reflexión y la refracción. Es importante resaltar que el ojo es un ovoide aproximadamente esférico de 23 mm [Mejía, 2021] y así mismo se estima que la superficie de la córnea sea lisa y de forma regular, lo que implica que una cornea saludable tenga una topografía concordante con una superficie de revolución, elipsoidal o esférica. Pero, debido a distorsiones en la superficie, se pueden presentar aberraciones ópticas como el astigmatismo, la cual también deben ser tenida en cuenta a la hora de analizar el comportamiento de la luz al incidir en la córnea [Lara, 2021].

Por lo anterior, resulta útil contar con un equipo que permita realizar un análisis topográfico de la córnea. Antes del desarrollo de este proyecto la Universidad contaba con un prototipo

de topógrafo no funcional, el cual a través de este trabajo se actualizó, permitiéndole entre diversas funciones realizar el análisis topográfico de corneas y superficies con características similares como lentes de contacto.

Seguido de la introducción, este trabajo de grado cuenta con seis capítulos. En el capítulo siguiente, se exponen los fundamentos teóricos del topógrafo, en los cuales se expresan temas de óptica geométrica como algunas aberraciones y sus implicaciones en el diseño del topógrafo, análisis topográfico por medio de los polinomios de Zernike y datos relacionados con el ojo como las imágenes de Purkinje. A partir de estos, en el capítulo posterior se expone la problemática que ataca el trabajo y cuál es su solución, seguido a esto, se expone el desarrollo de la solución en el capítulo 4 y en el capítulo 5 la obtención de resultados obtenidos. En el siguiente capítulo se analizan estos resultados, para finalmente llegar a las conclusiones descritas en el capítulo final. Cabe mencionar que para contribuir a un trabajo más organizado y sencillo a la lectura, los capítulos son complementados con la información presente en los Anexos, los cuales se encuentran a final del documento y son citados a lo largo del mismo.

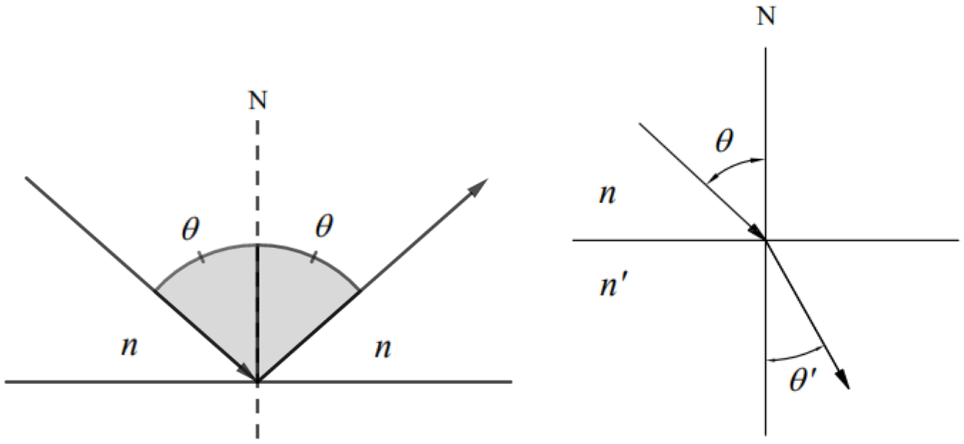
2. Fundamentos teóricos del topógrafo

La topografía de la córnea involucra diferentes conceptos relacionados con el comportamiento de la luz al incidir sobre una interfaz: la reflexión y refracción. Ambos fenómenos dependerán de la superficie e índice de refracción de dicho objeto. La reflexión de la luz se ve afectada de diferente manera al pasar sobre una superficie curva o sobre una plana, debido a que tienen forma diferente. Así mismo se ve que es diferente el efecto que se tiene sobre la luz el hacerla pasar por agua que por vidrio, ya que tienen diferente índice de refracción.

2.1. Reflexión y refracción

La refracción y reflexión, se pueden apreciar en la Figura 2-1 [Suárez, 2012], estos dos fenómenos se pueden entender a partir de la ley de reflexión y la ley de la refracción (Ley de Snell) [Jenkins and E.White, 2001]. La ley de reflexión nos dice que la luz se refleja con el mismo ángulo incidente medido desde la normal a la superficie en el punto de incidencia, mientras que la ley de Snell nos dice que la luz se refracta dependiendo del ángulo de incidencia y el cociente de los índices de refracción de los medios. La ley de la reflexión se puede apreciar en la Figura 2-2-a, mientras que la ley de Snell en Figura 2-2-b.

Figura 2-1.: Refracción y reflexión de la luz al cambiar de medio [Fernández, 2016].



(a) Ley de reflexión. (b) Ley de Snell [Mejía, 2021].

Figura 2-2.: Ley de reflexión y refracción.

La ley de Snell está dada por la ecuación (2-1).

$$n\, sin\theta = n'\, sin\theta', \tag{2-1}$$

donde n y n' son los índices de refracción de los medios [Mejía, 2021] [Suárez, 2012].

Algo similar ocurre en el ojo: mientras la luz se refracta al interior de él, permitiendo que esta sea procesada para que podamos ver, una fracción de la luz es reflejada, lo cual se observa cuando al mirar el ojo de una persona vemos que en él se refleja luz del entorno. Este ultimo fenómeno es el que permite el funcionamiento de algunas de las herramientas actuales utilizadas para identificar la topografía de superficies, incluyendo la de la córnea del ojo. Entre diferentes modelos y herramientas que existen para esto, se tienen aquellas que se denotan por topografía por reflexión especular y dentro de estos, está la topografía por arreglo de puntos; uno de estos topógrafos comerciales se puede apreciar en la Figura 2-3.

La medida de la topografía de la córnea a través de este instrumento se basa en capturar la luz reflejada por la córnea por medio de un sensor para después analizarla. Para esto, se necesita que el ojo se encuentre centrado y a la distancia apropiada respecto al arreglo de puntos, de esta manera se asegura una correcta interpretación de la lectura realizada por el sensor. Dado que los puntos de luz serán reflejados dependiendo de la geometría de la córnea, es posible deducir a través de los datos adquiridos, si se está presentando una forma simétrica o si la córnea cuenta con diferentes cambios en su superficie. En la Figura 2-4, podemos ver un esquema del funcionamiento de este equipo, donde el punto P simula la salida de luz dada por un punto del equipo, la cual es reflejada por el ojo haciendo que la

Figura 2-3.: Topógrafo comercial "Cassini" (Holanda), (Reproducido de: http://i-optics.com/cassini/).

luz atraviese el segmento L, el cual representa sistema óptico, para después llegar al sensor el cual esta ejemplificado por el segmento S.

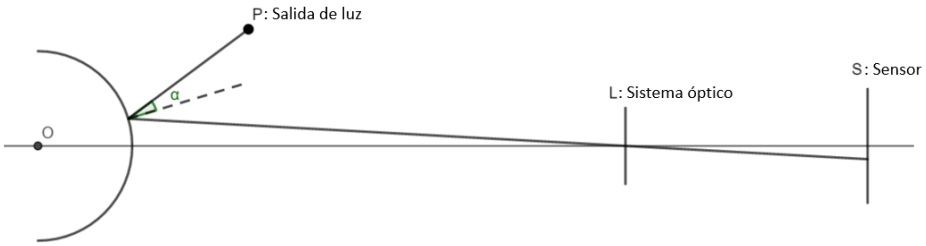


Figura 2-4.: Esquema del funcionamiento del topógrafo por reflexión especular de puntos.

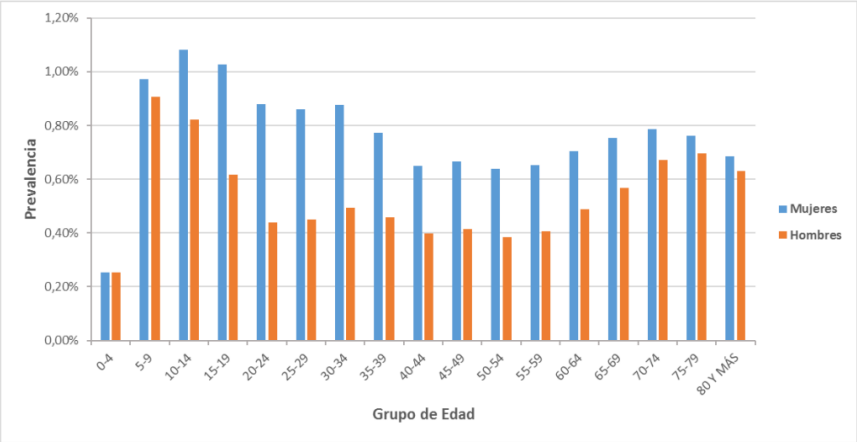
Los topógrafos de córnea suelen estar acompañados de un software, que se encarga de interpretar los datos del sensor y dar una representación de la forma de la córnea, ya sea a través de un mapa de elevación o alguna otra representación. Para conseguir esto, dicho software utiliza métodos numéricos donde compara la reflexión de los puntos con una referencia y con la superficie de la córnea.

2.2. Importancia

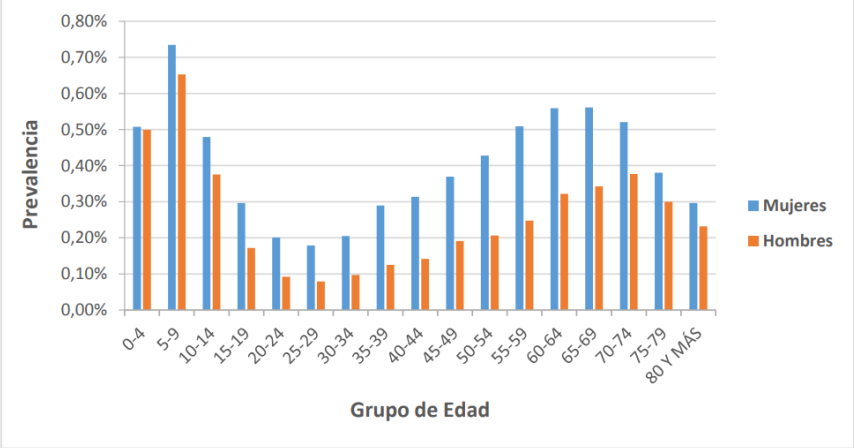
Los problemas de salud visual son de gran impacto para la persona que los posee, por lo que con el paso de los años se ha invertido tiempo en estudios que permitan identificar el

origen y dar una idea para el desarrollo de la solución. En la actualidad se encuentran bien estudiados los problemas visuales que pueden presentar las personas, los cuales comúnmente pueden deberse a problemas asociados a los sensores (conos y bastones) o malformaciones en el ojo, que es la que se suele estudiar en primera instancia, siendo la geometría de la córnea la encargada de generar el problema y por lo tanto brindar la información necesaria para solucionarlo. En estudios realizados en Colombia por el ministerio de salud, se ha demostrado que los problemas refractivos como la miopía, hipermetropía o astigmatismo, que dependen en gran medida por la córnea, están dentro de los problemas más comunes entre las personas que padecen dificultades en la visión, siendo estos causantes de la ceguera en caso de no ser tratados. En la Figura 2-5 se muestra la prevalencia de estos problemas en torno al sexo y edad, donde se aprecia que estas variables son significativas a la hora de combatir la miopía, hipermetropía y el astigmatismo [MINSALUD, 2016].

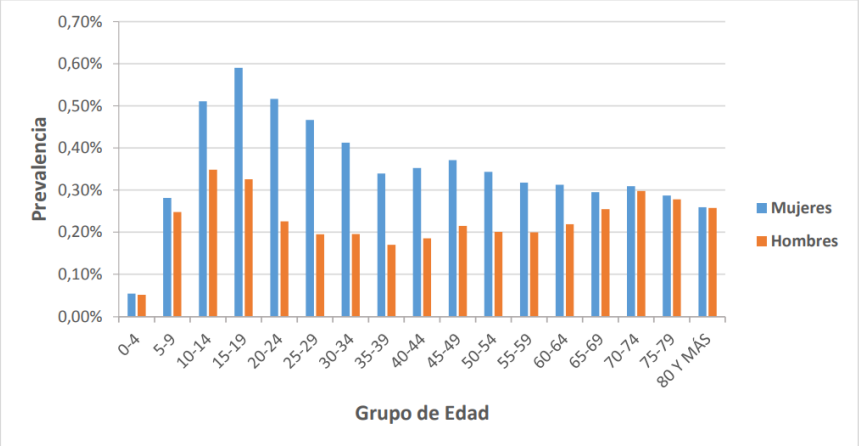
Dado que la córnea es la interfaz que separa al exterior e interior del ojo, esta es la que cuenta con el mayor poder refractivo del ojo como sistema óptico, por lo que es normal que sea la primera sección que se analice a la hora de buscar el origen de los problemas de salud visual, para lo que se requiere contar con una herramienta con la capacidad adecuada que permita la aproximación más cercana a la superficie real, lo que incluye tener la capacidad de representar incluso superficies no solo esféricas, sino que también cuente con la posibilidad de representar superficies aún más complejas [Klein and Mandell, 1995]. Para esto se desarrolla el estudio de la formación de imágenes descritas por la córnea y cómo estas pueden ser aprovechadas para interpretar su superficie, permitiendo en pacientes la adaptación de lentes de contacto o la toma de decisión acerca de si se puede o no realizar una cirugía refractiva [Khurana, 2007].



(a) Prevalencia del astigmatismo.



(b) Prevalencia de la hipermetropía.



(c) Prevalencia de la miopía.

Figura 2-5.: Prevalencia de problemas refractivos en torno al sexo y edad [MINSALUD, 2016].

2.3. Aberraciones y diseño del topógrafo

El diseño del topógrafo se basa en analizar la imagen formada por la córnea. Para esto conviene comenzar el diseño ejemplificando la córnea como un espejo esférico convexo, el cual forma una imagen virtual que depende de la extensión del objeto, como se aprecia en la Figura 2-6.

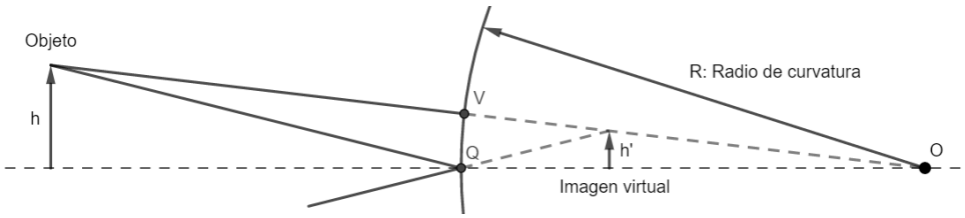


Figura 2-6.: Formación de la imagen de un espejo convexo.

Al conocer la ubicación de h' y h, que se muestran en la Figura 2-6, es posible utilizar la ecuación (2-2) para estimar el radio R del espejo, dado que las distancias s' y s, son las distancias de h a Q y de Q a h' respectivamente.

$$\frac{1}{s'} + \frac{1}{s} = \frac{2}{R}\,, \tag{2-2}$$

donde $s = \overline{hQ}$ y $s' = \overline{Qh'}$

Esto fue realizado por Christopher Scheiner en 1619 [Mejía, 2017] [Gellrich, 2011]. Sin embargo, este análisis presenta diferentes problemas de los cuales se destacan 2 en particular: El sistema debe tener un objeto de tal manera que la imagen virtual se produzca utilizando una mayor extensión de la córnea, para así adquirir una medida más general, ya que como se observa en la Figura 2-6, solo es necesario el segmento QV, para la formación de la imagen. La segunda problemática es el hecho de que el sistema no es capaz de identificar una superficie no esférica.

Para generar una imagen virtual que cuente con más información de la extensión de la interfaz, se debe utilizar un objeto de mayor tamaño en comparación con el radio de la córnea. Sin embargo, esto conlleva a sobrepasar la aproximación paraxial dando como consecuencia la aparición de aberraciones en el sistema, como lo son la curvatura de campo y el astigmatismo. La curvatura de campo se caracteriza porque la imagen del objeto no se encuentra sobre un plano (plano gaussiano), sino que por el contrario la imagen se ubica sobre una superficie curva. Como se observa en la Figura 2-7, se realiza el trazo de rayos

por una interface esférica convexa que separa dos medios con índice de refracción diferente y se observa que la imagen del objeto $\overline{OP}$ se sitúa sobre la llamada superficie de Petzval, la cual es posible aproximar como un paraboloide de revolución con vértice en O' [Mejía, 2021].

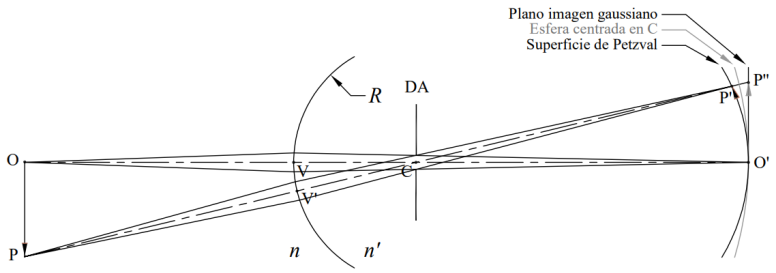


Figura 2-7.: Ilustración de la curvatura de campo [Mejía, 2021].

Esta aberración es de suma importancia dado que el sensor que se utiliza para capturar las imágenes suele ser plano, por lo que, al formarse la imagen virtual sobre una superficie curva, la lectura percibida por el sensor estará gradualmente desenfocada, siendo mayor el efecto a medida que se aleja del eje óptico. A parte de esto, es importante considerar la presencia del astigmatismo, el cual aparece debido a que los rayos de luz sobre diferentes planos meridionales convergen en puntos diferentes, lo que puede deberse a la extensión o posición del objeto o a una falta de simetría en los sistemas ópticos utilizados, como la córnea, lentes o diafragmas. Esta aberración se explica a partir del trazo de rayos de luz que van sobre los planos tangencial y sagital. En la Figura 2-8 se pueden apreciar dichos planos y en la Figura 2-9, se aprecia cómo la luz se enfoca en lugares diferentes dependiendo sobre qué plano está viajando [Mejía, 2021].

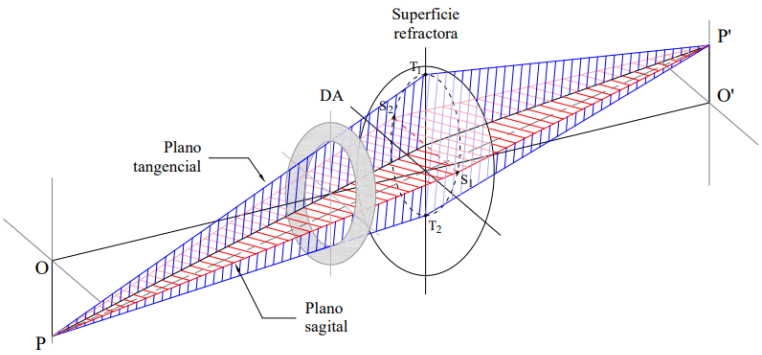


Figura 2-8.: Planos tangencial y sagital de una superficie refractora [Mejía, 2021].

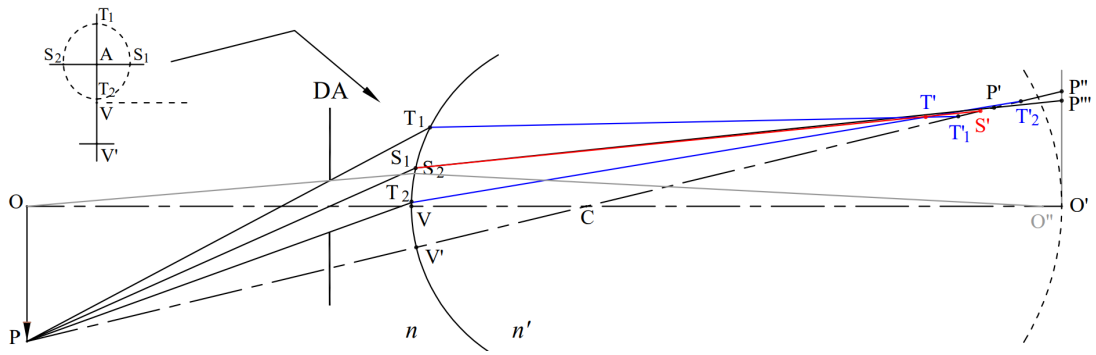


Figura 2-9.: Trazo de rayos sobre los planos tangencial y sagital, donde se evidencia una convergencia de los rayos a distancias diferentes, generando así el astigmatismo [Mejía, 2021].

Estas aberraciones conducen a que la imagen virtual formada, pueda ser vista sobre diferentes superficies curvas, dentro de las que destacan cuatro principales denotadas como T, M, S y Ptz y se pueden apreciar en la Figura 2-10. Sobre la superficie T converge la luz que viaja sobre el plano tangencial, mientras que en S la que viaja sobre el plano sagital. La superficie Ptz es la superficie de Petzval y M es la superficie promedio, donde a pesar de que la imagen no se encuentra bien enfocada, la imagen de cada punto corresponde con el círculo de menor confusión.

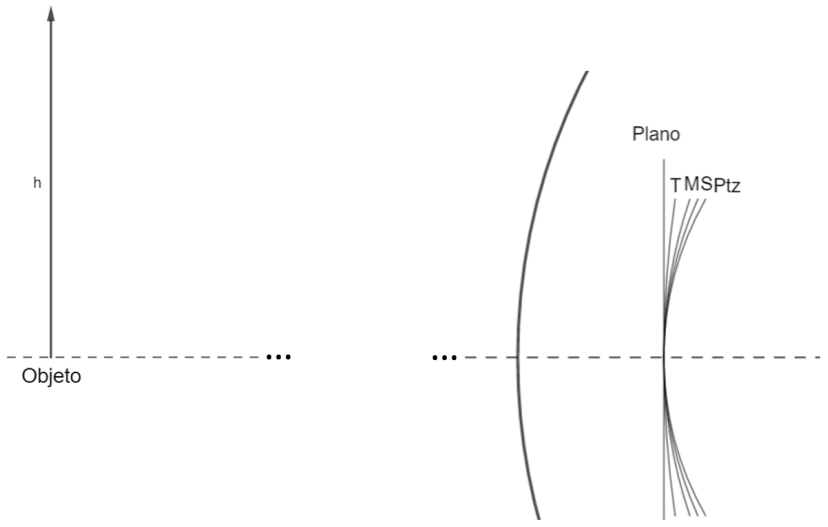


Figura 2-10.: Planos de ubicación de la imagen virtual formada por un objeto teniendo en cuenta las aberraciones de curvatura de campo y astigmatismo.

2.4. Pantalla de Hartmann

Partiendo de la interpretación del ojo como un espejo y como fue mencionado anteriormente, lo que se desea es que la imagen que llega al sensor sea lo más clara posible y plana. Pero de acuerdo con la curvatura de campo, se está generando la imagen sobre una superficie curva y dado el astigmatismo se tienen varias superficies donde en cada una hay un enfoque diferente de la imagen, por lo que es necesario implementar un mecanismo o montaje que permita obtener la imagen sobre un plano, siendo esta regular en su extensión [Mejía and Hernández, 2001]. Para esto se hace el análisis de forma inversa, asumiendo una imagen plana al interior del espejo como se observa en la Figura 2-11.

Figura 2-11.: Superficies de imagen.

La superficie donde se sitúa el objeto emisor de luz, se denota como pantalla de Hartmann y es la superficie M [Mejía and Galeano, 2009]. Ya que de esta esta forma, se asegura que la imagen virtual obtenida a partir de la reflexión, estará ubicada en un plano y contará con el menor desenfoque posible [Mejía, 2012] [Olarte and Mejía, 2006].

En el siglo pasado se desarrollaron dispositivos como el queratóscopio, el cual se basa en un objeto anillado, Figura 2-12, con la capacidad de identificar diferentes radios de la córnea [Singh and Krishna, 2022]. Sin embargo, cuenta con el problema de que solo puede describir superficies cónicas o de revolución [Daniel D. Garcia, 1998] [Evangelos and Konstantinos, 2020]. Por otro lado, se desarrolló el queratómetro, el cual cuenta con la capacidad de medir la superficie de la córnea a partir de dos radios de curvatura, propios de una superficie tórica [Aregay et al., 1996] [Darryl Meister, 1998], siendo este adecuado para medir problemas como el astigmatismo corneal, a pesar de esto no cuenta con la capacidad de describir superficies más complejas [Mejía, 2017].

Con el objetivo de poder estudiar la superficie de la córnea de una manera más precisa, el objeto luminoso que se utiliza se ubica sobre la superficie denotada como M, ya que un

Figura 2-12.: Queratóscopio.

punto lumínico sobre esta superficie tendrá como imagen el disco de menor confusión en la imagen plana virtual (detrás del espejo) como se muestra en la Figura 2-11. Al extender dicha superficie, se llega a un ovoide en el que se debe distribuir de manera apropiada los puntos luminosos para que la imagen virtual corresponda con un arreglo cuadriculado de puntos. Esta cuadricula regular de puntos se muestra en la Figura 4-10, la cual consta de 253 puntos equidistantes entre los puntos vecinos verticales y horizontales.

La distribución de los puntos sobre el ovoide generado por la superficie M, se realiza para una superficie de referencia, una esfera, que garantiza que la imagen sea una cuadricula ideal. Esto da como resultado la distribución de los puntos sobre la superficie de menor confusión, la cual es llamada como pantalla de Hartmann y su esquema se aprecia en la Figura 2-13.

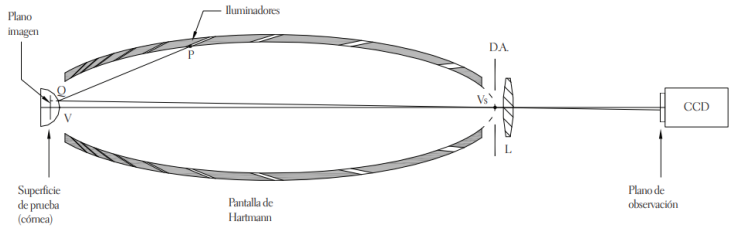


Figura 2-13.: Montaje de un topógrafo de córnea a partir de la pantalla de Hartmann [Mejía, 2012].

De esta manera se hace evidente que obtener una cuadricula deformada o con anomalías brinda información de un cambio en la superficie respecto a la superficie de referencia. Cabe aclarar que el punto central de la cuadricula forma imagen sobre el ovoide que describe la

pantalla de Hartmann, pero no es generado por esta, sino que es formado por una fuente de luz externa junto a una pantalla de vidrio pulido que brinda la figura de una cruz. Gracias a un sistema óptico dicha cruz forma imagen en el punto Vs que se aprecia en la Figura 2-13, y dado que es la imagen que debe mirar la persona a la que se le va a realizar el estudio topográfico, recibe el nombre de mirilla.

2.5. Imágenes de Purkinje

En el planteamiento teórico del topógrafo de córnea descrito anteriormente, se especifica que la imagen a analizar es generada específicamente por la reflexión de la córnea. Sin embargo, esta no es la única interfaz que atraviesa la luz al ingresar al ojo, por lo que es necesario aclarar el porque podemos asumir que la imagen que recibe el sensor, que es la que va a ser analizada, es exclusiva de la córnea.

Como se menciona anteriormente al interior del ojo se cuentan con diferentes índices de refracción, lo cual da lugar a diferentes interfaces al interior del mismo y con esto a la posibilidad de obtener diferentes imágenes. En la Figura 2-14, se muestra la parte al interior del ojo encargada de generar imagen, en la misma se muestra un haz de luz incidente el cual se denota como "I" y diferentes haces de luz salientes del ojo, denotados como P1, P2, P3 y P4. Estos haces son los encargados de generar imágenes, las cuales se conocen como las imágenes de Purkinje [Vieyra et al., 2018] [Abdulin et al., 2019].

Estas imágenes son enumeradas del uno al cuatro tal como se ven en la Figura 2-14, y son generadas a partir de la reflexión de la luz al incidir en la cara anterior de la córnea, en la cara posterior de la córnea, cara anterior del cristalino y en la cara posterior del cristalino respectivamente [Marín, 2006]. De esta manera se ve que la imagen número dos de Purkinje, se obtiene a partir de los rayos de luz que presentan una refracción en la cara anterior de la córnea, para posteriormente reflejarse en la cara posterior de la córnea y finalmente refractarse a la salida del ojo. De igual forma se tiene que las imágenes de Purkinje 3 y 4 se forman a partir de haces de luz que presentan más veces los fenómenos de reflexión y refracción en comparación a las imágenes 1 y 2, es por esto que para obtener dichas imágenes se requiere ya sea de un haz de luz incidente con una irradiancia alta o de un sensor muy preciso [Sigut and Sidha, 2011] [Tápias et al., 2014].

La afirmación anterior se puede sustentar a partir de las ecuaciones de Fresnel, las cuales indican el valor de los coeficientes de proporcionalidad que presenta la intensidad de las componentes perpendiculares y paralelas del campo eléctrico E de una onda de luz, al reflejarse y refractarse después de incidir por una interfase. Estas relaciones están descritas por las ecuaciones (2-3), (2-4), (2-5) y (2-6), donde los índices "i", "r" y "t" hacen referencia al rayo

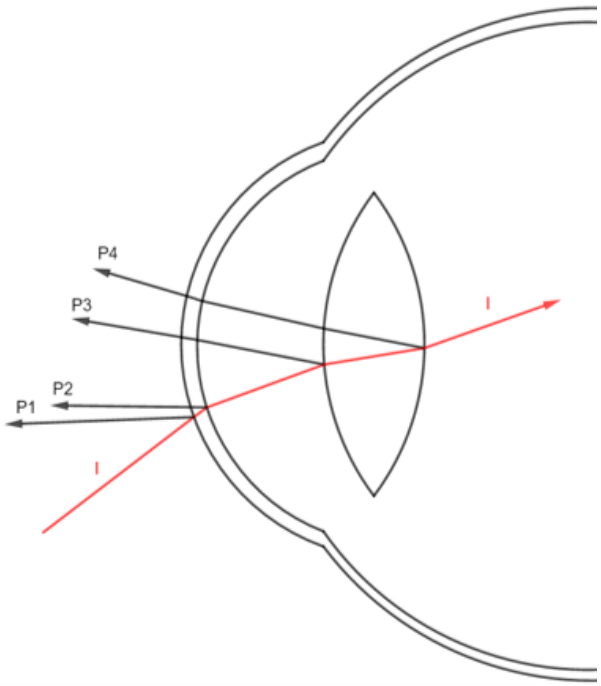


Figura 2-14.: Interfaces que generan las imágenes de Purkinje.

incidente, reflejado y transmitido respectivamente y "$\parallel$" y "$\perp$" a la componente paralela y perpendicular del mismo.

$$E_r^{\parallel} = \frac{n_t cos\theta_i - n_i cos\theta_t}{n_t cos\theta_i + n_i cos\theta_t} E_i^{\parallel} = r_{\parallel}\, E_i^{\parallel} \tag{2-3}$$

$$E_t^{\parallel} = \frac{2n_i cos\theta_i}{n_t cos\theta_i + n_i cos\theta_t} E_i^{\parallel} = t_{\parallel}\, E_i^{\parallel} \tag{2-4}$$

$$E_r^{\perp} = \frac{n_i cos\theta_i - n_t cos\theta_t}{n_i cos\theta_i + n_t cos\theta_t} E_i^{\perp} = r_{\perp}\, E_i^{\perp} \tag{2-5}$$

$$E_t^{\perp} = \frac{2n_i cos\theta_i}{n_i cos\theta_i + n_t cos\theta_t} E_i^{\perp} = t_{\perp}\, E_i^{\perp} \tag{2-6}$$

Donde $r_{\parallel}$, $t_{\parallel}$, $r_{\perp}$, y $t_{\perp}$ son llamados coeficientes de Fresnel y cumplen con ciertas restricciones, dado de que deben dar pie a leyes como la conservación de la energía y el principio de reversibilidad, lo que hace que, en pocas palabras, la suma de los rayos refractados y reflejados debe describir como resultado el rayo incidente. Dado esto, tenemos que a medida que el haz de luz se divide, tanto el haz reflejado como el trasmitido va disminuyendo en

magnitud de irradiancia, lo que conlleva a la idea de que a medida que se quiere obtener una figura de Purkinje más profunda, se va a contar con una mayor dificultad. Sumándole a esto el hecho de que los coeficientes de Fresnel dependen del ángulo de incidencia y de la diferencia entre los índices de refracción, se debe tener en cuenta que el cambio entre los índices de refracción es muy bajo entre las capas al interior del ojo, por lo que se tiene que el haz de luz reflejado al interior del ojo cuenta con una irradiancia mucho menor al haz incidente [Mejía, 2017].

Para el desarrollo de este trabajo se tendrá en cuenta solo la primera imagen de Purkinje, ya que las condiciones del sistema: sensibilidad del sensor e irradiancia emitida, hacen poco probable que se logre apreciar las imágenes de Purkinje 2, 3 y 4. Sin embargo, para evitar que la aparición de estas imágenes afecte la medida topográfica de la córnea, por medio del software se identifica a través de la irradiancia, qué imagen es la correcta para realizar el análisis.

2.6. Polinomios de Zernike

Los polinomios de Zernike son un conjunto de polinomios ortogonales sobre el disco de radio 1. Los cuales reciben su nombre en base al físico holandés Frits Zernike (1888– 1966) ganador del premio Nobel de Física en 1953 [16]. Estos polinomios son usados en óptica, dado que cuentan con la capacidad de representar aberraciones que puede sufrir el ojo humano, haciendo de estos una herramienta comúnmente usada en oftalmología [Recarte, 2017] [González, 2017].

Estos polinomios cuentan con un doble índice (n, m), que satisfacen $n \geq 0, |m| \leq n$ y $n - m$ es un número par. La expresión matemática se puede observar en la ecuación (2-7), la cual está dada en coordenadas polares (ρ, θ), donde $0 \leq \rho \leq 1$ y $0 \leq \theta \leq 2\pi$. A aparte de esto, la dependencia radial de estos polinomios, $R_n^{|m|}$ está dada por la ecuación (2-8), mientras que el factor de normalización N_n^m está dado por la ecuación (2-9).

$$Z_n^m(\rho,\theta) = \begin{cases} N_n^m R_n^{|m|}(\rho) cos(m\theta), & m \geq 0 \\ N_n^m R_n^{|m|}(\rho) sin(|m|\theta), & m < 0 \end{cases} \tag{2-7}$$

$$R_n^{|m|}(\rho) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s (n-s)!}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n-|m|}{2} - s\right)!} \rho^{n-2s} \tag{2-8}$$

$$N_n^m = \sqrt{\frac{(2-\delta_{0,m})(n+1)}{\pi}} = \sqrt{\frac{2(n+1)}{\pi(1+\delta_{0,m})}} \tag{2-9}$$

Habitualmente estos polinomios son identificados por los índices (n, m), sin embargo, también se pueden identificar a partir del índice OSA (j), el cual permite enumerar a estos polinomios. El orden radial del polinomio está dado por el índice n y la conversión entre la notación con doble índice y el índice OSA está dada por las ecuaciones (2-10), (2-11) y (2-12).

$$j = \frac{n(n+2)+m}{2} \tag{2-10}$$

$$n = \left\lceil \frac{-3+\sqrt{9+8j}}{2} \right\rceil \tag{2-11}$$

$$m = 2j - n(n+2) \tag{2-12}$$

Dado esto, en la Tabla 2-1 se puede apreciar cómo quedan descritos los polinomios de Zernike hasta de orden 4, junto con sus respectivos índices. Por otro lado, en la Figura 2-15 se muestra la representación gráfica de estos polinomios, de tal manera que se aprecia a partir de un difuminado de colores la intensidad dada por la magnitud obtenida del valor del polinomio en cada punto.

j	n	m	Polinomio de Zernike
0	0	0	1
1	1	-1	$2\rho\, sin(\theta)$
2	1	1	$2\rho\, cos(\theta)$
3	2	-2	$\sqrt{6}\rho^2\, sin(2\theta)$
4	2	0	$\sqrt{3}\,(2\rho^2-1)$
5	2	2	$\sqrt{6}\rho^2\, cos(2\theta)$
6	3	-3	$\sqrt{8}\rho^3\, sin(3\theta)$
7	3	-1	$\sqrt{8}\,(3\rho^3-2\rho)\, sin(\theta)$
8	3	1	$\sqrt{8}\,(3\rho^3-2\rho)\, cos(\theta)$
9	3	3	$\sqrt{8}\,\rho^3\, cos(3\theta)$
10	4	-4	$\sqrt{10}\rho^4\, sin(4\theta)$
11	4	-2	$\sqrt{10}\,(4\rho^4-3\rho^2)\, sin(2\theta)$
12	4	0	$\sqrt{5}\,(6\rho^4-6\rho^2+1)$
13	4	2	$\sqrt{10}\,(4\rho^4-3\rho^2)\, cos(2\theta)$
14	4	4	$\sqrt{10}\rho^4\, cos(4\theta)$

Tabla 2-1.: Polinomios de Zernike hasta n = 4.

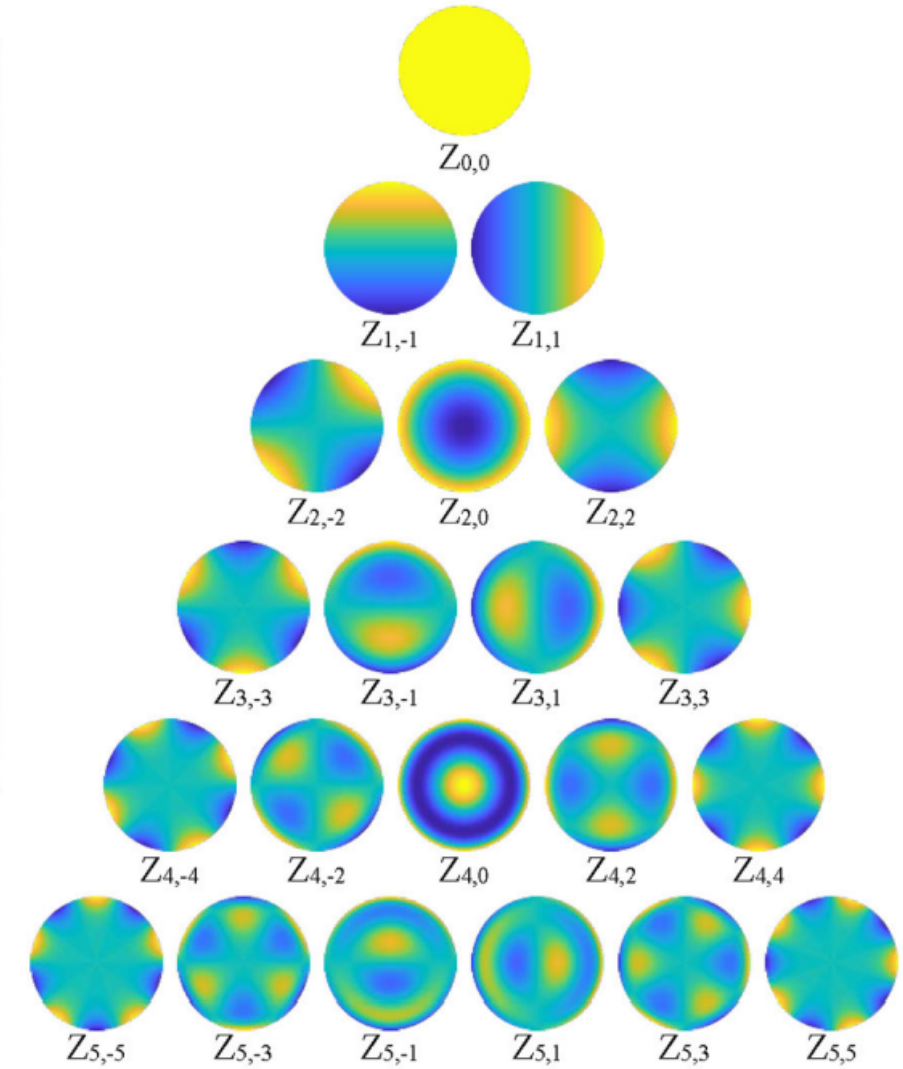


Figura 2-15.: Reparcelación gráfica de los polinomios de Zernike [Gwo et al., 2019].

A partir de esta figura se logran determinar ciertas simetrías con las que cuentan algunos polinomios, las cuales permiten entender aberraciones presentes en el ojo. Como el hecho de que los polinomios de orden 2, permiten identificar aberraciones como el astigmatismo vertical, miopía, hipermetropía y astigmatismo oblicuo y los polinomios de orden 3 y 4, brindan información de aberración esférica, coma y astigmatismo triangular [González, 2017] [Mejía, 2011]. De igual manera los polinomios de orden superior nos brindan información sobre estas y otras aberraciones. Sin embargo, los mencionados anteriormente son los principales [Martinez, 2019].

El análisis topográfico de este proyecto está basado en estos polinomios, buscando estimar y encontrar características puntuales de la simetría de las superficies, que permitan incluso

evidenciar en un análisis más profundo las aberraciones ópticas que pueden presentar. Para esto se busca representar la superficie matemáticamente como una expresión basada en la suma de los polinomios de Zernike con sus respectivos coeficientes, tal como se muestra en la ecuación 2-13, donde $W(\rho, \theta)$ es la superficie, el valor de m se da teniendo en cuenta las restricciones mencionadas anteriormente y matemáticamente $k \to \infty$, para lograr la exactitud. Sin embargo, a partir de diferentes estudios mencionados en [Mejía, 2011], se identifica que con $k = 6$ y $k = 7$, se obtienen 28 y 32 términos respectivamente. Siendo esto suficiente para describir la superficie [Mejía, 2011] [ming Dai, 2008].

$$W(\rho, \theta) = \sum_{n=0}^{k} \sum_{m} C_n^m \, Z_n^m(\rho, \theta) \tag{2-13}$$

Dentro de las características presentes al realizar el ajuste de la superficie a partir de la expresión mencionada anteriormente, está el hecho que para un sistema continuo el ajuste teniendo la ortogonalidad de los polinomios permite que los coeficientes C_n^m no dependan del valor de k. Esta característica es aprovechada para realizar el cálculo de variables como la constante cónica de la superficie, la cual puede ser escrita en términos de los coeficientes C_n^m de los primeros órdenes [Mejía et al., 2016]. En este proyecto no se cuenta con un continuo de datos, sin embargo, hay una gran cantidad de ellos a una corta distancia de uno con respecto al otro, lo que permite trabajar con una aproximación al continuo y aprovechar esta característica.

3. Planteamiento del problema y descripción de la solución

El desarrollo de la topografía de la córnea es de suma importancia a la hora de buscar posibles problemas de salud visual que evitan la buena visión. Es por esto, que centros oftalmológicos cuentan con dispositivos con la capacidad de desarrollar este tipo de mediciones, los cuales utilizan técnicas ópticas y métodos numéricos para lograr a una imagen topográfica de la córnea, donde como fue mencionado anteriormente está el topógrafo por reflexión especular de punto.

El laboratorio de óptica de la Universidad Nacional cuenta con un prototipo de este tipo de tecnología, que podría ofrecer a los estudiantes, nuevas prácticas para la comprensión del proceso óptico que es utilizado en dicha medición. Sin embargo, este instrumento fue desarrollado hace varios años, por lo que ha comenzado a presentar fallas en su funcionamiento, tanto en su software como en su hardware, y por lo tanto ha dejado de ser una herramienta útil y confiable a la hora de realizar prácticas y experimentos. Es por esto que a través del desarrollo del proyecto se desea realizar una actualización de hardware y software del equipo, que le permita tener un funcionamiento óptimo para prácticas y experimentos, por lo cual esta herramienta, a diferencia de una comercial contará con opciones adicionales de uso, haciéndola más interactiva y útil para la realización de prácticas.

Antes de comenzar con la actualización se realizó un diagnóstico del prototipo, en el que se identificó que a pesar que los elementos lumínicos funcionan, no se contaba con ningún tipo de control sobre estos, debido a que el elemento de control se encontraba obsoleto, lo que también impedía realizar una conexión con computador. Por otro lado, se contaba con una cámara que carece de conexión USB, lo que dificulta la comunicación, así que se debió reemplazar. También se evidenció que la fuente implementada para alimentar la electrónica es funcional, por lo que es utilizada en la actualización junto a los elementos lumínicos, con excepción del led usado en la mirilla, ya que este emitía una luz demasiado tenue.

A partir de lo mencionado anteriormente se propone que una vez se haya actualizado el dispositivo, cuente con las funciones y modos de uso descritas a continuación, donde se nombran partes del dispositivo que se pueden apreciar en la Figura 3-1.

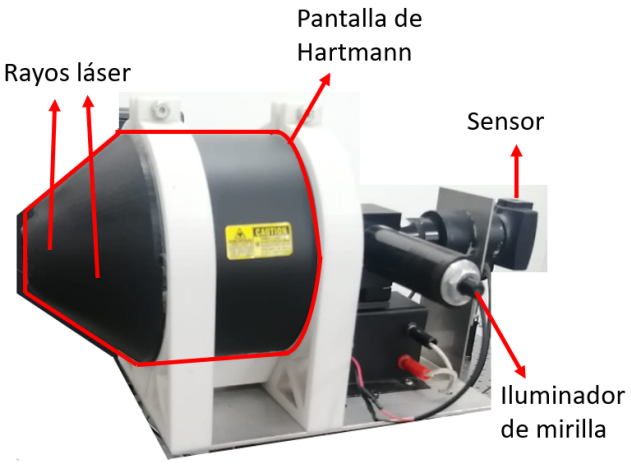


Figura 3-1.: Ubicación de los elementos luminosos y sensor del dispositivo.

3.1. Funciones

3.1.1. Encendido y apagado de rayos láser

Dado que el dispositivo cuenta con dos rayos láser que sirven como referencia para ubicar el ojo a la distancia apropiada, se tendrá la posibilidad de encenderlos o apagarlos, cuando se desee o sea necesario, esto de manera manual a través de un interruptor y dependiendo del modo en el que se esté operando el dispositivo.

3.1.2. Encendido y apagado de la cuadricula de puntos (Pantalla de Hartmann)

De igual manera se debe contar con la capacidad de encenderlos o apagarlos, cuando se desee o sea necesario, esto de manera manual a través de un interruptor y dependiendo del modo en el que se esté operando el dispositivo.

3.1.3. Encendido y apagado de la mirilla

Así como el dispositivo cuenta con una herramienta (láser) para ubicar el ojo a la distancia correcta, también cuenta con una guía para la ubicación correcta del ojo respecto al eje central del instrumento. Esta se denomina mirilla, la cual es una luz en forma de cruz que permite centrar el topógrafo con respecto al sensor y al igual que los anteriores también se debe contar con la capacidad de ser encendido o apagado cuando se desee o sea necesario,

esto de manera manual a través de un interruptor, dependiendo del modo en el que se esté operando la herramienta.

3.1.4. Captura de imágenes

El dispositivo debe contar con la capacidad de registrar imágenes y vídeos captadas por el sensor, a través de un botón y dependiendo del modo en el que se esté operando el dispositivo. Las imágenes o vídeos obtenidos contaran con una resolución de 480 pixeles x 480 pixeles y serán almacenados en el computador con el que tenga conexión, permitiéndole al usuario que se guarden en el formato de su preferencia entre: tif, jpg, png y avi.

3.1.5. Ajuste de posición manual

El dispositivo a su vez debe contar con la capacidad de ser ubicado manualmente, esto a través de una palanca que permita su desplazamiento en los tres ejes.

3.1.6. Visualización del sensor

Por medio de una conexión por cable USB a una pantalla o computador el dispositivo debe tener la capacidad de trasmitir la imagen adquirida por el sensor. Cabe resaltar que dado el sensor utilizado se debe tener un software que permita su visualización, como el que es desarrollado en este proyecto

3.2. Modos de uso

El software de control del topógrafo debe contar con la capacidad de identificar el modo de uso que desea el usuario y realizar las tareas correspondientes a este.

3.2.1. Inactivo

Cuando el dispositivo se encuentre en este modo, no permitirá ningún tipo de función.

3.2.2. Demostración

En este modo el dispositivo permitirá el encendido y apagado de la mirilla, el láser y la cuadricula de puntos, así como la visualización del sensor por medio del cable.

3.2.3. Uso por computador

Al estar en este modo, el dispositivo debe estar conectado a un computador encendido con el software ("Topógrafo corneal"), en el que se podrá realizar el control de los elementos lumínicos del topógrafo, la captura de imágenes y el análisis de las mismas.

Una vez desarrolladas estas funciones y modos de uso, se solucionará la problemática y se tendrá un dispositivo completamente funcional, interactivo y útil a la hora de realizar topografía de corneas o de superficies con características reflectivas y dimensionales parecidas a esta, como lentes de contacto.

4. Desarrollo de la solución

A partir de la descripción de la solución y los fenómenos ópticos detallados anteriormente, se desarrolla la actualización del topógrafo de córnea que como bien fue mencionado anteriormente consta de una parte de hardware y otra de software. El hardware del proyecto se basa en un microcontrolador que cuente con la capacidad de ser programado para acceder a cada uno de los modos de uso, permitiendo realizar las funciones pertinentes en cada uno, por otro lado, el software del topógrafo se divide en dos, el software del microcontrolador y el que se emplea en el computador. Ambos programas van de la mano y deben permitir una comunicación con el hardware del sistema.

De acuerdo con esto, a continuación se especifica cómo es el desarrollo de cada una de las partes de la solución, detallando el software y hardware que fueron utilizados.

4.1. Hardware

El desarrollo del hardware está basado en los elementos con los que ya contaba el topógrafo antes de la actualización, como lo son: lámpara, que se encarga de dar luz al arreglo de puntos; mirilla, que permite al usuario ubicar el ojo centrado respecto al topógrafo; y los láseres, que permiten ubicar al paciente a la distancia apropiada del topógrafo. Cada uno de estos elementos requiere una alimentación de corriente para ser encendidos, por lo que el hardware del sistema debe contar con la capacidad de controlar la alimentación de estos componentes, sumado a que el topógrafo debe poder utilizarse por medio de diferentes modos de uso, lo que hace que sea de vital importancia el uso de un microcontrolador.

Otra variable importante a la hora de construir el hardware es el diferencial de potencial eléctrico necesario para hacer funcionar cada uno de los elementos. En este proyecto se cuenta con un elemento que requiere un voltaje muy elevado, el arreglo de puntos, también llamado lampara y que requiere 2000 voltios para su funcionamiento, generando la siguiente pregunta: ¿qué tipo de circuito debe controlar el microcontrolador para que este no sufra daños?, la respuesta a esta pregunta depende de la forma en la que son alimentados los elementos y dado que el prototipo cuenta con una fuente de alimentación por separado para cada uno de ellos, se utiliza el microcontrolador como el activador del flujo de corriente de dichos elementos, lo que se conoce en electrónica como separar la parte de control a la parte de

potencia, garantizando que el microcontrolador no reciba daños por corriente o voltajes de una intensidad mayor a los que este soporta.

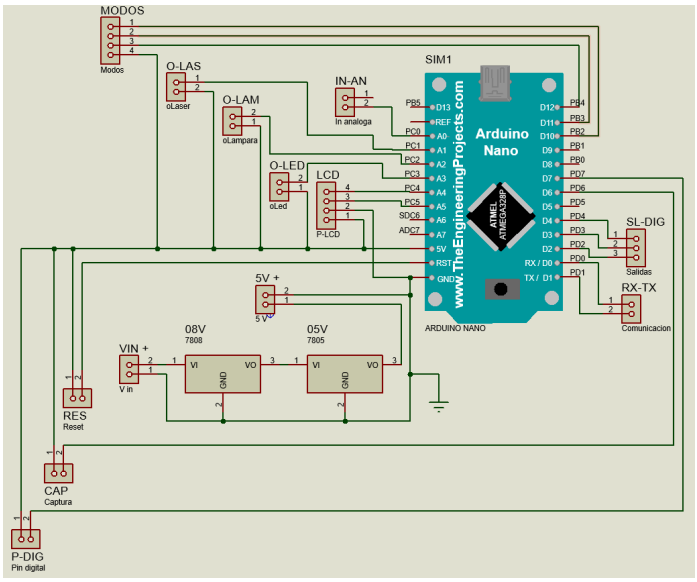
	Nombre	Entrada	Salida
1	Botón de captura	X	
2	Reset	X	
3	Modo demostración	X	
4	Modo computador	X	
5	Láseres		X
6	Lampara		X
7	Mirilla		X
8	Interruptor de láseres	X	
9	Interruptor de Lampara	X	
10	Interruptor de Mirilla	X	

Tabla 4-1.: Conexiones de entrada y salida del microcontrolador.

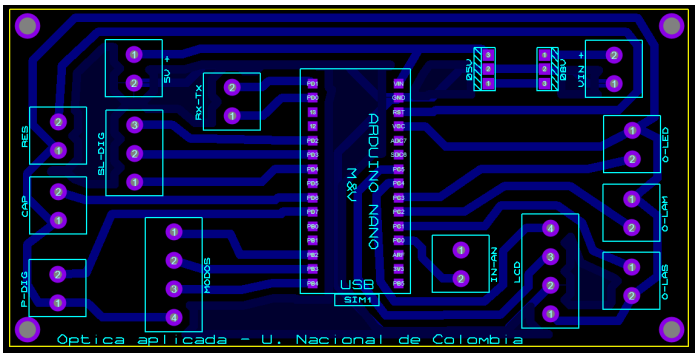
Es importante hacer énfasis en que el microcontrolador va a contar con salidas digitales para controlar el encendido y apagado de los elementos lumínicos del topógrafo y a su vez debe analizar información de entrada dada por: los interruptores encargados de encender y apagar los componentes, el interruptor encargado de determinar el modo en el que se va a utilizar el equipo y el botón encargado de enviar la señal para la captura de imágenes o videos. A partir de esto, se realiza la Tabla 4-1, donde se describen las señales de entrada y salida con las que debe contar el microcontrolador, en dicha tabla es posible evidenciar que el número de señales de entrada y de salida al microcontrolador, no son demasiadas: 7 entradas y 3 salidas, por lo que se decide utilizar el Arduino nano como componente principal del hardware. Esto teniendo en cuenta que este microcontrolador cuenta con la capacidad necesaria para el manejo del topógrafo, y a su vez goza con una facilidad en su programación y comunicación a computador por medio de Matlab y otros softwares, además es de bajo costo (35.000 pesos).

Teniendo en cuenta lo anterior, se desarrolla el circuito en el que se ubica el Arduino nano, el cual es un circuito impreso diseñado a través del software Proteus, y se puede observar en la Figura 4-1, en donde se dejan expresadas las conexiones para las cuales son soldadas diferentes tipos de borneras. A parte de las conexiones necesarias, el circuito cuenta con la posibilidad de añadir más conexiones a entradas digitales y análogas del microcontrolador, así como la capacidad de uso de los puertos de trasmisión y recepción de datos, donde estas conexiones fueron habilitadas con el fin de que puedan ser utilizadas en futuras actualizaciones donde se presente un aumento funciones del topógrafo. La alimentación eléctrica del topógrafo en general se realiza a través de su conexión a un tomacorriente, que brinda 120

voltios encargados de alimentar la lámpara después de pasar por un transformador de voltaje “POLI NEON”, el cual elevará el voltaje hasta el requerido por la lámpara. Aparte de esto, se utilizan los 120 voltios del tomacorriente mediante una conexión en paralelo, para la alimentación de una fuente de voltaje que consta de un transformador y elementos rectificadores que permite obtener como salida 12 voltios directos, con los cuales se realiza la alimentación al Arduino que es reducida a 8 voltios y posteriormente a 5 voltios para disipar la potencia en diferentes componentes y evitar que sea uno solo el encargado de regular el voltaje. A parte de esto, el circuito tiene una capa de polo a tierra que cuenta con la función de eliminar corrientes parásitas y disminuir el ruido eléctrico que pueda presentar el circuito y con un botón de “reset” encargado de reiniciar el programa en caso de ser necesario. Dicho circuito con sus componentes se puede apreciar en la Figura 4-2.



(a) Diagrama del circuito desarrollado en el software proteus.



(b) Diagrama del circuito impreso desarrollado en el software Proteus.

Figura 4-1.: Plano del circuito desarrollado en el software Proteus.

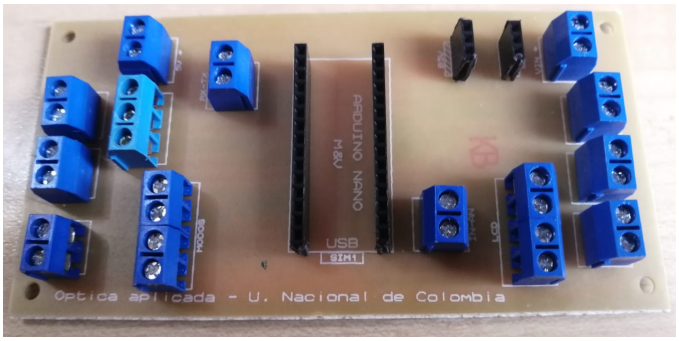
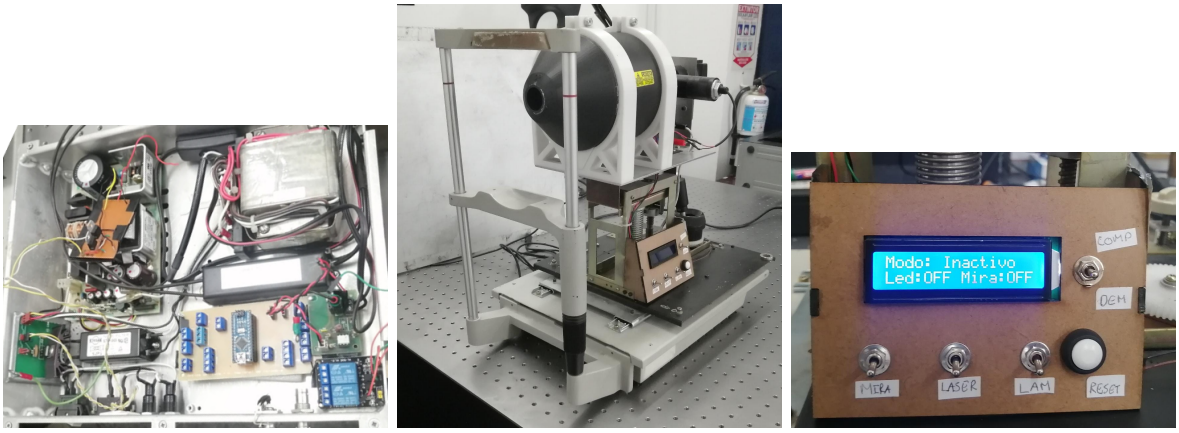


Figura 4-2.: Circuito físico con sus componentes soldadas.

El complemento del circuito, compuesto por interruptores, botones y una pantalla LCD, son colocados en un panel el cual está a un costado del dispositivo y le permite al usuario manipularla e identificar errores o el funcionamiento de la máquina. Por temas de comodidad, el botón encargado de hacer la captura de imagen se encuentra en la palanca que permite mover la estructura del topógrafo, de esta manera al ubicarla se podrá capturar la imagen inmediatamente. El hardware del topógrafo se puede apreciar en la Figura 4-3.



(a) Electrónica del topógrafo. (b) Topógrafo. (c) Panel del topógrafo.

Figura 4-3.: Hardware del topógrafo.

Los detalles de cómo son conectados los interruptores de entrada, así como la alimentación de los sistemas lumínicos del topógrafo se pueden apreciar en el Anexo A.

El diseño óptico del topógrafo se puede apreciar en el Anexo D, en el cual se ven las distancias, lentes utilizadas y demás componentes que permiten obtener la imagen adecuada. Sin embargo, aparte de lo mostrado en dicho anexo el sistema también cuenta con los rayos láser, los cuales como ya fue mencionado anteriormente sirven de referencia para el usuario

como indicador de la distancia entre la lente o córnea y el topógrafo. El caso ideal y como fue diseñado la alineación de estos láseres se puede apreciar en la Figura 4-4, donde "d" representa la distancia óptima para la captura de una imagen a analizar. A pesar de esto, al poner en funcionamiento los rayos láser y verificar su alineación se observa que no se comportan como deberían, lo que se aprecia en la Figura 4-5, donde se muestran dos imágenes capturadas del patrón de puntos y de los rayos láser a diferentes distancias del topógrafo.

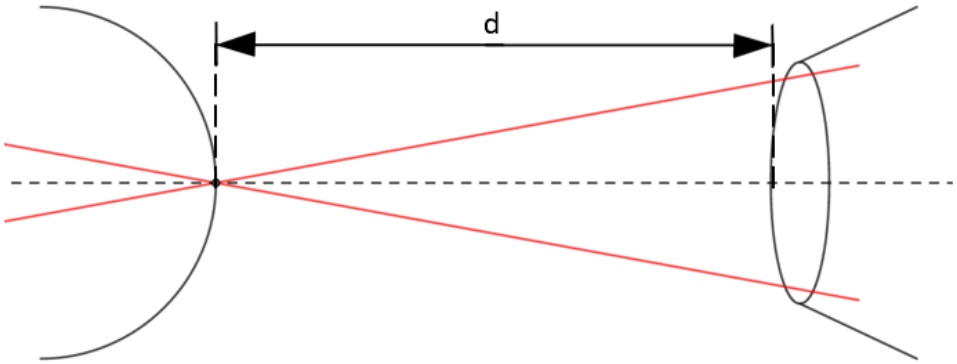
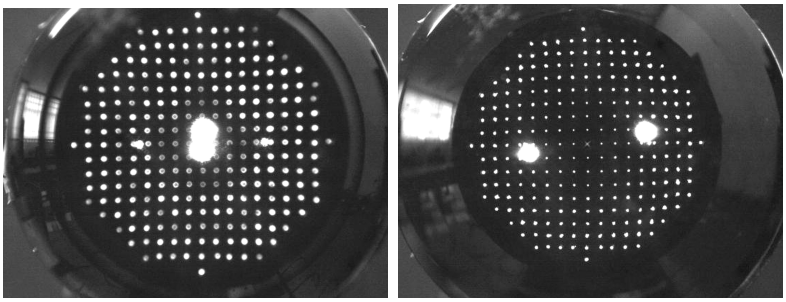


Figura 4-4.: Alineación ideal de los rayos láser para obtener una imagen enfocada.



(a) Imagen desenfocada. (b) Imagen enfocada.

Figura 4-5.: Alineación de los rayos láser en la captura de imágenes. Donde los círculos mas grandes y luminosos, son obtenidos por la reflexión de los rayos láser

La Figura 4-5 es tomada teniendo como objeto de medida una superficie cónica con una excentricidad de 0.45, y en la Figura 4-5-a se ve cómo los láser no logran una alineación perfecta, dado que se cuenta con una separación vertical de los centros de los rayos láser y por otro lado el patrón de puntos se ve desenfocado lo que es indicativo de que la alineación no es correcta. Para ver el margen de error se compara esta imagen con la de la Figura 4-5-b,

donde se ve la posición de los rayos láser cuando el patrón de puntos está enfocado. El corregir esta desalineación representaría un trabajo adicional el cual conlleva desmontar todo el topógrafo, por lo que antes de realizarlo se evalúa la posibilidad de cumplir los objetivos planteados para este proyecto sin la necesidad de dicha corrección, lo que es posible dado que los rayos láser no son encendidos al momento de capturar la imagen a analizar. Del mismo modo se identifica que el punto central del patrón, la mirilla, cuenta con muy poca visibilidad por lo que se cambió el elemento lumínico que producía esta imagen, pasando de un led convencional a una lámpara de 12 voltios, teniendo en cuenta que la luz no sea la suficiente para generar daños oculares en un posible usuario, siendo el limite de irradiancia máximo comúnmente aceptado para no causar daños oculares de $10mW/cm^2$ [OSHA, 2015]. A partir de este cambio y el ajuste entre las distancias de los elementos, permiten llevar la imagen del punto S al punto S' en el plano ubicado en el Anexo D, lo que permite asegurar que el usuario cuente con una buena visibilidad de la mirilla y que al estar enfocada garantice que el resto del patrón también lo esté. Con este enfoque es posible hallar la distancia "d" que aparece en la Figura 4-4.

4.2. Software de Arduino

El Arduino nano es el microprocesador del topógrafo y elemento capaz de interpretar información de entrada para controlar el funcionamiento de hardware. Es importante resaltar el hecho que el controlador del topógrafo "Arduino nano" cuenta con dos tipos de señales de entrada. Si el dispositivo se está utilizando en modo "demostración", las señales de entrada están dadas por interruptores que dejan pasar o no una señal eléctrica, mientras que en modo "uso por computador" las señales de entrada no son eléctricas sino dadas por el software del computador. Debido a esto, el software del Arduino debe poder recibir e interpretar cada tipo de señal de entrada.

A partir de las características mencionadas anteriormente se desarrolla el software para el Arduino siguiendo el orden de programación tradicional, lo primero que se realiza es declarar las variables, donde se ven incluidos los puertos de entrada y salida, tanto digitales como análogos. Seguido de esto declarar qué puertos van a ser utilizados como entradas y salidas, así como la inicialización de la comunicación con el computador, las librerías y variables necesarias para el uso de la pantalla LCD y demás herramientas. Lo siguiente es establecer las respectivas condiciones para poder identificar en qué modo de uso se encuentra el topógrafo y dependiendo de este, interpretar las posibles señales de entrada que tendrá el Arduino. Esto teniendo en cuenta que en caso de estar en modo "uso por computador" será el computador el encargado de enviar las señales al Arduino, las cuales serán recibidas como variables tipo "Char", mientras que cuando el topógrafo se encuentre en "modo demostración" las señales serán enviadas a través de interruptores manuales que serán utilizadas por el usuario.

A parte de esto en el "modo computador", también se cuenta con comunicación en sentido Arduino al computador, para lo que se aprovecha el hecho de que el computador cuenta con la capacidad de leer el puerto serial del Arduino, haciendo que este imprima la información a trasmitir en el puerto serial. En este caso, la información a trasmitir depende de la señal de entrada dada por el botón de captura y puesto a que la captura de imagen o video se realiza a través del computador, es necesario enviarle esta señal. Otra función añadida en este software es limitar el tiempo de duración de encendido de la lámpara, ya que esta es capaz de disipar calor después de un tiempo extendido de funcionamiento, el cual puede generar inconformidades a la hora de seguir realizando practicas o desajustes en el sistema. Para esto se utilizan funciones temporales con las que cuenta el Arduino que permiten almacenar la información temporal de cierto suceso, para luego compararla y de esta manera identificar el tiempo que lleva encendida la lampara.

El código comentado y completo que fue desarrollado e implementado en el Arduino del topógrafo se puede observar en el Anexo B, donde se evidencia la declaración de variables, uso de librerías, condiciones de identificación y el uso señales de entrada y salida requeridas para el funcionamiento de cada modo de uso.

4.3. Software de control por computador

Cuando el topógrafo se esté utilizado en el modo "uso por computador", realizará las funciones de acuerdo con lo que el usuario señale a través del computador. Es por esto que se diseña una interfaz gráfica que cuenta con una pantalla inicial que permite al usuario indicar si desea realizar un control sobre las funciones del topógrafo, la captura o análisis de una imagen o video a través del mismo. Esta interfaz se puede apreciar en la Figura 4-6.

Por temas de compatibilidad, licencias y facilidad a la hora de realizar la conexión con el computador, la interfaz se desarrolla en la herramienta "App Designer" del software Matlab con licencia de estudiante otorgada por la universidad. Como se aprecia en la Figura 4-6, la interfaz cuenta con varias ventanas que ofrecen la posibilidad de navegar a través de ellas, con el fin de brindarle al usuario la facilidad para realizar la tarea que desee. Esta interfaz gráfica cuenta con las características normales de un programa, por lo que puede ser instalada en cualquier computador que cumpla con las condiciones de sistema operativo, en este caso, el computador al que se le va a instalar debe contar con Windows de 64 bits y no sera necesaria ninguna licencia o software adicional. Una vez el programa sea instalado, se podrá usar sin restricciones, donde al abrirlo se despliega la pantalla de inicio Figura 4-6-a. Esta pantalla permite redirigir hacia la pantalla con las funciones que el usuario desee hacer, las cuales se dividen en tres: captura, control, y análisis, para lo que es importante resaltar que

(a) Ventana de inicio.

(b) Ventana de control.

(c) Ventana de captura.

(d) Ventana de análisis.

Figura 4-6.: Ventanas de la interfaz.

para el uso de la ventana de control y de captura es necesario que el topógrafo esté conectado. Aparte de esto, en la ventana de captura la cámara también debe estar conectada. Por otro lado, las funciones en la ventana de análisis no cuentan con ningún tipo de restricción de conexión, por lo que es posible usarlas aún sin tener el topógrafo o la cámara conectadas. Sin embargo, se debe contar con el archivo de video o imagen a analizar.

De manera más específica, a continuación se habla del funcionamiento de cada una de las ventanas. Cabe aclarar que, en el Anexo C, se encuentra el código desarrollado para cada una de las ellas.

4.4. Ventana de control

En la ventana de control, Figura 4-6-b, se cuenta un interruptor por cada uno de los elementos lumínicos del topógrafo: lámpara, mirilla y láseres. Estos interruptores tienen dos estados posibles, "on" y "off", y es cuando el usuario cambie dicho estado, que el software envía una señal distintiva del interruptor y estado del mismo al Arduino. De esta manera el microcontrolador recibe la señal y, en consecuencia, activa o desactiva la salida que da control a cada elemento del topógrafo. Esto ocurre siempre y cuando se tenga conexión con el topógrafo, lo cual se evidencia a través de la "luz" llamada comunicación: cuando esta esté en verde indicara que hay conexión, mientras que, si esta se ve de color rojo, significa que no hay conexión con el topógrafo. La ventana a su vez cuenta con botones de conectar y desconectar, esto para que el usuario tenga total control sobre la conexión con el topógrafo.

4.5. Ventana de captura

En la ventana de captura, Figura 4-6-c, se aprecia directamente un espacio en el cual se puede iniciar la visualización de la cámara del topógrafo, en la misma se ubican unas líneas de referencia, las cuales no serán visualizadas en la imagen o video capturado, pero sirven de referencia al usuario a la hora de tomar la imagen. Dado que es posible hacer dos tipos de captura, el usuario deberá indicar si desea capturar una imagen o un video, para seguido de esto utilizar el botón "captura" de la interfaz o interruptor ubicado en la manija del topógrafo, como activador de la toma de imagen o video, haciendo énfasis en que si se desea capturar un video deberá mantener el interruptor presionado durante el tiempo que desee que dure el video. Una vez la imagen o video sea tomado se desplegará una ventana en la cual se podrá guardar el archivo en la carpeta de preferencia y el usuario podrá seguir realizando capturas, las cuales podrán ser analizadas a través de esta ventana.

4.6. Ventana de análisis

La ventana de análisis, Figura 4-6-d, es usada para el estudio de imágenes, en este caso la interfaz le permitirá seleccionar una imagen o video entre los archivos de la computadora. En caso que el archivo sea un video, esta interfaz le permitirá al usuario adelantarlo o atrasarlo "frame" a "frame", esto con el fin de ubicar la imagen del vídeo que se quiera analizar, esta imagen podrá ser guardada en el computador. Una vez identificada la imagen, el usuario podrá pulsar el botón "analizar" el cual permite que se realice el respectivo estudio de la figura, el cual se basa en la ejecución de un algoritmo en el cual se utilizan métodos numéricos para encontrar la ecuación de la superficie reflectora (la córnea).

Figura 4-7.: Esquema vectorial de la reflexión de un punto de luz en la córnea.

El algoritmo utilizado fue desarrollado inicialmente por el profesor Yobani Mejía [Mejía, 2017]. En este, se utiliza el hecho de que se pueden expresar las trayectorias de cada uno de los haces de luz teniendo en cuenta el trazado de rayos de luz (Figura 4-7), en el cual se puede identificar el vector de haz incidente, reflejado y normal a la superficie. A partir de dichos vectores, se puede identificar la relación que tienen entre sí, dada por la ecuación 4-1, donde $\widehat{\mathbf{R}}$ es el vector de reflexión, $\widehat{\mathbf{I}}$ el vector incidente y $\widehat{\mathbf{N}}$ el vector normal a la superficie. También se resalta que estos dependen de la superficie denotada con "Z" o "f(x,y)", al ser el punto Q el lugar de origen de cada uno de los vectores y a la vez la intercepción de estos con la superficie. Teniendo en cuenta esta relación entre los vectores y la superficie, es posible desarrollar matemáticamente una expresión que permita hallar la ubicación de los puntos en la superficie. Para esto se toma como primera consideración el hecho de que se conoce la ubicación del punto P, el cual corresponde a un punto luminoso en la pantalla de Hartmann. La ecuación 4-2, expresa la superficie "$f(x, y)$" como una superficie de nivel, de la cual es posible hallar su gradiente. A partir de propiedades de cálculo vectorial podemos decir que el gradiente es perpendicular a la superficie de nivel y por lo tanto paralelo al vector normal a esta [Stewart, 2012].

$$\widehat{\mathbf{R}} - \widehat{\mathbf{I}} = \mathbf{N} \tag{4-1}$$

$$g(x, y, z) = z - f(x, y) \tag{4-2}$$

Dada la relación de paralelismo entre el vector normal $\widehat{\mathbf{N}}$ y el gradiente de la superficie de nivel (ecuación 4-3), estos se pueden relacionar de acuerdo con la ecuación 4-4. Teniendo en cuenta la ecuación 4-1 se llega a la ecuación 4-5, que al desarrollarla tiene la forma dada por la ecuación 4-6.

$$\nabla g = \left\{ -\frac{\partial f}{\partial x}, -\frac{\partial f}{\partial y}, 1 \right\} \tag{4-3}$$

$$\mathbf{N} \times \nabla g = 0 \tag{4-4}$$

$$(\widehat{\mathbf{R}} - \widehat{\mathbf{I}}) \times \nabla g = 0 \tag{4-5}$$

$$\begin{aligned} \widehat{\mathbf{i}} \left[(R_y - I_y) + (R_z - I_z)\frac{\partial f}{\partial y} \right] - \widehat{\mathbf{j}} \left[(R_x - I_x) + (R_z - I_z)\frac{\partial f}{\partial x} \right] \\ + \widehat{\mathbf{k}} \left[-(R_x - I_x)\frac{\partial f}{\partial y} + (R_y - I_y)\frac{\partial f}{\partial x} \right] = 0 \end{aligned} \tag{4-6}$$

Por otra parte, es posible realizar una aproximación de paralelismo entre el rayo reflejado y el eje óptico, esto debido a las dimensiones entre la distancia del punto Q al eje óptico y al sistema óptico, el cual se ubica en L en la Figura 4-7, por lo que se dice que el vector $\widehat{\mathbf{R}}$ solo cuenta con componente a lo largo del eje Z, y teniendo en cuenta este como vector unitario se llega a la simplificación de la ecuación 4-7, en la cual podemos igualar a cero cada uno de los términos y así extraer las ecuaciones 4-8 y 4-9, de los dos primeros términos. Y es a partir de estos que se realiza una integración para obtener la superficie de interés "$f(x, y)$".

$$\widehat{\mathbf{i}} \left[(-I_y) + (1 - I_z)\frac{\partial f}{\partial y} \right] - \widehat{\mathbf{j}} \left[(-I_x) + (1 - I_z)\frac{\partial f}{\partial x} \right] + \widehat{\mathbf{k}} \left[(I_x)\frac{\partial f}{\partial y} + (-I_y)\frac{\partial f}{\partial x} \right] = 0 \tag{4-7}$$

$$\frac{\partial f}{\partial x} = \frac{I_x}{1 - I_z} \tag{4-8}$$

$$\frac{\partial f}{\partial y} = \frac{I_y}{1 - I_z} \tag{4-9}$$

A partir del proceso descrito anteriormente, se desarrolla el software de análisis el cual debe tener en cuenta cada uno de los puntos y una condición de error, permitiendo que el desarrollo de la reconstrucción topográfica sea un proceso iterativo, donde la condición de error depende de los parámetros "Delta Z" y "RMS - Rv". De estos se esperan valores cercanos a cero, milésimas y micras respectivamente, ya que dan información de que tan alejados están los valores representados por la superficie obtenida respecto a los valores nominales. Sin embargo, de no cumplirse con esta aproximación, el proceso parará cuando estos valores

no presenten un cambio mayor al 5 % entre iteraciones. Estos y otros parámetros se exponen más adelante en el documento.

El diagrama de este proceso se puede apreciar en la Figura 4-8, el cual da como resultado la topografía a través de un polinomio. Este es un polinomio compuesto por polinomios de Zernike que dependiendo de la imagen a analizar, puede ser de un orden superior como el obtenido con k = 7 en la ecuación 2-13, esto dado que las aberraciones que se puede presentar en el sistema pueden ser descritas por polinomios de diferentes grados cercanos o inferiores al séptimo [González, 2017], como por ejemplo la aberración de astigmatismo se puede describir con polinomios que van desde segundo orden [en F. Luis Gabriel Valdivieso González, 2014] hasta sexto orden [Olarte, 2011], algo similar ocurre con la aberración esférica [Mejía, 2011], tal fue mencionado en el capítulo 2.

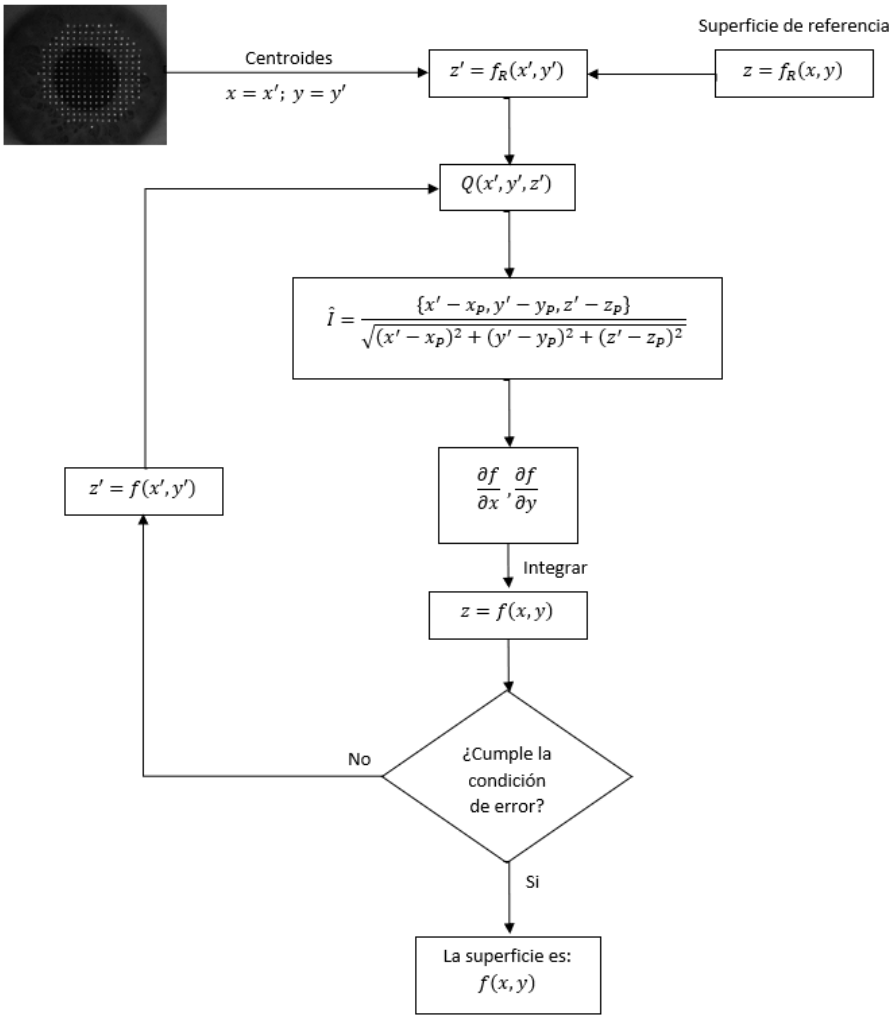


Figura 4-8.: Diagrama de flujo del proceso de reconstrucción topográfica de la córnea.

Una vez identificada la superficie de la córnea, es posible realizar mapas de elevación y de curvatura, los cuales son realizados por el software, ya que son de gran ayuda para el usuario del topógrafo a la hora de entender la superficie de la córnea. El proceso por el cual se desarrollan dichas imágenes junto con el código implicado para el envío de las señales al Arduino se puede apreciar con detalle en el Anexo C. Cabe mencionar que matemáticamente las aberraciones que se presentan en sistemas oculares suelen ser descritas en coordenadas polares por los polinomios de Zernike [Recarte, 2017].

A parte de esto es importante resaltar el hecho de que para que el método funcione se debe tener en cuenta la siguiente información: ubicación física de los puntos de iluminación de la pantalla de Hartmman, distancia del topógrafo al elemento a medir y el distanciamiento entre puntos de la imagen generada por la pantalla de Hartman de una superficie esférica, dada en pixeles. La tercera magnitud depende de la resolución de la cámara o sensor que se esté usando. En el prototipo se tenía un sensor "WAT - 902C", que permite obtener una imagen útil con una resolución de 480x480 pixeles, la cual por temas de manejo, resolución y acoplamiento con el software fue remplazado por un sensor "Thor Labs C1285R12M", esta permite obtener la imagen con una resolución de 1024x1280. Tomando en cuenta esta información se hace la respectiva medida del espaciado entre puntos en la imagen de referencia, dicha imagen se obtiene a partir del montaje que se muestra en la Figura 4-9, en la cual se observa la superficie de referencia, una esfera de radio 7.78 mm.

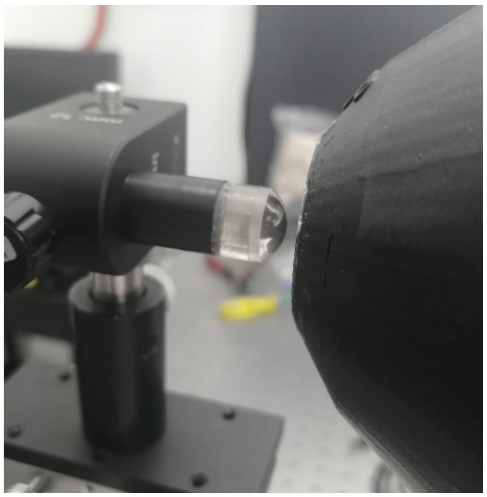


Figura 4-9.: Montaje para la toma de imagen de referencia, medida de una esfera de radio 7.78 mm.

Esto conduce a la toma de la imagen que se aprecia en la Figura 4-10, imagen de referencia, y como se observa en esta figura el resultado que se obtiene es el de una cuadricula de puntos con simetría en el espaciado entre puntos, la cual es analizada a través del programa

ThorCam, un software desarrollado por la empresa "Thorlabs, Inc" [Thorlabs, 2014], que permite extraer la información de distanciamiento entre puntos del sistema.

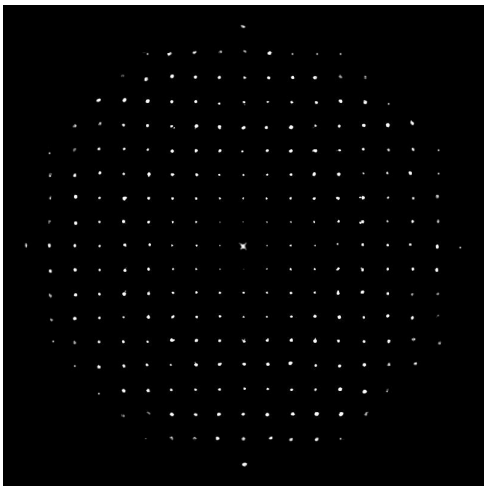


Figura 4-10.: Imagen formada con la superficie de referencia.

Con lo mencionado anteriormente, se cuenta con toda la información necesaria para desarrollar el análisis topográfico de la imagen, dicha información se encuentra almacenada para el uso del software de análisis y este cuenta con la capacidad de brindar como salida para el usuario un mapa de elevación, el cual está referenciado sobre una superficie esférica cuyo radio es brindado por el programa y un mapa de curvatura.

La ventana, Figura 4-6-d, cuenta con el objetivo de realizar un primer filtro de la imagen a analizar, permitiendo que esta pueda ser extraída de un video, escalada o cambiada de formato, para una vez elegida el usuario pulse el botón "Analizar", el cual desplegará la ventana que se aprecia en la Figura 4-11, donde se cargará la imagen previamente escogida, el desarrollo inicial de esta interfaz fue hecha por el profesor Yobani Mejía y dado los cambios presentados durante la actualización del topógrafo se le realizaron cambios para su adaptación, los cuales van desde el acople con las otras ventanas hasta algunos datos requeridos para el análisis como el distanciamiento de puntos en pixeles y el tamaño de la imagen.

Al interior de esta interfaz encontramos diferentes valores y funciones, dentro de los cuales se encuentra una lista de opciones en color amarillo, dicha lista permite seleccionar uno de seis filtros posibles a realizar a la imagen, esto en dado caso que por efectos luminosos o dificultades a la hora de capturar la imagen esta no haya quedado del todo bien. A continuación, se describen los filtros:

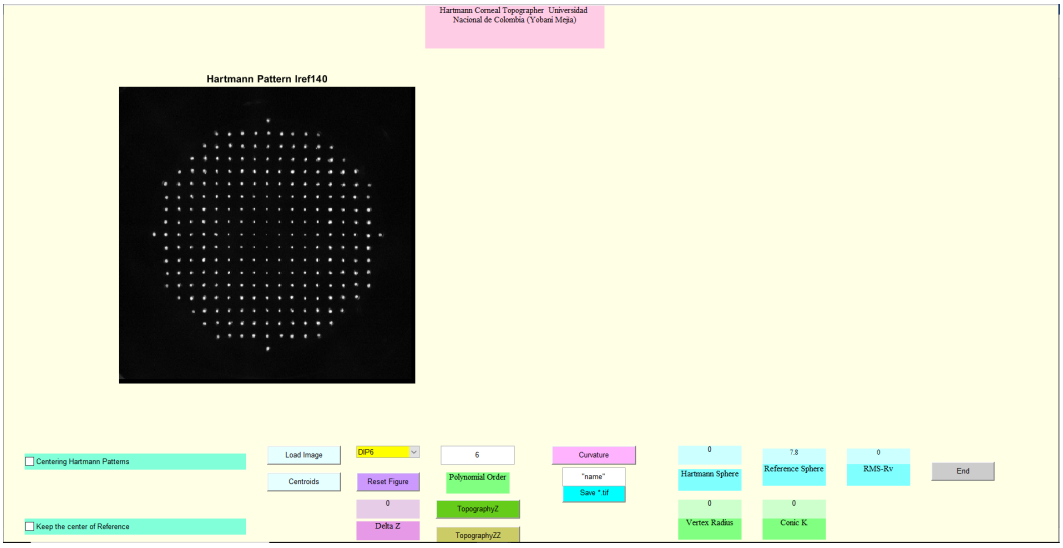


Figura 4-11.: Interfaz de análisis.

- DIP1: Este filtro permite extraer una región circular de la imagen a través de la ubicación de dos puntos dentro de la misma, donde el primer punto es el centro de la circunferencia y el otro es un punto sobre la circunferencia.
- DIP2: Este es un filtro de mediana de la imagen, lo que reduce la variación de intensidad entre pixeles cercanos, a partir de reasignar el valor de cada pixel teniendo en cuenta la mediana de una vecindad de 3x3 alrededor de cada pixel.
- DIP3: Este filtro permite adicionar un punto en caso que el usuario desee resaltar algún punto que no es claro en la imagen.
- DIP4: Este filtro permite eliminar un punto en caso que el usuario desee eliminar algún punto que haya aparecido producto de algún error.
- DIP5: Este filtro permite que el usuario filtre pixeles de la imagen con cierta intensidad de luz a través de un histograma que el programa hace presente, donde se evidencia la cantidad de pixeles que cuentan de cada intensidad lumínica. Esto teniendo en cuenta que el sensor de la imagen cuenta con un escalado de intensidad por pixel de 0 a 255.
- DIP6: Este filtro permite al usuario ubicar el centro de la imagen al seleccionar un punto sobre ella.

El trabajo realizado por estos filtros se puede eliminar al utilizar el botón “Reset Figure”, el cual vuelve la imagen a su estado original. A parte de dichos filtros, la interfaz cuenta con los botones “Load Image” y “Centroids”, los cuales sirven para cargar una nueva imagen y ubicar los centroides de los puntos del arreglo respectivamente. Al lado izquierdo de estos

botones se encuentran las posibles opciones a marcar, las cuales cuentan con el objetivo de ubicar y mantener la imagen centrada si es que así se desea, teniendo en cuenta que para realizar el análisis se requiere que el patrón esté centrado. Para realizar el análisis de la figura se tienen los botones "TopographyZ" y "TopographyZZ", el primero indica el resultado después de haber realizado la primera iteración del algoritmo planteado en la figura 4-8, mientras que el segundo realiza iteraciones sucesivas hasta cumplir la condición de error mencionada anteriormente. En ambos casos se tiene en cuenta el valor indicado el campo "Polynomial Orden", el cual es orden del polinomio sobre el cual se va a realizar el ajuste de la superficie medida. Ambos botones de análisis darán como resultado el mapa de elevación de la superficie en contraste con una superficie esférica de referencia. A parte de esto se cuenta con el botón "Curvature", el cual genera un mapa de curvatura meridional de la superficie, permitiendo al usuario identificar las zonas de igual curvatura.

Por otro lado, la interfaz de análisis arroja las siguientes metricas como resultado: Hartmann Sphere, Vertex Radio, Conic k, RMS-Rv y Delta Z. El valor "Hartmann Sphere", es el radio de la esfera que genera la imagen de los cuatro puntos más próximos a la mirilla. El valor dado por "Vertex Radius", equivale al radio de la esfera que mejor ajusta la superficie reconstruida en el vértice. El valor de "Conic K" representa la constante cónica obtenida de la superficie hallada, la cual permite interpretar que tan esférica es la misma. El valor "RMS-Rv" corresponde al cálculo de la desviación promedio de los puntos respecto a la superficie, lo cual sirve como un indicador de error. Finalmente, el otro valor que arroja la interfaz es el llamado "Delta Z", el cual es la diferencia entre la posición del vértice en Z real de la superficie medida con el valor correspondiente al de la superficie obtenida, este valor también sirve como indicativo de error, puesto que al ser la superficie calculada correspondiente con la superficie real este valor debe ser cero. Los datos antes mencionados cuentan con unidades de milímetros a excepción de "Conic K" y "RMS-Rv" y permiten al usuario del topógrafo dar claridad de la superficie obtenida, permitiéndole al mismo sacar conclusiones de dicha superficie.

Las métricas mencionadas anteriormente, se aprecian a detalle en la Tabla 4-2.

Métrica	Unidad	Símbolo	Calculo	Descripción
Delta Z	mm	Δ Z	Dada la ubicación del vértice de la superficie recreada y la ubicación real del vértice.	Diferencia entre la posición del vértice en Z real y el de la superficie obtenida.
Hartmann Sphere	mm	Hartmann Sphere	Usando la ubicación real de los puntos dado por la superficie de referencia y la superficie a analizar. Se utiliza la aproximación paraxial y las ecuaciones de aumento en la imagen, para calcular el radio de la esfera.	Radio de la esfera que genera la imagen de los cuatro puntos más próximos a la mirilla.
Vertex Radius	mm	Rv	Con base en la ubicación del vértice dado por la superficie de referencia y la superficie a analizar. Se utiliza la aproximación paraxial y las ecuaciones de aumento en la imagen, para calcular el radio de la esfera.	Radio de la esfera que mejor ajusta la superficie reconstruida en el vértice.
Conic K	-	K	Se da a partir de los coeficientes C_n^m obtenidos de la reconstrucción de la superficie.	Constante cónica obtenida de la superficie.
Root mean square-Rv	mm	RMS-Rv	Se identifica la diferencia entre la ubicación de los puntos de la figura imagen y los dados por la superficie obtenida. A partir de estas diferencias se calcula la raíz cuadrada media.	Desviación promedio de los puntos respecto a la superficie.

Tabla 4-2.: Métricas y sus características.

5. Toma de topografía

Para comprobar el buen funcionamiento del topógrafo y como complemento a la documentación del equipo, se realiza la medida topográfica de piezas convexas que son utilizadas como moldes para la creación de lentes de contacto. Estas piezas están hechas en PMMA (Polymetilmaetacrliato) y ya están caracterizadas, lo que permite tener una comparación de las medidas obtenidas con las medidas teóricas. Para esto se requiere de una base donde la pieza a medir pueda ubicarse de manera frontal al dispositivo, simulando a un ojo. Dicha montura se aprecia en la Figura 5-1 y cuenta con los grados de libertad necesarios para mover de manera precisa la pieza, axialmente y transversalmente. Seguido de esto, se utiliza el dispositivo ya sea en modo demostración o en modo computadora para poder utilizar las funciones de encendido y apagado de los láser y la mirilla, y con esto asegurar que la ubicación de la pieza sea correcta. A partir de esto, se utiliza el arreglo de puntos o pantalla de Hartman y procede a capturar la imagen, que posteriormente con ayuda del Software desarrollado en Matlab se realiza el respectivo análisis.

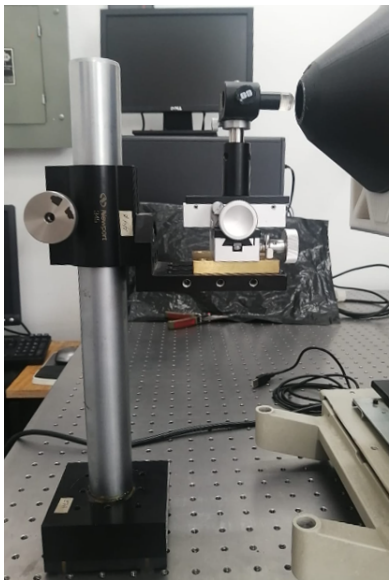


Figura 5-1.: Montaje para la medida topográfica de las piezas.

Como fue mencionado anteriormente la calibración del software de análisis requiere de la

medida de una imagen de referencia, la cual se observa en la Figura 4-10, lo que permite fijar la ubicación del centro de la imagen en la cámara, el cual es el punto en el que se ubica la "mira" y sirve de referencia para la captura de las imágenes como se observa en la Figura 5-2, donde el tamaño de la imagen ya ha sido calibrado para que el usuario pueda ver la zona útil adquirida por el sensor.

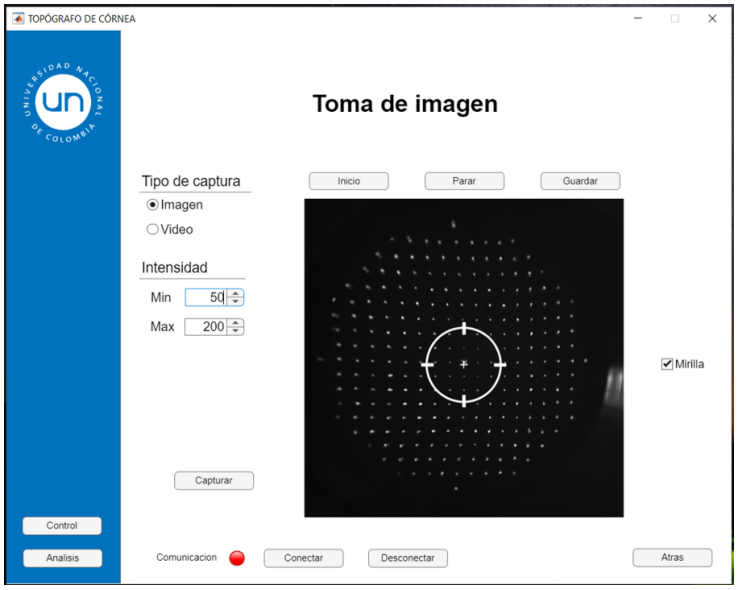
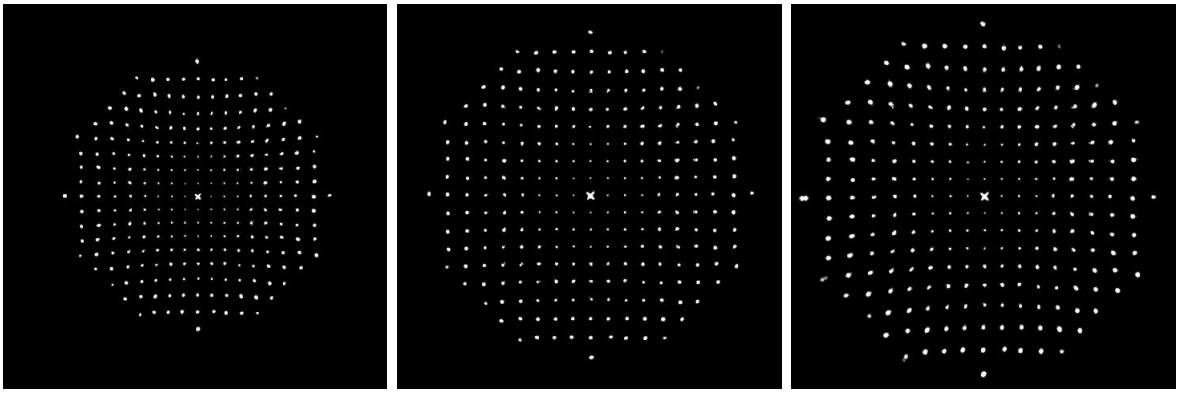


Figura 5-2.: Interfaz para la captura de imágenes con "mira" activa.

A partir de esto, se realiza la captura de las piezas que se aprecian en la Figura 5-3, los cuales son superficies cónicas que pueden identificar a partir de dos parámetros, el radio "R" y la excentricidad "e" o la constante cónica "K". Como se puede apreciar en esta figura, son tres piezas cónicas, donde al identificarlas de izquierda a derecha, es posible decir que sus radios de curvatura son de 8.5 mm, 8.5 mm y 6.5 mm respectivamente, y sus excentricidades son de 1, 0.45 y 1 respectivamente.

Para garantizar una buena imagen topográfica se realiza la captura de un video por cada una de las superficies, del cual podemos extraer posteriormente el "frame" o imagen adecuada para el análisis, esta se seleccionará teniendo en cuenta dos criterios, que esté centrada con la mira y que cuente con un buen enfoque. Esta imagen se identifica de manera objetiva, ubicando su centro a partir de la mirilla centrada con la que cuenta la interfaz y que se puede apreciar en la Figura 5-2, mientras que el mejor enfoque se identifica cuando cada uno de los puntos cuenta con la menor distorsión, es decir, cuentan con una forma circular y son de menor tamaño A partir de este proceso se llega a las imágenes de la Figura 5-4.

Figura 5-3.: Piezas cónicas.



(a) Parámetros R = 6.5 mm y K = -1.

(b) Parametros R = 8.5 mm y e = 0.45.

(c) Parametros R = 8.5 mm y K = -1.

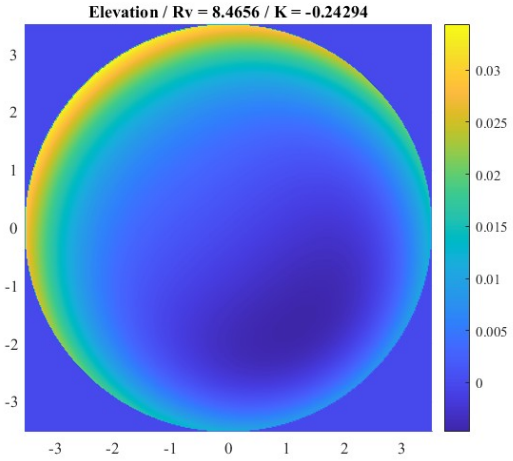
Figura 5-4.: Imágenes a analizar obtenidas de las piezas cónicas.

Una vez adquiridas las imágenes, el usuario puede hacer uso de la ventana de análisis del software desarrollado para realizar el respectivo estudio de las mismas, el cual conduce a los mapas de elevación respecto a una superficie esférica y los mapas de curvatura dados por las Figuras 5-5, 5-6 y 5-7, y como complemento del análisis de las imágenes se muestran los datos obtenidos en la Tabla 5-1. Donde el valor esperado para los parámetros "Delta Z" y "RMS-Rv" es cero, mientras que para los parámetros "Vertex Radius" y "Hartmann Sphere" se espera un valor cercano al radio "R" de la pieza cónica y finalmente se espera que el parámetro "Conic K" coincida con el valor de la constante cónica "K" de la pieza.

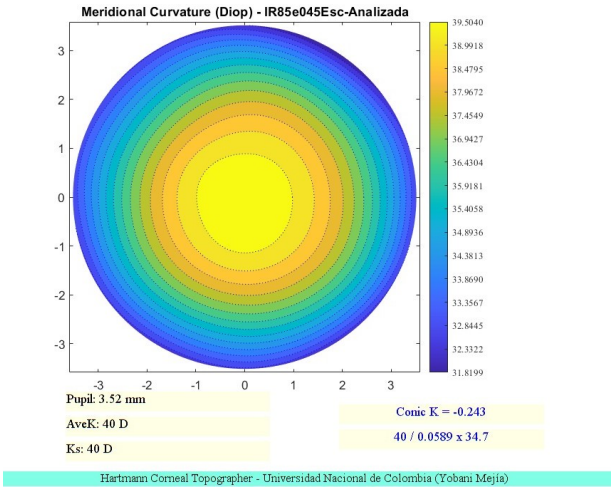
A partir de estos resultados se pueden obtener conclusiones, permitiendo comparar con los datos reales de cada una de las superficies cónicas.

	Superficies cónicas		
	R= 6.5 y K= -1	R= 8.5 y K= -1	R= 8.5, e= 0.45 y K= -0.2
Delta Z (mm)	-0.649	0.319	0.343
Hartmann Sphere (mm)	6.52	8.45	8.51
Vertex Radius (mm)	6.49	8.47	8.47
Conic K	-1.08	-0.964	-0.243
RMS-Rv (mm)	4.83 x10^{-3}	9.71 x10^{-3}	5.83 x10^{-3}

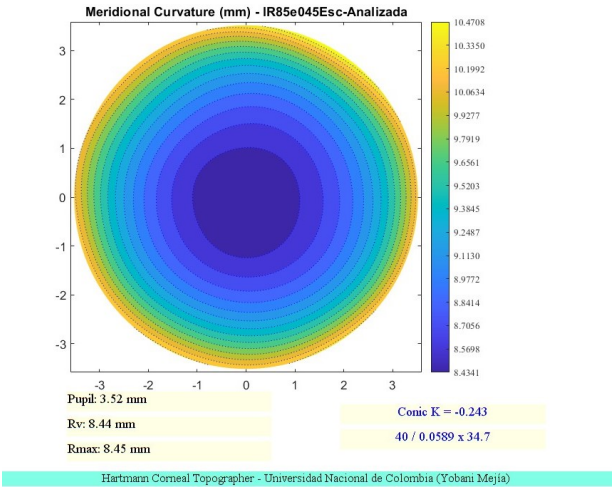
Tabla 5-1.: Tabla de resultados para las superficies cónicas.



(a) Mapa de elevación.

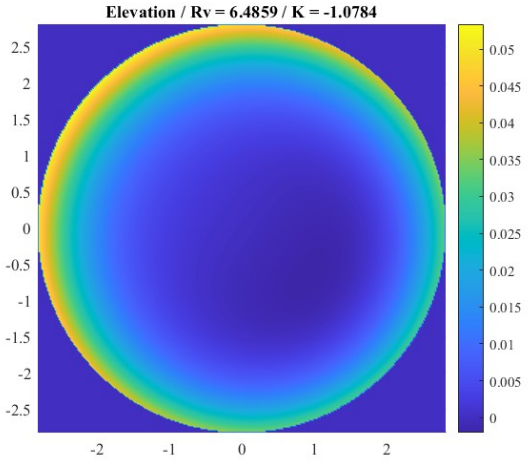


(b) Mapa de curvatura.

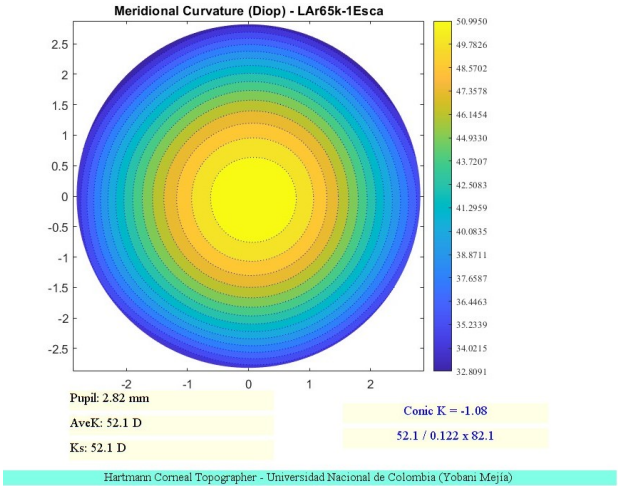


(c) Mapa de curvatura en mm.

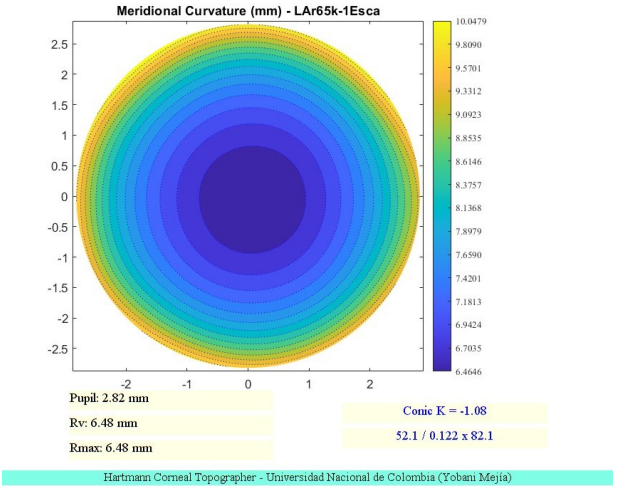
Figura 5-5.: Mapas obtenidos para el análisis topográfico del elemento cónico con parámetros R= 8.5 mm y e= 0.45. Corresponde al objeto de la Figura 5-4-b.



(a) Mapa de elevación.

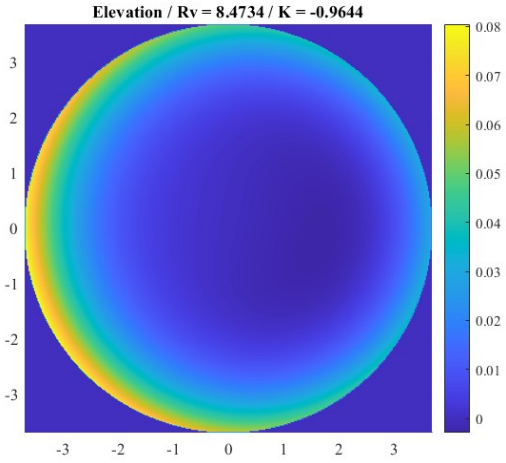


(b) Mapa de curvatura.

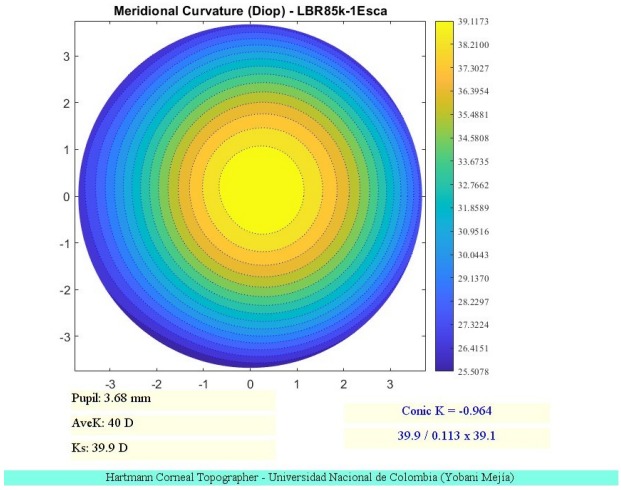


(c) Mapa de curvatura en mm.

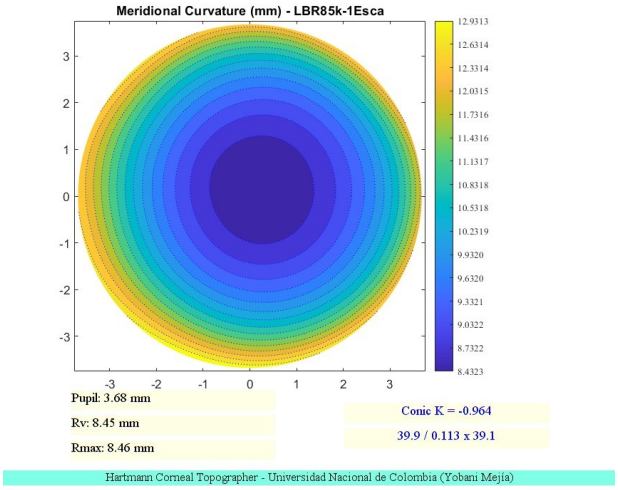
Figura 5-6.: Mapas obtenidos para el análisis topográfico del elemento cónico con parámetros R= 6.5 mm y K= -1. Corresponde al objeto de la Figura 5-4-a.



(a) Mapa de elevación.



(b) Mapa de curvatura.



(c) Mapa de curvatura en mm.

Figura 5-7.: Mapas obtenidos para el análisis topográfico del elemento cónico con parámetros R= 8.5 mm y K= -1. Corresponde al objeto de la Figura 5-4-c.

6. Análisis de resultados

A partir de los resultados previamente mostrados en las Figuras 5-4, se evidencia cómo cada una de las superficies cónicas genera una imagen del patrón de puntos diferente, siendo por ejemplo la Figura 5-4-a, la imagen del patrón mas pequeño, la cual a su vez coincide con la superficie de menor radio. Otro cambio evidente son las distorsiones en el patrón de puntos, siendo la Figura 5-4-b la que presenta una distorsión con mayor diferencia respecto a las otras dos, esto tiene una explicación gracias a la constante cónica de esta superficie, la cual es diferente en comparación demás.

Gracias a las Figuras 5-5, 5-6 y 5-7 y a la información expuesta en la Tabla 5-1, se observa cómo las superficies cuentan con una forma más esférica en el vértice, siendo esto un resultado de esperarse, ya que a escalas de profundidad pequeñas como las que se están trabajando es difícil notar todas las características. Esto también está relacionado con el hecho, de que la aproximación paraxial cuenta con una mayor validez en el centro de la imagen, siendo ahí donde se hace menos evidente las distorsiones que se puedan presentar en las imágenes. Sin embargo, el mapa de curvatura hace evidente diferentes aspectos de la superficie. Por ejemplo, en la Figura 5-5 se aprecia a través del mapa de curvatura cómo la elevación de la superficie no cuenta con una tendencia esférica en todo su rango, algo que se hace más claro a través la líneas de curvatura, en la cuales se observa como estas cuentan con un distanciamiento más corto entre sí a medida que se alejan del vértice de la imagen, lo cual permite evidenciar la excentricidad con la que cuenta, esta característica también se puede evidenciar en algunas córneas humanas [Mejía, 2021]. Por otro lado, también contamos con medidas como las que se aprecian en las Figuras 5-6 y 5-7, las cuales no cuentan con una tendencia esférica marcada, esto dado que la constante cónica de las superficies que dan lugar a estos resultados, es de -1, lo que infiere una excentricidad de 1, siendo esta la característica de una parábola [Warren J, 2000]. Por otra parte, la Figura 5-5 cuenta con una constante cónica, entre 0 y -1 lo que indica que la superficie medida cuenta con características de una elipse [Warren J, 2000]. La diferencia entre estos dos tipos de superficies, se evidencia observando las líneas en los mapas de curvatura, donde vemos que el cambio en el distanciamiento de dichas líneas cuenta con diferencias entre las superficies con características parabólicas y elipsoidales, lo que se ve de igual manera en los mapas de elevación, donde al estar referenciados respecto a una superficie esférica se ve cómo los cambios respecto a la misma varían de forma diferente para los dos tipos de superficie.

El dato "RMS-Rv", debe ser del orden de los micrómetros para este tipo de superficies, lo que al compararlo con los valores brindados por el software para cada una de las imágenes, se evidencia que la escala del cálculo es la correcta, permitiendo decir que el proceso iterativo para el cálculo de la expresión matemática fue un proceso en el que no se obtuvo una discrepancia mínima entre los datos comparados, haciendo fiable el ajuste. Sin embargo, el parámetro "Delta Z" a pesar de ser un parámetro que debería estar muy cercano, no se comporta de esta manera, indicando una diferencia entre la superficie real y la superficie hallada, a pesar de esto el valor no es lo suficientemente elevado para descartar las mediciones dado que es de una magnitud de apenas alrededor de medio milímetro.

Al comparar estos datos con la información teórica de cada superficie, la cual es brindada por el propio diseñador de las mismas, Tabla 6-1, se observa que no se cuenta con una exactitud del cien por ciento, sin embargo, el error con el que se cuenta en la mayoría de las medidas es del orden de las centésimas, por lo que las medidas tomadas cuentan con un buen margen de certeza. En particular, resultados como el mostrado en la Figura 5-4-a, se puede apreciar una alta concordancia entre los datos experimentales y teóricos, infiriendo la buena toma de la medida, sin embargo, los mapas asociados a dicha figura indican un vértice no centrado, lo que junto al hecho de que se sabe teóricamente la superficie analizada, refleja un ligero error en la toma de la medida. Para resolver este tipo de problemas, es posible tomar mas medidas de la pieza y de esta manera evidenciar si se trata de un error en la misma o uno a la hora de tomar la imagen. Por otro lado, también es posible realizar medidas de otras priesas caracterizadas con alta fiabilidad, como lo sería una esfera acero de alta precisión, para de esta manera tener más medidas de error.

Piezas	Parámetro	Valor nominal (Vn)	Valor calculado (Vc)	Error absoluto ($\|Vn - Vc\|$)
1	Conic K	-1	-1.08	8 x10^{-2}
	Hartmann Sphere (mm)	6.5	6.52	2 x10^{-2}
2	Conic K	-1	-0.964	3.6 x10^{-2}
	Hartmann Sphere (mm)	8.5	8.45	5 x10^{-2}
3	Conic K	-0.2	-0.243	4.3 x10^{-2}
	Hartmann Sphere (mm)	8.5	8.51	1 x10^{-2}

Tabla 6-1.: Error entre los datos calculados y nominales por cada pieza cónica. Donde la pieza 1 corresponde a los resultados dados en la Figura 5-6, la pieza 2 corresponde a los resultados dados en la Figura 5-7 y la pieza 3 corresponde a los resultados dados en la Figura 5-5.

7. Conclusiones

Durante la actualización del hardware y software del topógrafo fueron encontrados diferentes inconvenientes de funcionamiento que fueron solucionados a lo largo del proyecto. Sin embargo, el alineamiento de los láseres no fue alterado durante la ejecución del proyecto debido a que a pesar de que no cumplen con su objetivo de diseño, indicar la distancia al elemento a medir, no impiden el buen uso del topógrafo y siguen siendo útiles para uso de referencia en cierto tipo de medidas. Por otro lado, los resultados topográficos de los moldes de lentes de contacto reflejaron una medida confiable, donde los datos obtenidos por el software de análisis muestran un radio de los datos centrales "Hartmann phere" y un radio general "Vertex Radius", muy cercanos entre sí y al resultado esperado, el cual está dado por los datos de fabricación de los lentes. A pesar de esto, el valor obtenido para el parámetro "Delta Z" a pesar de ser un valor cercano a cero, cuenta con un valor de magnitud considerable (décimas de milímetros), por lo que es una razón para estimar un error en la medida.

A partir del desarrollo de la captura de las imágenes antes mencionadas, se logra concluir el buen desempeño del topógrafo, el cual queda como herramienta de posibles trabajos pedagógicos del grupo de óptica aplicada, del cual se concluye que la distancia óptima del topógrafo al objeto a medir su topografía es de aproximadamente 3 mm, contando con un rango superior a 15 mm de radio del objeto para analizar topografía, los cuales son valores adecuados teniendo en cuenta que la córnea humana cuenta con radio medio de 6.5 mm [Mejía, 2021], lo que impide la formación de una imagen al interior del ojo de un tamaño superior al máximo posible para medición.

Dado el software de análisis implementado, el cual sigue la secuencia iterativa donde halla la expresión matemática de la superficie del elemento a medir, en polinomios de Zernike y los resultados obtenidos con el mismo, es posible plantear oportunidades de mejora para el proyecto, como utilizar una aproximación matemática en términos de los polinomios de Taylor, dado que estos ya han sido utilizados para este tipo de análisis [de la Fuente Arriaga, 2017] [Piñera, 1998]. A parte de esto, también es posible mejorar la calidad de la medida al implementar elementos de mayor calidad, como un mejor sensor o filamentos lumínicos para la pantalla de Hartman con una calibración superior.

Un aspecto importante a ser mencionado es el hecho de que la calidad de las medidas tomadas cuenta con una alta responsabilidad del usuario, ya que se demostró que una variación

pequeña de la distancia respecto del lente puede repercutir en resultados con cambios significativos, lo que hace más difícil tomar medidas de córneas, teniendo en cuenta que para esto se requiere que la persona a ser examinada cuente con una buena capacidad de concentración y enfoque en el ojo. Este problema busca encontrar una solución al darle la posibilidad al dispositivo de capturar medidas en forma de vídeos, donde el usuario tendrá la posibilidad de elegir el "frame" más adecuado para el estudio.

El desarrollo de este proyecto deja como resultado una herramienta funcional que abre la puerta a futuros estudios, como podría ser el estudio de malformaciones en la córnea como el queratocono, la falta de simetría que dé pie a la presencia del astigmatismo corneal o la comparación entre diferentes métodos de medición de topografía para lentes. A parte de esto, el dispositivo desarrollado cuenta con la posibilidad de ser complementada a través de la automatización, la cual permitiría la ubicación automática o manejada electrónicamente por medio de motores de precisión, lo que minimizaría la probabilidad de errores en las medidas tomadas. Otra oportunidad de mejora con la que cuenta el dispositivo está en complementar el software de análisis, al buscar la posibilidad de que este sea capaz de interpretar por sí mismo cuando una imagen cuenta con las características necesarias para dar resultados de alta precisión.

A. Anexo: Hardware

En nuestro topógrafo de córnea el hardware cuenta con una electrónica no muy complicada, pero diseñada específicamente para garantizar la durabilidad y el buen funcionamiento del dispositivo. Las generalidades de la electrónica ya fueron descritas previamente en el capítulo 4, sin embargo, no ha sido especificado los artefactos utilizados para el accionamiento de los elementos del topógrafo, así como su alimentación, por otro lado, también hace falta aclarar la conexión de los interruptores al Arduino.

Para dar claridad a las conexiones, se parte del hecho de que se cuenta con 12 voltios de entrada y se debe tener en cuenta que el voltaje requerido dado para las entradas de los interruptores hacia el Arduino como para encender los láseres y la mirilla, es de 5 voltios. Dado esto, y tomando en cuenta el hecho de que el Arduino es alimentado a partir de los 12 voltios extraídos de la fuente tenemos que deben compartir la conexión a tierra "GND" para el buen funcionamiento y con el fin de evitar, minimizar y controlar incidentes se realiza una conexión de estas tierras con la conexión polo a tierra que brinda él toma corriente, lo que da tranquilidad y facilidad a la hora de hacer conexiones dado que ahora solo tenemos una tierra, la cual es la tierra común del sistema.

Los interruptores y pulsadores funcionan permitiendo el paso o no de la corriente hacia las entradas del Arduino, es por esto que el voltaje al que se conectan estos elementos es extraído directamente de la salida de 5 voltios que brinda el Arduino, lo que permite asegurar que a las entradas del Arduino no va a llegar un voltaje mayor del que ese acepta. Por otro lado, la corriente de entrada es limitada por una resistencia que se conecta en paralelo y a tierra, permitiendo al microcontrolador tener como entrada una corriente aceptable y una referencia de tierra en caso de que el interruptor o pulsador estén en estado abierto.

El proceso de accionamiento de los elementos lumínicos del topógrafo: lámpara, mirilla y láseres se realizan a través del voltaje de salida digital del Arduino (5 voltios), para el cual se utiliza un circuito integrado "ULN2803", que funciona como amplificador Darlington y se puede ver en un diagrama lógico como una compuerta "Not" para cada una de sus entradas con su respectiva salida [Incorporated, 2004]. Este circuito integrado permite el manejo de corrientes hasta de 500 mA y funciona como protección de corrientes elevadas e inversas hacia el Arduino [Boylestad and Nashelsky, 2009]. Teniendo en cuenta que los láseres tienen un bajo consumo de corriente, el "ULN2803", permite que estos sean conectados de manera

directa a su salida, el cual cuenta con la salida del Arduino como entrada facilitando su control. Por otro lado, la mirilla y la lámpara cuentan con voltajes de funcionamiento superiores a 5 voltios, por lo que se utiliza alimentación externa cuyo control se hace a partir de módulos relé a 5 voltios, donde sus respectivos pines de control son alimentados a partir del voltaje de salida de los respectivos puertos digitales del Arduino pasando por el "UNL2803", por lo que a la hora de realizar la programación de estos elementos se debe tener en cuenta que la lógica es cambiada por el circuito integrado y el módulo relé [Guevara, 2018]. Cabe aclarar que estos módulos funcionan como interruptores digitales por lo que cuentan con las conexiones de un interruptor normal, siendo la alimentación de los elementos aislada a la del interruptor, lo que evita que la corriente requerida por los elementos lumínicos no afecte al microcontrolador, ni la etapa de control. Como fue mencionado anteriormente, la lámpara es el elemento que requiere de una mayor cantidad de voltaje por lo que sus conexiones deben hacerse con mucho cuidado, para su caso, el módulo relé permite o no el paso del voltaje alterno de 120 voltios el cual después es elevado posteriormente y de esta manera se evitan tener voltajes muy altos cuando no sea necesario. Para la mirilla se requiere una alimentación de 12 voltios, el cual es extraído de la fuente original, de esta manera aseguramos que la corriente requerida para el funcionamiento de este elemento sea extraída de la fuente y no del Arduino. El esquema de este circuito se puede apreciar en la Figura A-1, donde se puede evidenciar la relación entre este plano eléctrico y el esquema mostrado en la Figura 4-1 a través de las conexiones de entradas al Arduino, las cuales cuentan con el mismo nombre en ambos esquemas.

En este plano se evidencia que el circuito cuenta con un componente denotado como activador, el cual es un componente reciclado del del laboratorio, que cuenta con el circuito integrado "moc3031" como componente principal, el cual permite funcionar como activador, en este caso, del flujo de corriente alterna que alimenta a la lampara. Permitiendo que este elemento cuente con dos componentes de activación, lo cual es sumamente útil teniendo en cuenta la cantidad de voltaje que este maneja. Por otra parte, es importante aclarar el hecho de que solo se cuenta con salidas y entradas que se comportan digitalmente, es decir, encendidas o apagadas. Sin embargo, por fines prácticos y facilidad a la hora de realizar el diseño del circuito, se conectaron tres interruptores a entradas análogas, por lo que por medio de la programación se interpreta el valor de entrada como encendido o apagado.

Para el proyecto también se desarrolla un circuito que proteja y haga practica la alimentación del microprocesador, junto con la comunicación con el computador. Para lo que se utiliza un módulo relé, el cual en es alimentado por el voltaje proveniente del computador, que a su vez es extraído a partir del cable USB, de este cable se extrae las líneas de GND y 5 voltios, haciendo que el GND sea una tierra común con la correspondiente a la salida de la fuente del topógrafo. El pin de control del módulo relé, se conecta al GND común, lo que permite que al conectar cable USB, se active el módulo, y de esta manera elegir entre alimentación ex-

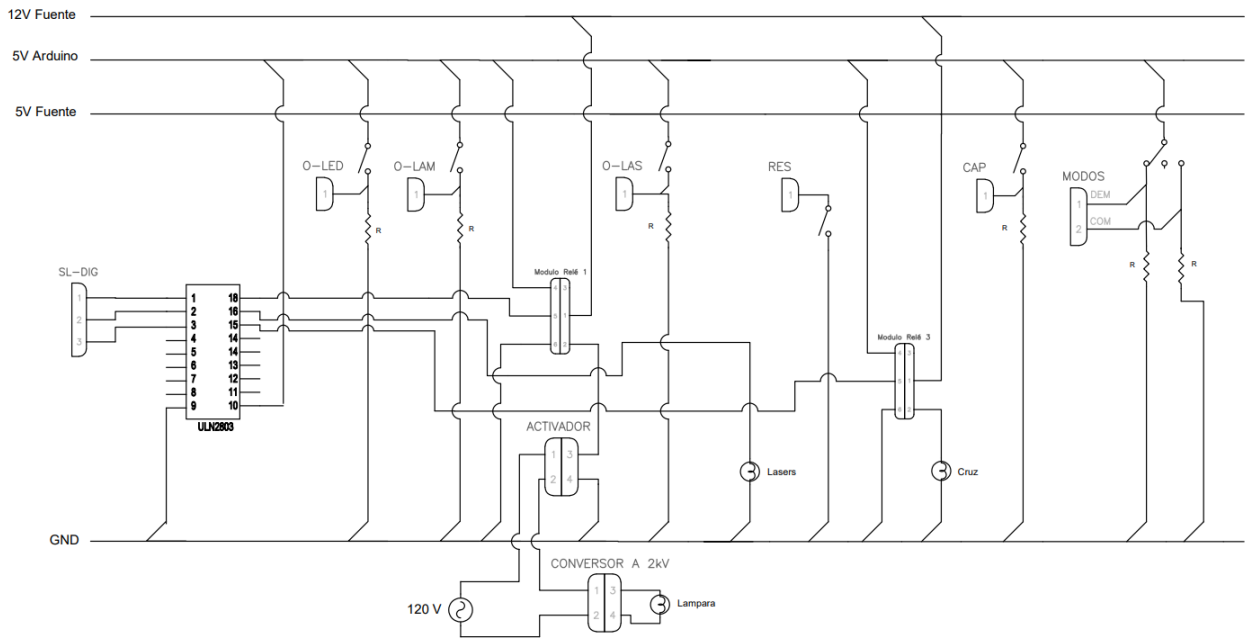


Figura A-1.: Plano eléctrico de la conexión de interruptores, botones y elementos luminosos del topógrafo, desarrollado en el software AutoCad Electrical.

terna al Arduino o alimentación por cable. Dicho circuito se puede apreciar en la Figura A-2.

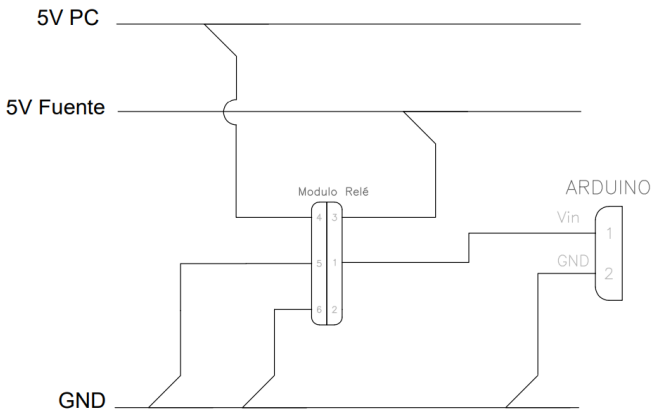


Figura A-2.: Plano eléctrico de alimentación del microprocesador (Arduino), desarrollado en el software AutoCad Electrical.

Cabe aclarar que en las figuras A-1 y A-2, se observa los módulos relés, lo cuales cuentan con 6 puntos de conexiones que van enumerados del 1 al 6 y estos corresponden en ese orden respectivo a las conexiones de: común, normal abierto, normal cerrado, alimentación, control

y GND.

B. Anexo: Software del microcontrolador (Arduino)

A continuación, se pone en evidencia el código desarrollado e implementado en el Arduino del topógrafo, en el cual resalta el uso de las librerías (`#include <SoftwareSerial.h>`) y (`#include <LiquidCrystal_I2C.h>`), las cuales son utilizadas para la comunicación con el computador e impresión de información en la pantalla LCD.

```
/*
 * Código desarrollado para control del topógrafo de cornea
 * del grupo de óptica aplicada. (2022)
 */

// Inclusión de librerias
#include <SoftwareSerial.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Inicialización de pantalla lcd
LiquidCrystal_I2C lcd(0x27,16,2);

// Declaración de puertos de salida
int lampara = 2;
int mirilla = 3;
int laser = 4;

// Declaración de puertos de entradas digitales
int reset = 5;
int captura = 6;
int modInac = 12;
int modCom = 10;

// Declaración de puertos de entradas analogas
int Alaser = 1;
int Alampara = 2;
int Amirilla = 3;
int ADem = 0;

// Declaración de variables de entradas analogas
int lLaser = 0;
int lLampara = 0;
int lMirilla = 0;
int IDem = 0;

// Declaración de tiempo para el apagado de lámpara (ms)
double timeLam = 600000; // 10 minutos

// Declaración de tiempo actual para comparar
unsigned long timeAct = 0;
unsigned long timeOn = 0;

// Declaración de variable de comunicación serial
char tipoSerial;

// Declaración de variable para validad el encendido de la lámpara
int fueOff = 1;
int fueOffCom = 1;

void setup() {
  // Declaración de velocidades para la comunicación
  Serial.begin(9600);
  Serial.setTimeout(10);

  // Declaración de puertos de salidas digitales
  pinMode(laser, OUTPUT);
```

```
  pinMode(lampara, OUTPUT);
  pinMode(mirilla, OUTPUT);

  // Declaración de puertos de entradas digitales
  pinMode(captura, INPUT);
  pinMode(reset, INPUT);
  pinMode(modInac, INPUT);
  pinMode(modCom, INPUT);

  // Inicialización el objeto LCD
  lcd.init();
  // Activación de la luz de fondo de la pantalla
  lcd.backlight();

  // Se apagan los elementos del topografo
  digitalWrite(laser,LOW);
  digitalWrite(lampara,HIGH);
  digitalWrite(mirilla,HIGH);
}

void loop() {
  // Se establece un tiempo entre cada iteración
  delay(500);

  // Lectura de entradas análogas
  lLaser = analogRead(Alaser);
  lLampara = analogRead(Alampara);
  lMirilla = analogRead(Amirilla);
  IDem = analogRead(ADem);

  // Validación del tiempo de encendido de la lampara y de ser necesario,
apagado de la misma
  timeAct = millis();
  if (timeAct > timeOn+timeLam){
    digitalWrite(lampara,HIGH);
  }

  // Identificación del modo inactivo
  if ((IDem < 900 && digitalRead(modCom)==LOW )){
    // Se apagan los elementos del topografo y se indica mediante la
pantalla LCD
    lcd.setCursor(0, 0);
    lcd.print("Modo: Inactivo  ");
    lcd.setCursor(0, 1);
    lcd.print("Las:OFF");
    lcd.setCursor(8, 1);
    lcd.print("Mira:OFF");
    delay(100);

    // Apagamos todo
    digitalWrite(laser,LOW);
    digitalWrite(lampara,HIGH);
    digitalWrite(mirilla,HIGH);

  // Identificación del modo demostración
  }else if(IDem > 900 ){  //  Entra en modo demostración
    // Se indica mediante la pantalla LCD
```

```
    lcd.setCursor(0, 0);
    lcd.print("Modo: Demosatra ");
    delay(100);

    // Se apagan o enciende los elementos del topógrafo y se indica
mediante la pantalla LCD
    if(lLaser > 950){
      digitalWrite(laser,HIGH);
      lcd.setCursor(0, 1);
      lcd.print("Las:ON ");
    }else{
      digitalWrite(laser,LOW);
      lcd.setCursor(0, 1);
      lcd.print("Las:OFF");
    }

    // Verifica si la lampara fue apagada por mucho tiempo de uso, de ser
así así se puede volver a encender
    if(lLampara > 950 && fueOff == 1){
      digitalWrite(lampara,LOW);
      timeOn = millis();
      fueOff=0;
    }else if(lLampara <= 950 && fueOff == 0){
      digitalWrite(lampara,HIGH);
      fueOff = 1;
    }

    if(lMirilla > 950){
      digitalWrite(mirilla,LOW);
      lcd.setCursor(8, 1);
      lcd.print("Mira:ON ");
    }else{
      digitalWrite(mirilla,HIGH);
      lcd.setCursor(8, 1);
      lcd.print("Mira:OFF");
    }

  // Identificación del modo computador
  }else if(digitalRead(modCom)==HIGH){ //  Entra en modo computadora
    // Se indica mediante la pantalla LCD
    lcd.setCursor(0, 0);
    lcd.print("Modo: Computador");
    delay(100);

    // Identificación de comunicación serial
    if(Serial.available()){
      // Leectura de comunicación serial
      tipoSerial = Serial.read();
      delay(10);

      // Se apagan o enciende los elementos del topógrafo y se indica
mediante la pantalla LCD,
      // de acuerco L = laser, M = lampara y C = mirilla (Cruz), al ser
la letra recibida minuscula
      // se encendera el elemento, de lo contrario se apagará
      switch(tipoSerial){
        case 'L':
```

```
          digitalWrite(laser,LOW);
          lcd.setCursor(0, 1);
          lcd.print("Las:OFF");
        break;

        case 'l':
          digitalWrite(laser,HIGH);
          lcd.setCursor(0, 1);
          lcd.print("Las:ON ");
        break;

        case 'M':
        // Verifica si la lampara fue apagada por mucho tiempo de uso, de
ser así así se puede volver a encender
        if(fueOffCom == 0){
          digitalWrite(lampara,HIGH);
          fueOffCom = 1;
        }

        break;

        case 'm':
          if(fueOffCom == 1){
            digitalWrite(lampara,LOW);
            fueOffCom = 0;
            timeOn = millis();
          }
        break;

        case 'C':
          digitalWrite(mirilla,HIGH);
          lcd.setCursor(8, 1);
          lcd.print("Mira:OFF");
        break;

        case 'c':
          digitalWrite(mirilla,LOW);
          lcd.setCursor(8, 1);
          lcd.print("Mira:ON ");
        break;

      }
    }

    // Se escribe en el puerto serial la señal de captura
    if(digitalRead(captura)==HIGH){
      Serial.println("1");
    }else{
      Serial.println("0");
    }
    delay(100);
  }

}
```

C. Anexo: Software de la interfaz en computadora (MATLAB)

La interfaz en computadora es un programa desarrollado en Matlab el cual cuenta con diferentes ventanas, que a su vez cuentan con su respectivo código de programación. Dado que este programa cuenta con la capacidad de comunicarse con el Arduino, realizar análisis de imágenes y comunicación con la cámara, para su desarrollo es indispensable en contar con los paquetes de software que se aprecian en la Tabla C-1, los cuales son instalados como herramientas de Matlab.

Paquete	Descripción
MATLAB Support Packege for Arduino Hardware	Permite la comunicación entre Matlab y Arduino.
Matlab Support Packege for USB Webcams	Permite la comunicación entre Matlab y cámaras WebCam.
Image Acquisition toolbox Support Packege for OS Generic Video Interface	Permite la adquisición de imágenes y videos.
Image Processing ToolBox	Permite el procesamiento de imágenes.
Thorlabs DCx in MATLAB	Permite el uso de las cámaras diseñadas por la empresa ThorLabs de la categoria DCx.
Matlab Compiler	Permite generar la aplicación (archivo.exe) desde Matlab.

Tabla C-1.: Paquetes de software de Matlab instalados para el desarrollo de la interfaz.

Cabe aclarar que el esta interfaz cuenta con la capacidad de identificar cuando se encuentra o no conectada la cámara o el mismo topógrafo, lo que permite enviar mensajes de error al usuario y con lo mismo evitar errores de funcionamiento de la pantalla. Dichos mensajes se aprecian en la Tabla C-2, junto con su respectiva descripción.

Mensaje	**Descripción**
No se detectó el topógrafo.	El software no detecta el topógrafo. Puede aparecer en la ventana de control y de captura.
Recuerde mantener los láseres apagados a la hora de realizar tomas sobre el ojo.	Recuerda al usuario que a la hora de realizar la captura de imágenes sobre ojos los láseres deben de estar apagados. Puede aparecer en la ventana de captura. Cabe resaltar que los láseres se encontrar apagados o se apagaran por defecto a la hora de realizar la captura.
No se detectó la cámara.	El software no detecta la cámara. Puede aparecer en la ventana de captura.
Se cancelo el proceso de guardado.	No se realizó el proceso de guardado de imagen o video. Puede aparecer en la ventana de captura o análisis.
No se reconoce el archivo seleccionado.	El archivo seleccionado no está en formato valido de imagen o video. Puede aparecer en la ventana de análisis.

Tabla C-2.: Mensajes de error y advertencia.

A continuación, se presenta el código desarrollado por cada una de las ventanas utilizadas por la interfaz.

C.1. Ventana de inicio

En las siguientes paginas se muestra el código asociado a la ventana de inicio.

```
classdef Inicio < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        TOPGRAFODECRNEAUIFigure  matlab.ui.Figure
        Panel                    matlab.ui.container.Panel
        AnlisisButton            matlab.ui.control.Button
        CapturaButton            matlab.ui.control.Button
        ControlButton            matlab.ui.control.Button
        TOPGRAFODECRNEALabel     matlab.ui.control.Label
        Image_2                  matlab.ui.control.Image
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: ControlButton
        function ControlButtonPushed(app, event)
            % Se abre la ventana de control y se cierra la ventana de inicio
            VenControl
            delete(app)
        end

        % Button pushed function: AnlisisButton
        function AnlisisButtonPushed(app, event)
            % Se abre la ventana de analisis y se cierra la ventana de inicio
            VenAnalisis
            delete(app)
        end

        % Button pushed function: CapturaButton
        function CapturaButtonPushed(app, event)
            % Se abre la ventana de captura y se cierra la ventana de inicio
            VenCaptura
            delete(app)
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Get the file path for locating images
            pathToMLAPP = fileparts(mfilename('fullpath'));

            % Create TOPGRAFODECRNEAUIFigure and hide until all components are created
            app.TOPGRAFODECRNEAUIFigure = uifigure('Visible', 'off');
            app.TOPGRAFODECRNEAUIFigure.Color = [0 0.4471 0.7412];
            app.TOPGRAFODECRNEAUIFigure.Position = [100 100 612 339];
            app.TOPGRAFODECRNEAUIFigure.Name = 'TOPÓGRAFO DE CÓRNEA';
            app.TOPGRAFODECRNEAUIFigure.Resize = 'off';

            % Create Panel
            app.Panel = uipanel(app.TOPGRAFODECRNEAUIFigure);
            app.Panel.BackgroundColor = [1 1 1];
            app.Panel.Position = [34 1 545 339];

            % Create Image_2
            app.Image_2 = uiimage(app.Panel);
            app.Image_2.Position = [29 85 488 184];
            app.Image_2.ImageSource = fullfile(pathToMLAPP, 'LopoAplicacion.PNG');
```

```
            % Create TOPGRAFODECRNEALabel
            app.TOPGRAFODECRNEALabel = uilabel(app.Panel);
            app.TOPGRAFODECRNEALabel.FontSize = 24;
            app.TOPGRAFODECRNEALabel.FontWeight = 'bold';
            app.TOPGRAFODECRNEALabel.Position = [115 268 315 30];
            app.TOPGRAFODECRNEALabel.Text = 'TOPÓGRAFO DE CÓRNEA';

            % Create ControlButton
            app.ControlButton = uibutton(app.Panel, 'push');
            app.ControlButton.ButtonPushedFcn = createCallbackFcn(app, @ControlButtonPushed, true);
            app.ControlButton.Position = [78 39 100 22];
            app.ControlButton.Text = {'Control'; ''};

            % Create CapturaButton
            app.CapturaButton = uibutton(app.Panel, 'push');
            app.CapturaButton.ButtonPushedFcn = createCallbackFcn(app, @CapturaButtonPushed, true);
            app.CapturaButton.Position = [372 39 100 23];
            app.CapturaButton.Text = 'Captura';

            % Create AnlisisButton
            app.AnlisisButton = uibutton(app.Panel, 'push');
            app.AnlisisButton.ButtonPushedFcn = createCallbackFcn(app, @AnlisisButtonPushed, true);
            app.AnlisisButton.Position = [223 39 100 23];
            app.AnlisisButton.Text = 'Análisis';

            % Show the figure after all components are created
            app.TOPGRAFODECRNEAUIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Inicio

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.TOPGRAFODECRNEAUIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.TOPGRAFODECRNEAUIFigure)
        end
    end
end
```

C.2. Ventana de control

En las siguientes paginas se muestra el código asociado a la ventana de control.

```
classdef VenControl < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        TOPGRAFODECRNEAUIFigure  matlab.ui.Figure
        Image_2                  matlab.ui.control.Image
        CapturaButton            matlab.ui.control.Button
        AnlisisButton            matlab.ui.control.Button
        Panel                    matlab.ui.container.Panel
        DesconectarButton        matlab.ui.control.Button
        LedCom                   matlab.ui.control.Lamp
        ComunicacionLampLabel    matlab.ui.control.Label
        AtrasButton              matlab.ui.control.Button
        CruzSwitch               matlab.ui.control.Switch
        CruzSwitchLabel          matlab.ui.control.Label
        LamparaSwitch            matlab.ui.control.Switch
        LamparaSwitchLabel       matlab.ui.control.Label
        LaserSwitch              matlab.ui.control.Switch
        LaserSwitchLabel         matlab.ui.control.Label
        ConectarButton           matlab.ui.control.Button
        ControldeltopgrafoLabel  matlab.ui.control.Label
    end


    properties (Access = private)
        % Se crea la propiedad cumu, la cual se utilizará para la
        % comunicación
        comu % Comunición serial (topografo)
    end

    methods (Access = private)
        % Funciones desarrolladas para el funcionamiento de la ventana

        function establecerComu(app)
            % Se identifica si hay o no la comunicación con el Arduino,
            % de haberse establece la comunicación serial.
            puertos=serialportlist;

            if isempty(serialportlist)
                f=warndlg("No se detecto el topografo",'Error');
                app.LedCom.Color = 'red';
            else
                puerto=puertos(1);
                com = puerto;
                app.comu = serial(com,'BaudRate',9600,'Terminator','CR/LF');
                app.LedCom.Color = 'red';
            end

        end

        function apagarTodo(app)
            % En caso de estar la comunicación encendida, se envía la
            % información para apagar todos los elementos del topógrafo
            if colorangle([0 1 0],app.LedCom.Color)==0
                try
                    fwrite(app.comu,'L');
                    pause(0.5);
                    fwrite(app.comu,'M');
                    pause(0.5);
                    fwrite(app.comu,'C');
                catch

                end
            end
        end

    end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            % Se inicializa las variables globales del sistema y se
            % establece la comunicación en caso de ser posible, para
            % seguido de esto apagar los elementos del topógrafo
            global lamp
            global cruz
            global laser
```

```
    lamp = 'Off';
    cruz = 'Off';
    laser = 'Off';

    puertos=serialportlist;

    if isempty(serialportlist)
        f=warndlg("No se detecto el topografo",'Error');
        app.LedCom.Color = 'red';
    else
        puerto=puertos(1);
        com = puerto;
        app.LedCom.Color = 'red';

        try
            app.comu = serial(com,'BaudRate',9600,'Terminator','CR/LF');
            fopen(app.comu);
            app.LedCom.Color = 'green';
        catch
            f=warndlg("No se detecto el topografo",'Error');
        end
    end

    apagarTodo(app);


end

% Button pushed function: AtrasButton
function AtrasButtonPushed(app, event)
    % Se abre la ventana de inicio, se cierra la comunicación y
    % la ventana de control
    Inicio
    try
        fclose(app.comu);
    catch

    end
    delete(app.comu);
    delete(app)

end

% Close request function: TOPGRAFODECRNEAUIFigure
function TOPGRAFODECRNEAUIFigureCloseRequest(app, event)
    % Se cierra la ventana de inicio, la comunicación y
    % la ventana de control
    try
        fclose(app.comu);
    catch

    end
    delete(app.comu);
    delete(app);
end

% Value changed function: LaserSwitch
function LaserSwitchValueChanged(app, event)
    % De ser posible, se envía la señal de encendido o apagado de
    % los lásers.
    global laser
    value = app.LaserSwitch.Value;
    laser = value;

    try
        if strcmp(laser,'Off')
            fwrite(app.comu,'L');
        elseif strcmp(laser,'On')
            fwrite(app.comu,'l');
        end
    catch
        f=warndlg("No se detecto el topografo",'Error');
        app.LedCom.Color = 'red';
    end


end

% Button pushed function: ConectarButton
function ConectarButtonPushed(app, event)
    % Se estable o reinicia la comunicación serial con el topógrafo
    try
        fclose(app.comu);
    catch
```

```
    end
    if isempty(app.comu)
        establecerComu(app);
        try
            fopen(app.comu);
            app.LedCom.Color = 'green';

        catch
            f=warndlg("No se detecto el topografo",'Error');
        end
    else
        try
            fopen(app.comu);
            app.LedCom.Color = 'green';
        catch
            f=warndlg("No se detecto el topografo",'Error');
        end
    end

end

% Button pushed function: DesconectarButton
function DesconectarButtonPushed(app, event)
    % Se cierra y elimina la comunicación serial con el topógrafo
    try
        fclose(app.comu);
    catch

    end
    app.LedCom.Color = 'red';
    clear all;
end

% Value changed function: LamparaSwitch
function LamparaSwitchValueChanged(app, event)
    % De ser posible, se envía la señal de encendido o apagado de
    % los lampara.
    global lamp

    value = app.LamparaSwitch.Value;
    lamp = value;

    try
        if strcmp(lamp,'Off')
            fwrite(app.comu,'M');
        elseif strcmp(lamp,'On')
            fwrite(app.comu,'m');
        end
    catch
        f=warndlg("No se detecto el topografo",'Error');
        app.LedCom.Color = 'red';
    end

end

% Value changed function: CruzSwitch
function CruzSwitchValueChanged(app, event)
    % De ser posible, se envía la señal de encendido o apagado de
    % los cruz.
    global cruz
    value = app.CruzSwitch.Value;
    cruz = value;

    try
        if strcmp(cruz,'Off')
            fwrite(app.comu,'C');
        elseif strcmp(cruz,'On')
            fwrite(app.comu,'c');
        end
    catch
        f=warndlg("No se detecto el topografo",'Error');
        app.LedCom.Color = 'red';
    end

end

% Button pushed function: AnlisisButton
function AnlisisButtonPushed(app, event)
    % Se abre la ventana de análisis , se cierra la comunicación y
    % la ventana de control
    VenAnalisis
    try
        fclose(app.comu);
```

```
        catch

        end
        delete(app.comu);
        delete(app)
    end

    % Button pushed function: CapturaButton
    function CapturaButtonPushed(app, event)
        % Se abre la ventana de captura, se cierra la comunicación y
        % la ventana de control
        VenCaptura
        try
            fclose(app.comu);
        catch

        end
        delete(app.comu);
        delete(app)
    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Get the file path for locating images
        pathToMLAPP = fileparts(mfilename('fullpath'));

        % Create TOPGRAFODECRNEAUIFigure and hide until all components are created
        app.TOPGRAFODECRNEAUIFigure = uifigure('Visible', 'off');
        app.TOPGRAFODECRNEAUIFigure.Color = [0 0.4471 0.7412];
        app.TOPGRAFODECRNEAUIFigure.Position = [100 100 530 493];
        app.TOPGRAFODECRNEAUIFigure.Name = 'TOPÓGRAFO DE CÓRNEA';
        app.TOPGRAFODECRNEAUIFigure.Resize = 'off';
        app.TOPGRAFODECRNEAUIFigure.CloseRequestFcn = createCallbackFcn(app, @TOPGRAFODECRNEAUIFigureCloseRequest, true);

        % Create Panel
        app.Panel = uipanel(app.TOPGRAFODECRNEAUIFigure);
        app.Panel.BackgroundColor = [1 1 1];
        app.Panel.Position = [93 1 440 493];

        % Create ControldeltopgrafoLabel
        app.ControldeltopgrafoLabel = uilabel(app.Panel);
        app.ControldeltopgrafoLabel.FontSize = 24;
        app.ControldeltopgrafoLabel.FontWeight = 'bold';
        app.ControldeltopgrafoLabel.Position = [76 421 253 30];
        app.ControldeltopgrafoLabel.Text = 'Control del topógrafo';

        % Create ConectarButton
        app.ConectarButton = uibutton(app.Panel, 'push');
        app.ConectarButton.ButtonPushedFcn = createCallbackFcn(app, @ConectarButtonPushed, true);
        app.ConectarButton.Position = [84 314 100 22];
        app.ConectarButton.Text = 'Conectar';

        % Create LaserSwitchLabel
        app.LaserSwitchLabel = uilabel(app.Panel);
        app.LaserSwitchLabel.HorizontalAlignment = 'center';
        app.LaserSwitchLabel.Position = [208 80 35 22];
        app.LaserSwitchLabel.Text = 'Laser';

        % Create LaserSwitch
        app.LaserSwitch = uiswitch(app.Panel, 'slider');
        app.LaserSwitch.ValueChangedFcn = createCallbackFcn(app, @LaserSwitchValueChanged, true);
        app.LaserSwitch.Position = [202 117 45 20];

        % Create LamparaSwitchLabel
        app.LamparaSwitchLabel = uilabel(app.Panel);
        app.LamparaSwitchLabel.HorizontalAlignment = 'center';
        app.LamparaSwitchLabel.Position = [199 226 53 22];
        app.LamparaSwitchLabel.Text = 'Lampara';

        % Create LamparaSwitch
        app.LamparaSwitch = uiswitch(app.Panel, 'slider');
        app.LamparaSwitch.ValueChangedFcn = createCallbackFcn(app, @LamparaSwitchValueChanged, true);
        app.LamparaSwitch.Position = [202 263 45 20];

        % Create CruzSwitchLabel
        app.CruzSwitchLabel = uilabel(app.Panel);
        app.CruzSwitchLabel.HorizontalAlignment = 'center';
        app.CruzSwitchLabel.Position = [210 155 31 22];
        app.CruzSwitchLabel.Text = 'Cruz';
```

```
            % Create CruzSwitch
            app.CruzSwitch = uiswitch(app.Panel, 'slider');
            app.CruzSwitch.ValueChangedFcn = createCallbackFcn(app, @CruzSwitchValueChanged, true);
            app.CruzSwitch.Position = [202 192 45 20];

            % Create AtrasButton
            app.AtrasButton = uibutton(app.Panel, 'push');
            app.AtrasButton.ButtonPushedFcn = createCallbackFcn(app, @AtrasButtonPushed, true);
            app.AtrasButton.Position = [328 16 100 22];
            app.AtrasButton.Text = 'Atras';

            % Create ComunicacionLampLabel
            app.ComunicacionLampLabel = uilabel(app.Panel);
            app.ComunicacionLampLabel.HorizontalAlignment = 'right';
            app.ComunicacionLampLabel.Position = [24 16 82 22];
            app.ComunicacionLampLabel.Text = {'Comunicacion'; ''};

            % Create LedCom
            app.LedCom = uilamp(app.Panel);
            app.LedCom.Position = [121 16 20 20];
            app.LedCom.Color = [1 0 0];

            % Create DesconectarButton
            app.DesconectarButton = uibutton(app.Panel, 'push');
            app.DesconectarButton.ButtonPushedFcn = createCallbackFcn(app, @DesconectarButtonPushed, true);
            app.DesconectarButton.Position = [256 314 100 22];
            app.DesconectarButton.Text = 'Desconectar';

            % Create AnlisisButton
            app.AnlisisButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
            app.AnlisisButton.ButtonPushedFcn = createCallbackFcn(app, @AnlisisButtonPushed, true);
            app.AnlisisButton.Position = [21 59 58 23];
            app.AnlisisButton.Text = 'Análisis';

            % Create CapturaButton
            app.CapturaButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
            app.CapturaButton.ButtonPushedFcn = createCallbackFcn(app, @CapturaButtonPushed, true);
            app.CapturaButton.Position = [21 17 58 23];
            app.CapturaButton.Text = 'Captura';

            % Create Image_2
            app.Image_2 = uiimage(app.TOPGRAFODECRNEAUIFigure);
            app.Image_2.Position = [13 399 73 77];
            app.Image_2.ImageSource = fullfile(pathToMLAPP, 'Recurso 1.png');

            % Show the figure after all components are created
            app.TOPGRAFODECRNEAUIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = VenControl

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.TOPGRAFODECRNEAUIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.TOPGRAFODECRNEAUIFigure)
        end
    end
end
```

C.3. Ventana de captura

La conectividad con la interfaz desarrollada en Matlab y la cámara Thor Labs C1285R12M, se realiza a través de funciones desarrolladas y subidas como software público en la plataforma GitHub [Matthew, 2017].

En las siguientes paginas se muestra el código asociado a la ventana de captura.

```
classdef VenCaptura < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        TOPGRAFODECRNEAUIFigure   matlab.ui.Figure
        ImageMira                 matlab.ui.control.Image
        MirillaCheckBox           matlab.ui.control.CheckBox
        DetenergrabacinButton     matlab.ui.control.Button
        RecLamp                   matlab.ui.control.Lamp
        RecLampLabel              matlab.ui.control.Label
        PararButton               matlab.ui.control.Button
        InicioButton              matlab.ui.control.Button
        AtrasButton               matlab.ui.control.Button
        ComunicacionLamp          matlab.ui.control.Lamp
        ComunicacionLampLabel     matlab.ui.control.Label
        DesconectarButton         matlab.ui.control.Button
        ConectarButton            matlab.ui.control.Button
        COMSpinner                matlab.ui.control.Spinner
        COMSpinnerLabel           matlab.ui.control.Label
        CapturarButton            matlab.ui.control.Button
        GuardarButton             matlab.ui.control.Button
        TipodecapturaButtonGroup  matlab.ui.container.ButtonGroup
        VideoButton               matlab.ui.control.RadioButton
        ImagenButton              matlab.ui.control.RadioButton
        TomadeimagenLabel         matlab.ui.control.Label
        Panel                     matlab.ui.container.Panel
        Image2                    matlab.ui.control.Image
        ControlButton             matlab.ui.control.Button
        AnalisisButton            matlab.ui.control.Button
        IntensidadPanel           matlab.ui.container.Panel
        MaxSpinner                matlab.ui.control.Spinner
        MaxSpinnerLabel           matlab.ui.control.Label
        MinSpinner                matlab.ui.control.Spinner
        MinSpinnerLabel           matlab.ui.control.Label
        UIAxes                    matlab.ui.control.UIAxes
    end


    properties (Access = private)
        % Se crea las propiedades, utilizadas para las comunicaciones
        comu % Comunición serial (topografo)
        cam % Comunición de camara o sensor
    end

    methods (Access = private)
        % Funciones desarrolladas para el funcionamiento de la ventana

        function establecerComu(app)
            % Se identifica si hay o no la comunicación con el Arduino,
            % de haberse establece la comunicación serial.
            puertos=serialportlist;

            if isempty(serialportlist)
                f=warndlg("No se detecto el topografo",'Error');
                app.ComunicacionLamp.Color = 'red';
            else
                puerto=puertos(1);
                com = puerto;
                app.comu = serial(com,'BaudRate',9600,"Timeout",5);
                app.ComunicacionLamp.Color = 'red';
            end
        end

        function verArduino(app)
            % En caso de haber conexión con el topógrafo, se revisa si
            % este esta enviando la señal de captura, en caso de detectar
            % una señal de captura, se procede a verificar si se esta en
            % modo video o imagen, para luego realzar dicha captura.
            global tipoCap;
            global frame;
            global workingDir;

            dato =0;

            try
                % Se verifica si hay comunicación
                if colorangle([0 1 0],app.ComunicacionLamp.Color)==0
                    dato=fscanf(app.comu,'%d');
                    pause(0.01);

                    if tipoCap==1 && dato == 1 % La captura es una imagen

                        filter = {'*.jpg';'*.jpeg';'*.bmp';'*.png';'*.tif'};
                        [file, path] = uiputfile(filter);

                        try
                            % Se escala la imagen a 480*480
                            [F,C] = size(frame);
```

```
                    I = imresize(frame,480/F);

                    imagePath = fullfile(path,file);
                    imwrite(I,imagePath);
                catch
                    f=warndlg("Se cancelo el proceso de guardado",'Error');
                end

                % Identificar si hay o no captura
                while dato == 1
                    % Se captura la imagen
                    dato=fscanf(app.comu,'%d');
                    frame = obtenerIma(app);
                    axis(app.UIAxes, 'image');
                    imshow(frame,'Parent',app.UIAxes);
                    pause(0.02);
                end

            elseif tipoCap ==2 && dato == 1 % La captura es un video

                % Encender el simbolo de grabando "Rec"
                app.RecLamp.Visible="on";
                app.RecLampLabel.Visible="on";

                % Identificar si hay o no captura, de ser asi se
                % comienza la captura
                cont = 0;
                while dato ==1
                    dato=fscanf(app.comu,'%d');

                    % Toma imagenes que componen el video
                    frame = obtenerIma(app);
                    axis(app.UIAxes, 'image');
                    imshow(frame,'Parent',app.UIAxes);
                    pause(0.02);

                    filename = [sprintf('%03d',cont) '.jpg'];
                    fullname = fullfile(workingDir,'images',filename);

                    % escalamos para que sea de 480*480
                    [F,C] = size(frame);
                    I = imresize(frame,480/F);

                    imwrite(I,fullname);
                    cont=cont+1;

                end

                % Desarrollo del video con las imagenes
                filter = {'*.avi'};
                [file, path] = uiputfile(filter);

                imageNames = dir(fullfile(workingDir,'images','*.jpg'));
                imageNames = {imageNames.name}';

                videoPath = fullfile(path,file);
                outputVideo = VideoWriter(videoPath);
                outputVideo.FrameRate = 7;
                open(outputVideo)

                for ii = 1:length(imageNames)
                   img = imread(fullfile(workingDir,'images',imageNames{ii}));
                   writeVideo(outputVideo,img)

                end

                close(outputVideo);

                try
                    % Eliminar la carpeta de las imagenes
                    rmdir(workingDir,'s');
                catch

                end
                app.RecLamp.Visible="off";
                app.RecLampLabel.Visible="off";

            end
        end

    catch
        if colorangle([0 1 0],app.ComunicacionLamp.Color)==0
            app.ComunicacionLamp.Color = 'red';
        end
    end
end

function image = obtenerIma(app)
    % Se adquiere la imagen percibida por la cámara, y se da
    % como variable de salida
```

```
        global hCam
        global width
        global height

        % Verificación de la existencia de parámetros necesarios
        % para la captura
        if exist('ThorCamSaveSettings.mat', 'file') == 2
            maxgain = GetMaxGain(hCam);
            gain = GetGain(hCam);
            expset = GetExposureInfo(hCam);
            gboost = SetGainBoost(hCam,-1);
            cm = 1;

            settings = load('ThorCamSaveSettings.mat');
            gain = settings.sgain;
            result = SetGain(hCam,gain);
            expset(1) = settings.sexp;
            result = SetExposure(hCam,expset(1));
            gboost = settings.sgainb;
            SetGainBoost(hCam,gboost);
            cm = settings.scm;
        end

        % Uso de código externo "GetImage" para la obtención de la imagen
        image = GetImage(hCam,width,height);

        % Configuraciones de la imagen, orientación, tamaño y brillo
        image = flipud(rot90(reshape(image,1280,1024)));

        targetSize = [970 970];
        centro = [490,644]; % poner en y,x
        r = centerCropWindow2d(centro*2,targetSize);
        %figure()
        image = imcrop(image,r);

        low = (app.MinSpinner.Value)/255;
        high = (app.MaxSpinner.Value)/255;
        image = imadjust(image,[low high]);

        % Escalamos para que sea de 480*480
        [F,C] = size(image);
        image = imresize(image,480/F);

    end
end


% Callbacks that handle component events
methods (Access = private)

    % Code that executes after component creation
    function startupFcn(app)
        % Se inicializa las variables globales del sistema y se
        % establece la comunicación con el topógrafo, esto en caso de
        % ser posible
        global hayCam
        global runLoop;
        global tipoCap
        global parar

        parar =0;
        tipoCap = 1;
        runLoop=true;
        hayCam = false;
        puertos=serialportlist;

        if isempty(serialportlist)
            f=warndlg("No se detecto el topografo",'Error');
            app.ComunicacionLamp.Color = 'red';
        else
            puerto=puertos(1);
            com = puerto;
            app.comu = serial(com,'BaudRate',9600,"Timeout",5);
            app.ComunicacionLamp.Color = 'red';
            offLeds(app);
        end

    end

    % Button pushed function: ConectarButton
    function ConectarButtonPushed(app, event)
        % Se estable o reinicia la comunicación serial con el
        % topógrafo, y se permite abrir el ciclo de captura
        global runLoop
        runLoop=true;

        try
            fclose(app.comu);
        catch
```

```
    end
    if isempty(app.comu)
        establecerComu(app);
        try
            fopen(app.comu);
            app.ComunicacionLamp.Color = 'green';
        catch
            f=warndlg("No se detecto el topografo",'Error');
        end
    else
        try
            fopen(app.comu);
            app.ComunicacionLamp.Color = 'green';
        catch
            f=warndlg("No se detecto el topografo en el puerto",'Error');
        end
    end

end

% Button pushed function: DesconectarButton
function DesconectarButtonPushed(app, event)
    % Se cierra y elimina la comunicación serial con el topógrafo
    global runLoop
    runLoop=false;

    try
        fclose(app.comu);
    catch

    end
    app.ComunicacionLamp.Color = 'red';
    clear all;

end

% Button pushed function: AnalisisButton
function AnalisisButtonPushed(app, event)
    % Se cierra la ventana de captura, la comunicación, el ciclo de
    % video, junto con la comunicacion con la camara y se abre la
    % ventana de análisis. Se elimina la carpeta temporal de
    % imagenes tomadas.
    global runLoop
    runLoop=false;

    try
        fclose(app.comu);
    catch

    end
    try
        % Eliminar la carpeta de las imagenes
        rmdir(workingDir,'s');
    catch

    end
    delete(app.comu);
    delete(app)
    clear all;
    VenAnalisis

end

% Button pushed function: ControlButton
function ControlButtonPushed(app, event)
    % Se cierra la ventana de captura, la comunicación, el ciclo de
    % video, junto con la comunicacion con la camara y se abre la
    % ventana de control. Se elimina la carpeta temporal de
    % imagenes tomadas.
    global runLoop
    runLoop=false;

    try
        fclose(app.comu);
    catch

    end
    try
        % Eliminar la carpeta de las imagenes
        rmdir(workingDir,'s');
    catch

    end
    delete(app.comu);
    delete(app);
    clear cam;
    clear all;
    VenControl
end
```

```
% Button pushed function: AtrasButton
function AtrasButtonPushed(app, event)
    % Se cierra la ventana de captura, la comunicación, el ciclo de
    % video, junto con la comunicacion con la camara y se abre la
    % ventana de inicio. Se elimina la carpeta temporal de
    % imagenes tomadas.
    global runLoop
    runLoop=false;

    try
        fclose(app.comu);
    catch

    end
    try
        % Eliminar la carpeta de las imagenes
        rmdir(workingDir,'s');
    catch

    end
    delete(app.comu);
    delete(app);
    clear cam;
    clear all;
    Inicio
end

% Close request function: TOPGRAFODECRNEAUIFigure
function TOPGRAFODECRNEAUIFigureCloseRequest(app, event)
    % Al cerrar la aplicación de cierra el ciclo que pueda estar
    % en funcionamiento, así como variables y la conexiones tanto
    % con el Arduino como con la cámara.
    global runLoop
    runLoop=false;

    try
        fclose(app.comu);
        cancel(f)
    catch

    end
    try
        % Eliminar la carpeta de las imagenes
        rmdir(workingDir,'s');
    catch

    end

    delete(app.comu);
    delete(app);
    clear all;
    clear cam;
    clearvars;
end

% Selection changed function: TipodecapturaButtonGroup
function TipodecapturaButtonGroupSelectionChanged(app, event)
    % Se le asigna el valor a la variable global tipoCap, 1 si es
    % imagen y 2 si es video. En caso de ser video se crea la
    % carpeta temporal donde se guardarán los frames que harán
    % parte del mismo, mientras que si es imagen esta carpeta será
    % eliminada
    selectedButton = app.TipodecapturaButtonGroup.SelectedObject;
    global tipoCap
    global workingDir

    if app.VideoButton.Value == 1
        tipoCap = 2;

        workingDir = tempname;
        mkdir(workingDir)
        mkdir(workingDir,'images')

    elseif  app.ImagenButton.Value == 1
        tipoCap = 1;
        try
            % Eliminar la carpeta de las imagenes
            rmdir(workingDir,'s');
        catch

        end
    end

end

% Button pushed function: InicioButton
function InicioButtonPushed(app, event)
    % Se establece la cámara, encaso ser exitoso el vínculo se
    % inicia el ciclo de visualización de imagen, en el que ubica
    % la imanen obtenida en el UIaxes en el tamaño correspondiente
```

```
    global frame;
    global hayCam;
    global runLoop;
    runLoop=true;

    if isempty(app.cam)

        hayCam = establecerCam(app);
    end

    if hayCam

        app.CapturarButton.Visible="on";
        app.MirillaCheckBox.Visible="on";
        while runLoop == true % volverlo false al cerrar y etc
            % Establecer la camara
            try
                I = obtenerIma(app);
                axis(app.UIAxes, 'image');
                frame=I;
                imshow(frame,'Parent',app.UIAxes);
                pause(0.03);

                verArduino(app);
            catch
                f=warndlg("No se detecto la camara",'Error');
                break;
            end
            if exist('runLoop','var') == 0
                break;
            end

        end
    else
        f=warndlg("No se detecto la camara",'Error');
    end


end

% Button pushed function: PararButton
function PararButtonPushed(app, event)
    % Se cambia el valor de la variable que maneja el ciclo de
    % visualización de la cámara, para que este se detenga
    global runLoop;
    runLoop=false;

end

% Button pushed function: CapturarButton
function CapturarButtonPushed(app, event)
    % Se identifica que tipo de captura se desea realizar, en caso
    % de ser imagen, la imagen mostrada se guarda, mientras que,
    % si es video se inicia un ciclo en el que se irán guardando
    % los frames del mismo hasta que el usuario detenga dicha
    % grabación. En ambos casos se le permite al usuario elegir la
    % carpeta de destino de los archivos, así como la extensión de
    % los mismos.
    global frame
    global tipoCap
    global parar
    global workingDir

    if tipoCap==1 % La captura es una foto

        filter = {'*.jpg';'*.jpeg';'*.bmp';'*.png';'*.tif'};
        [file, path] = uiputfile(filter);

        try
            % Escalamos la imagen a 480*480
            [F,C] = size(frame);
            I = imresize(frame,480/F);

            imagePath = fullfile(path,file);
            imwrite(I,imagePath);
        catch
            f=warndlg("Se cancelo el proceso de guardado",'Error');
        end

    elseif tipoCap ==2 % La captura es un video
        app.DetenergrabacinButton.Visible = "on";
        app.RecLamp.Visible="on";
        app.RecLampLabel.Visible="on";

        cont = 1;

        % Creacion de carpeta temporal en caso de que esta no
        % exista
        if exist(workingDir) == 0
            workingDir = tempname;
```

```
            mkdir(workingDir);
            mkdir(workingDir,'images');
            parar = 0;
        end

        while parar == 0

            I = obtenerIma(app);
            axis(app.UIAxes, 'image');
            frame=I;
            imshow(frame,'Parent',app.UIAxes);
            pause(0.03);

            if parar == 1
                parar = 0;
                break;
            end

            try
                filename = [sprintf('%03d',cont) '.jpg'];
                fullname = fullfile(workingDir,'images',filename);

                imwrite(I,fullname);
                cont=cont+1;
            catch
                f=warndlg("Hubo un erro con el video",'Error');
            end

        end

        try
            % Eliminar la carpeta de las imagenes
            rmdir(workingDir,'s');
        catch
        end

    end

end

% Button pushed function: GuardarButton
function GuardarButtonPushed(app, event)
    % Se guarda la imagen que se está mostrando en el Uiaxes,
    % permitiendo que el usuario elegir la carpeta de destino de
    % los archivos, así como la extensión de los mismos.
    global frame;

    filter = {'*.jpg';'*.jpeg';'*.bmp';'*.png';'*.tif'};
    [file, path] = uiputfile(filter);

    try
        % Escalamos la imagen a 480*480
        [F,C] = size(frame);
        I = imresize(frame,480/F);

        imagePath = fullfile(path,file);
        imwrite(I,imagePath);
    catch
        f=warndlg("Se cancelo el proceso de guardado",'Error');
    end

end

% Button pushed function: DetenergrabacinButton
function DetenergrabacinButtonPushed(app, event)
    % Se detiene la grabación del video, permitiéndole al usuario
    % identificar la carpeta de destino del video, para una vez
    % elegida, crear el video a partir de los frames obtenidos y
    % almacenarlos en el formato correspondiente
    global parar
    global workingDir
    parar = 1;

    filter = {'*.avi'};
    [file, path] = uiputfile(filter);

    try

        imageNames = dir(fullfile(workingDir,'images','*.jpg'));
        imageNames = {imageNames.name}';

        videoPath = fullfile(path,file);
        outputVideo = VideoWriter(videoPath);
        %outputVideo.FrameRate = shuttleVideo.FrameRate;
        outputVideo.FrameRate = 7;
        open(outputVideo)

        for ii = 1:length(imageNames)
           img = imread(fullfile(workingDir,'images',imageNames{ii}));
           writeVideo(outputVideo,img)
```

```
            end

            close(outputVideo);
        catch
            f=warndlg("Se cancelo el proceso de guardado",'Error');
        end

        app.RecLamp.Visible="off";
        app.RecLampLabel.Visible="off";
        app.DetenergrabacinButton.Visible = "off";

    end

    % Value changed function: MirillaCheckBox
    function MirillaCheckBoxValueChanged(app, event)
        % Se ubica la mirilla en el centro de la imagen y se cambia su
        % estado de visible a no visible o viceversa

        value = app.MirillaCheckBox.Value;
        if value ==1
            % Se calculan las coordenadas para que la mirilla quede en
            % el centro, donde el punto 1,1 es en la esquina inferior
            % izquierda
            framePosition=app.UIAxes.Position;
            w=framePosition(3)/4;
            h=framePosition(4)/4;
            x=framePosition(1)+(framePosition(3)/2)-w/2;
            y=framePosition(2)+(framePosition(4)/2)-h/2;

            app.ImageMira.Position = [x y w h];

            app.ImageMira.Visible="on";
        else
            app.ImageMira.Visible="off";
        end
    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Get the file path for locating images
        pathToMLAPP = fileparts(mfilename('fullpath'));

        % Create TOPGRAFODECRNEAUIFigure and hide until all components are created
        app.TOPGRAFODECRNEAUIFigure = uifigure('Visible', 'off');
        app.TOPGRAFODECRNEAUIFigure.Color = [1 1 1];
        app.TOPGRAFODECRNEAUIFigure.Position = [100 100 912 697];
        app.TOPGRAFODECRNEAUIFigure.Name = 'TOPÓGRAFO DE CÓRNEA';
        app.TOPGRAFODECRNEAUIFigure.Resize = 'off';
        app.TOPGRAFODECRNEAUIFigure.CloseRequestFcn = createCallbackFcn(app, @TOPGRAFODECRNEAUIFigureCloseRequest, true);

        % Create UIAxes
        app.UIAxes = uiaxes(app.TOPGRAFODECRNEAUIFigure);
        app.UIAxes.Toolbar.Visible = 'off';
        app.UIAxes.XColor = 'none';
        app.UIAxes.YColor = 'none';
        app.UIAxes.ZColor = 'none';
        app.UIAxes.XGrid = 'on';
        app.UIAxes.YGrid = 'on';
        app.UIAxes.Position = [327 62 499 429];

        % Create IntensidadPanel
        app.IntensidadPanel = uipanel(app.TOPGRAFODECRNEAUIFigure);
        app.IntensidadPanel.BorderType = 'none';
        app.IntensidadPanel.Title = 'Intensidad';
        app.IntensidadPanel.BackgroundColor = [1 1 1];
        app.IntensidadPanel.FontSize = 18;
        app.IntensidadPanel.Position = [168 312 134 99];

        % Create MinSpinnerLabel
        app.MinSpinnerLabel = uilabel(app.IntensidadPanel);
        app.MinSpinnerLabel.HorizontalAlignment = 'right';
        app.MinSpinnerLabel.FontSize = 16;
        app.MinSpinnerLabel.Position = [11 39 31 22];
        app.MinSpinnerLabel.Text = 'Min';

        % Create MinSpinner
        app.MinSpinner = uispinner(app.IntensidadPanel);
        app.MinSpinner.Limits = [0 255];
        app.MinSpinner.FontSize = 16;
        app.MinSpinner.Position = [59 39 76 22];
        app.MinSpinner.Value = 3;

        % Create MaxSpinnerLabel
        app.MaxSpinnerLabel = uilabel(app.IntensidadPanel);
        app.MaxSpinnerLabel.HorizontalAlignment = 'right';
```

```
app.MaxSpinnerLabel.FontSize = 16;
app.MaxSpinnerLabel.Position = [11 2 35 22];
app.MaxSpinnerLabel.Text = 'Max';

% Create MaxSpinner
app.MaxSpinner = uispinner(app.IntensidadPanel);
app.MaxSpinner.Limits = [0 255];
app.MaxSpinner.FontSize = 16;
app.MaxSpinner.Position = [58 2 77 22];
app.MaxSpinner.Value = 15;

% Create Panel
app.Panel = uipanel(app.TOPGRAFODECRNEAUIFigure);
app.Panel.BackgroundColor = [0 0.4471 0.7412];
app.Panel.Position = [-1 0 146 698];

% Create AnalisisButton
app.AnalisisButton = uibutton(app.Panel, 'push');
app.AnalisisButton.ButtonPushedFcn = createCallbackFcn(app, @AnalisisButtonPushed, true);
app.AnalisisButton.Position = [23 21 100 23];
app.AnalisisButton.Text = 'Analisis';

% Create ControlButton
app.ControlButton = uibutton(app.Panel, 'push');
app.ControlButton.ButtonPushedFcn = createCallbackFcn(app, @ControlButtonPushed, true);
app.ControlButton.Position = [22 62 100 23];
app.ControlButton.Text = 'Control';

% Create Image2
app.Image2 = uiimage(app.Panel);
app.Image2.Position = [23 556 100 100];
app.Image2.ImageSource = fullfile(pathToMLAPP, 'Recurso 1.png');

% Create TomadeimagenLabel
app.TomadeimagenLabel = uilabel(app.TOPGRAFODECRNEAUIFigure);
app.TomadeimagenLabel.FontSize = 30;
app.TomadeimagenLabel.FontWeight = 'bold';
app.TomadeimagenLabel.Position = [385 587 242 39];
app.TomadeimagenLabel.Text = 'Toma de imagen';

% Create TipodecapturaButtonGroup
app.TipodecapturaButtonGroup = uibuttongroup(app.TOPGRAFODECRNEAUIFigure);
app.TipodecapturaButtonGroup.SelectionChangedFcn = createCallbackFcn(app, @TipodecapturaButtonGroupSelectionChanged, true);
app.TipodecapturaButtonGroup.BorderType = 'none';
app.TipodecapturaButtonGroup.Title = 'Tipo de captura';
app.TipodecapturaButtonGroup.BackgroundColor = [1 1 1];
app.TipodecapturaButtonGroup.FontSize = 18;
app.TipodecapturaButtonGroup.Position = [168 436 141 84];

% Create ImagenButton
app.ImagenButton = uiradiobutton(app.TipodecapturaButtonGroup);
app.ImagenButton.Text = 'Imagen';
app.ImagenButton.FontSize = 16;
app.ImagenButton.Position = [11 32 75 22];
app.ImagenButton.Value = true;

% Create VideoButton
app.VideoButton = uiradiobutton(app.TipodecapturaButtonGroup);
app.VideoButton.Text = 'Video';
app.VideoButton.FontSize = 16;
app.VideoButton.Position = [11 1 65 22];

% Create GuardarButton
app.GuardarButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.GuardarButton.ButtonPushedFcn = createCallbackFcn(app, @GuardarButtonPushed, true);
app.GuardarButton.Position = [673 496 100 23];
app.GuardarButton.Text = 'Guardar';

% Create CapturarButton
app.CapturarButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.CapturarButton.ButtonPushedFcn = createCallbackFcn(app, @CapturarButtonPushed, true);
app.CapturarButton.Visible = 'off';
app.CapturarButton.Position = [212 119 100 23];
app.CapturarButton.Text = 'Capturar';

% Create COMSpinnerLabel
app.COMSpinnerLabel = uilabel(app.TOPGRAFODECRNEAUIFigure);
app.COMSpinnerLabel.HorizontalAlignment = 'right';
app.COMSpinnerLabel.Visible = 'off';
app.COMSpinnerLabel.Position = [598 22 33 22];
app.COMSpinnerLabel.Text = 'COM';

% Create COMSpinner
app.COMSpinner = uispinner(app.TOPGRAFODECRNEAUIFigure);
app.COMSpinner.Visible = 'off';
app.COMSpinner.Position = [646 22 100 22];
app.COMSpinner.Value = 5;

% Create ConectarButton
app.ConectarButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
```

```
            app.ConectarButton.ButtonPushedFcn = createCallbackFcn(app, @ConectarButtonPushed, true);
            app.ConectarButton.Position = [325 21 100 23];
            app.ConectarButton.Text = 'Conectar';

            % Create DesconectarButton
            app.DesconectarButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
            app.DesconectarButton.ButtonPushedFcn = createCallbackFcn(app, @DesconectarButtonPushed, true);
            app.DesconectarButton.Position = [456 21 100 23];
            app.DesconectarButton.Text = 'Desconectar';

            % Create ComunicacionLampLabel
            app.ComunicacionLampLabel = uilabel(app.TOPGRAFODECRNEAUIFigure);
            app.ComunicacionLampLabel.HorizontalAlignment = 'right';
            app.ComunicacionLampLabel.Position = [184 23 82 22];
            app.ComunicacionLampLabel.Text = 'Comunicacion';

            % Create ComunicacionLamp
            app.ComunicacionLamp = uilamp(app.TOPGRAFODECRNEAUIFigure);
            app.ComunicacionLamp.Position = [281 23 20 20];
            app.ComunicacionLamp.Color = [1 0 0];

            % Create AtrasButton
            app.AtrasButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
            app.AtrasButton.ButtonPushedFcn = createCallbackFcn(app, @AtrasButtonPushed, true);
            app.AtrasButton.Position = [789 22 100 23];
            app.AtrasButton.Text = 'Atras';

            % Create InicioButton
            app.InicioButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
            app.InicioButton.ButtonPushedFcn = createCallbackFcn(app, @InicioButtonPushed, true);
            app.InicioButton.Position = [382 496 100 23];
            app.InicioButton.Text = 'Inicio';

            % Create PararButton
            app.PararButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
            app.PararButton.ButtonPushedFcn = createCallbackFcn(app, @PararButtonPushed, true);
            app.PararButton.Position = [527 496 100 23];
            app.PararButton.Text = 'Parar';

            % Create RecLampLabel
            app.RecLampLabel = uilabel(app.TOPGRAFODECRNEAUIFigure);
            app.RecLampLabel.HorizontalAlignment = 'right';
            app.RecLampLabel.Visible = 'off';
            app.RecLampLabel.Position = [246 209 27 22];
            app.RecLampLabel.Text = 'Rec';

            % Create RecLamp
            app.RecLamp = uilamp(app.TOPGRAFODECRNEAUIFigure);
            app.RecLamp.Visible = 'off';
            app.RecLamp.Position = [288 209 20 20];
            app.RecLamp.Color = [1 0 0];

            % Create DetenergrabacinButton
            app.DetenergrabacinButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
            app.DetenergrabacinButton.ButtonPushedFcn = createCallbackFcn(app, @DetenergrabacinButtonPushed, true);
            app.DetenergrabacinButton.Visible = 'off';
            app.DetenergrabacinButton.Position = [197 165 115 23];
            app.DetenergrabacinButton.Text = 'Detener grabación';

            % Create MirillaCheckBox
            app.MirillaCheckBox = uicheckbox(app.TOPGRAFODECRNEAUIFigure);
            app.MirillaCheckBox.ValueChangedFcn = createCallbackFcn(app, @MirillaCheckBoxValueChanged, true);
            app.MirillaCheckBox.Visible = 'off';
            app.MirillaCheckBox.Text = 'Mirilla';
            app.MirillaCheckBox.FontSize = 14;
            app.MirillaCheckBox.Position = [825 266 59 22];

            % Create ImageMira
            app.ImageMira = uiimage(app.TOPGRAFODECRNEAUIFigure);
            app.ImageMira.Visible = 'off';
            app.ImageMira.Position = [809 141 100 100];
            app.ImageMira.ImageSource = fullfile(pathToMLAPP, 'MiraBlanca.png');

            % Show the figure after all components are created
            app.TOPGRAFODECRNEAUIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = VenCaptura

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.TOPGRAFODECRNEAUIFigure)
```

```
            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.TOPGRAFODECRNEAUIFigure)
        end
    end
end
```

C.4. Ventana de análisis

El código a presentado en las siguientes paginas corresponde únicamente al código desarrollado para el funcionamiento de la ventana que se aprecia en la Figura 4-6-c, mas no para el funcionamiento de la ventana presentada en la Figura 4-11.

```
classdef VenAnalisis < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        TOPGRAFODECRNEAUIFigure  matlab.ui.Figure
        IntensidadPanel          matlab.ui.container.Panel
        MaxSpinner               matlab.ui.control.Spinner
        MaxSpinnerLabel          matlab.ui.control.Label
        MinSpinner               matlab.ui.control.Spinner
        MinSpinnerLabel          matlab.ui.control.Label
        AnalizarButton           matlab.ui.control.Button
        ImageMira                matlab.ui.control.Image
        MirillaCheckBox          matlab.ui.control.CheckBox
        SubirarchivoButton       matlab.ui.control.Button
        Adelante2Button          matlab.ui.control.Button
        Atras2Button             matlab.ui.control.Button
        GuardarframeButton       matlab.ui.control.Button
        Adelante1Button          matlab.ui.control.Button
        Atras1Button             matlab.ui.control.Button
        AtrasButton              matlab.ui.control.Button
        AnalisisLabel            matlab.ui.control.Label
        Panel                    matlab.ui.container.Panel
        CapturaButton            matlab.ui.control.Button
        ControlButton            matlab.ui.control.Button
        Image                    matlab.ui.control.Image
        UIAxes                   matlab.ui.control.UIAxes
    end


    methods (Access = private)
        % Funciones desarrolladas para el funcionamiento de la ventana

        function PonTxt(app,path,file)
            % Se almacena la ruta de la imagen a analizar en un archivo txt
            path = [path file];
            fileID = fopen('PathImage.txt','w');
            fprintf(fileID,'%s%d%s',path);
            fclose(fileID);
        end

        function LimTxt(app)
            % Se limpia el archivo txt de ruta de la imagen a analizar
            msm = "";
            fileID = fopen('PathImage.txt','w');
            fprintf(fileID,'%s%d%s',msm);
            fclose(fileID);
        end

        function imagen = comprobarImagen(app,ima)
            [F,C,d] = size(ima);

            if F == 1024 && C == 1280
                targetSize = [970 970];
                centro = [490,644]; % poner en y,x
                r = centerCropWindow2d(centro*2,targetSize);
                figure()
                ima = imcrop(ima,r);

                % Se escala la imagen a 480*480
                imagen = imresize(ima,480/F);

            else

                if C>F
                    targetSize = [F F];
                    r = centerCropWindow2d(size(ima),targetSize);
                    ima = imcrop(ima,r);
                    T=F;
                elseif F>C
                    targetSize = [C C];
                    r = centerCropWindow2d(size(ima),targetSize);
                    ima = imcrop(ima,r);
                    T=C;
                else
                    T=F;
                end

                % Se escala la imagen a 480*480
                imagen = imresize(ima,480/T);
            end
        end

    end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)

        end

        % Button pushed function: AtrasButton
```

```
function AtrasButtonPushed(app, event)
    % Se abre la ventana de inicio, se cierra la ventana de
    % analisis y se limpia el archivo txt
    Inicio
    delete(app)
    LimTxt(app);
end

% Button pushed function: ControlButton
function ControlButtonPushed(app, event)
    % Se abre la ventana de control, se cierra la ventana de
    % analisis y se limpia el archivo txt
    VenControl
    delete(app)
    LimTxt(app);
end

% Button pushed function: Adelante1Button
function Adelante1ButtonPushed(app, event)
    % Se verifica la existencia de que se esté proyectando el frame
    % de un video, de ser así, este se cambia por el segundo frame
    % siguiente, de no ser posible se cambia por el primer frame
    global nFrame;
    global archivo;
    global filename;
    global tipoArchivo;

    video1 = archivo;
    n=2;
    framesVideo = video1;

    if isequal(filename,0) || isempty(video1)
        f=warndlg("No se ha cargado ningun video",'Error');

    elseif tipoArchivo ==".mp4" || tipoArchivo == ".avi"
        Cantframes=framesVideo.numFrames;
        if isempty(nFrame)
            nFrame=1;
        end

        if nFrame >= Cantframes-n
            nFrame=1;
        else
            nFrame=nFrame+n;
        end

        frame = read(framesVideo,nFrame);
            imshow(frame,'Parent',app.UIAxes);

    end
end

% Button pushed function: CapturaButton
function CapturaButtonPushed(app, event)
    % Se abre la ventana de captrua, se cierra la ventana de
    % analisis y se limpia el archivo txt
    VenCaptura
    delete(app)
    LimTxt(app);
end

% Button pushed function: GuardarframeButton
function GuardarframeButtonPushed(app, event)
    % Se extrae la imagen del frame del video que se esta
    % visualizando para guardarlo como un archivo nuevo
    global nFrame;
    global archivo;
    global tipoArchivo;

    if tipoArchivo ==".mp4" || tipoArchivo == ".avi"

        video1=archivo;
        framesVideo = video1;
        frame = read(framesVideo,nFrame);
        filter = {'*.jpg';'*.jpeg';'*.bmp';'*.png';'*.tif'};
        [file, path] = uiputfile(filter);

        imagePath = fullfile(path,file);
        imwrite(frame,imagePath);
    end
end

% Button pushed function: Atras1Button
function Atras1ButtonPushed(app, event)
    % Se verifica la existencia de que se esté proyectando el frame
    % de un video, de ser así, este se cambia por el segundo frame
    % anterior, de no ser posible se cambia por el ultimo frame
    global nFrame;
    global archivo;
    global filename;
    global tipoArchivo;

    video1 = archivo;
    n=2;
    framesVideo = video1;

    if isequal(filename,0) || isempty(video1)
```

```
        f=warndlg("No se ha cargado ningun video",'Error');

    elseif tipoArchivo ==".mp4" || tipoArchivo == ".avi"
        Cantframes=framesVideo.numFrames;
        if isempty(nFrame)
            nFrame=1;
        end

        if nFrame-n <= 0
            nFrame=Cantframes;
        else
            nFrame=nFrame-n;
        end

        frame = read(framesVideo,nFrame);
            imshow(frame,'Parent',app.UIAxes);
    end
end

% Button pushed function: Adelante2Button
function Adelante2ButtonPushed(app, event)
    % Se verifica la existencia de que se esté proyectando el frame
    % de un video, de ser así, este se cambia por el decimo frame
    % siguiente, de no ser posible se cambia por el primer frame
    global nFrame;
    global archivo;
    global filename;
    global tipoArchivo;

    video1 = archivo;
    n=10;
    framesVideo = video1;

    if isequal(filename,0) || isempty(video1)
        f=warndlg("No se ha cargado ningun video",'Error');

    elseif tipoArchivo ==".mp4" || tipoArchivo == ".avi"
        Cantframes=framesVideo.numFrames;

        if isempty(nFrame)
            nFrame=1;
        end

        if nFrame >= Cantframes-n
            nFrame=1;
        else
            nFrame=nFrame+n;
        end

        frame = read(framesVideo,nFrame);
            imshow(frame,'Parent',app.UIAxes);
    end
end

% Button pushed function: Atras2Button
function Atras2ButtonPushed(app, event)
    % Se verifica la existencia de que se esté proyectando el frame
    % de un video, de ser así, este se cambia por el decimo frame
    % anterior, de no ser posible se cambia por el ultimo frame
    global nFrame;
    global archivo;
    global filename;
    global tipoArchivo;

    video1 = archivo;
    n=10;
    framesVideo = video1;

    if isequal(filename,0) || isempty(video1)
        f=warndlg("No se ha cargado ningun video",'Error');

    elseif tipoArchivo ==".mp4" || tipoArchivo == ".avi"
        Cantframes=framesVideo.numFrames;
        if isempty(nFrame)
            nFrame=1;
        end

        if nFrame-n <= 0
            nFrame=Cantframes;
        else
            nFrame=nFrame-n;
        end

        frame = read(framesVideo,nFrame);
            imshow(frame,'Parent',app.UIAxes);
    end
end

% Button pushed function: SubirarchivoButton
function SubirarchivoButtonPushed(app, event)
    % Se le permite al usuario seleccionar un archivo de tipo video
    % o imagen, para luego ubicarlo en el UIaxes con el tamaño
    % adecuado, en el caso de ser un video se mostrara el primer
    % frame de este
    global archivo
    global tipoArchivo
    global filename
```

```
    [filename pathname] = uigetfile({'*.jpg;*.jpeg;*.bmp;*.png;*.mp4;*.avi;*.tif'},'File Selector');
    if isequal(filename,0)
        disp('Seleccion cancelada');
        f=warndlg("No se ha cargado ningun video",'Error');
    else
        [filepath,name,ext] = fileparts(filename);
        tipoArchivo = convertCharsToStrings(ext);
        tipoArchivo = lower(tipoArchivo);
        if tipoArchivo ==".mp4" || tipoArchivo == ".avi"

            app.UIAxes.Position = [119,49,552,369];
            app.UIAxes.PlotBoxAspectRatio = [1,0.6238095238095238,0.6238095238095238];

            videoPath = fullfile(pathname,filename);
            archivo=VideoReader(videoPath);
            frame1 = read(archivo,1);
                imshow(frame1,'Parent',app.UIAxes);

            app.GuardarframeButton.Visible ="on";
            app.Adelante1Button.Visible = "on";
            app.Adelante2Button.Visible="on";
            app.Atras1Button.Visible="on";
            app.Atras2Button.Visible="on";
            app.AnalizarButton.Visible="on";
            app.MirillaCheckBox.Visible="on";
            app.IntensidadPanel.Visible="on";

        elseif tipoArchivo == ".jpg" || tipoArchivo == ".jpeg" || tipoArchivo == ".bmp" || tipoArchivo == ".png" || tipoArchivo == ".tif"
            app.UIAxes.Position = [82,28,550,366];
            app.UIAxes.PlotBoxAspectRatio = [1,0.6241467,0.6241467];

            fullpathname = strcat(pathname,filename);
            archivo=imread(fullpathname);
            imshow(archivo,'Parent',app.UIAxes);

            app.GuardarframeButton.Visible ="off";
            app.Adelante1Button.Visible = "off";
            app.Adelante2Button.Visible="off";
            app.Atras1Button.Visible="off";
            app.Atras2Button.Visible="off";
            app.AnalizarButton.Visible="on";
            app.MirillaCheckBox.Visible="on";
            app.IntensidadPanel.Visible="on";

            PonTxt(app,pathname,filename);

        else
            f=warndlg("No se reconoce el archivo",'Error');
        end


    end
end

% Button pushed function: AnalizarButton
function AnalizarButtonPushed(app, event)
    % Se identifica que tipo de archivo está viendo el usuario, si
    % es el frame de un video o una imagen en un formato diferente,
    % se pide guardar dicha imagen como corresponde, para luego
    % almacenar su ruta en el txt, y abrirla en el programa
    % “ftopotchZ”, la cual corresponde a un código externo.
    global archivo;
    global tipoArchivo;
    global nFrame;
    global imagen;

    if tipoArchivo ==".mp4" || tipoArchivo == ".avi"

        video1=archivo;
        framesVideo = video1;
        frame = read(framesVideo,nFrame);

        % Se analiza la imagen
        I = comprobarImagen(app,frame);

        % Agregarle los cambios en la intensidad luminica
        low = (app.MinSpinner.Value)/255;
        high = (app.MaxSpinner.Value)/255;
        I = imadjust(I,[low high]);

        % Se guarda la imagen
        filter = {'*.jpg';'*.jpeg';'*.bmp';'*.png';'*.tif'};
        [file, path] = uiputfile(filter);
        imagePath = fullfile(path,file);
        imwrite(I,imagePath);

        % Se pone la ruta en txt
        PonTxt(app,path,file);

    elseif tipoArchivo == ".jpg" || tipoArchivo == ".jpeg" || tipoArchivo == ".bmp" || tipoArchivo == ".png" || tipoArchivo == ".tif"
        imagen = archivo;
        frame = archivo;

        % Se pide guardar la imagen si no cumple con el tamaño 480*480
        % o si se le ha hecho algun ajuste de intensidad
        [F,C] = size(frame);
```

```
        if C ~= 480 || F~=480 || app.MaxSpinner.Value ~= 255 || app.MinSpinner.Value ~= 0
            % Se analiza la imagen
            I = comprobarImagen(app,frame);

            % Se guarda la imagen
            filter = {'*.jpg';'*.jpeg';'*.bmp';'*.png';'*.tif'};
            [file, path] = uiputfile(filter);
            imagePath = fullfile(path,file);
            imwrite(I,imagePath);

            % Se pone la ruta en txt
            PonTxt(app,path,file);
        end

    else
        f=warndlg("No se reconoce el archivo seleccionado",'Error');
    end

    % Abrir la ventana de analisis
    ftopotchZ();
end

% Value changed function: MirillaCheckBox
function MirillaCheckBoxValueChanged(app, event)
    % Se ubica la mirilla en el centro de la imagen y se cambia su
    % estado de visible a no visible o viceversa
    global archivo;
    global tipoArchivo;
    global nFrame;
    value = app.MirillaCheckBox.Value;

    if tipoArchivo == ".mp4" || tipoArchivo == ".avi"
        video1=archivo;
        framesVideo = video1;
        frame = read(framesVideo,nFrame);
        [F,C,d]=size(archivo);

    elseif tipoArchivo == ".jpg" || tipoArchivo == ".jpeg" || tipoArchivo == ".bmp" || tipoArchivo == ".png" || tipoArchivo == ".tif"
        [F,C,d]=size(archivo);
    end

    if value == 1
        if F == 1024 && C == 1280
            framePosition=app.UIAxes.Position; % El punto 1,1 es en la esquina inferior izquierda

            w=framePosition(3)/4;
            h=framePosition(4)/4;
            x=framePosition(1)+(framePosition(3)/2)-w/2;
            y=framePosition(2)+(framePosition(4)/2)-h/2;

            app.ImageMira.Position = [x y w h];

            app.ImageMira.Visible="on";

        else
            framePosition=app.UIAxes.Position; % El punto 1,1 es en la esquina inferior izquierda

            w=framePosition(3)/4;
            h=framePosition(4)/4;
            x=framePosition(1)+(framePosition(3)/2)-w/2;
            y=framePosition(2)+(framePosition(4)/2)-h/2;

            app.ImageMira.Position = [x y w h];

            app.ImageMira.Visible="on";
        end
    else
        app.ImageMira.Visible="off";
    end
end

% Close request function: TOPGRAFODECRNEAUIFigure
function TOPGRAFODECRNEAUIFigureCloseRequest(app, event)
    % Se cierra la ventana de analisis y se limpia el archivo txt
    delete(app)
    LimTxt(app);
end

% Value changed function: MinSpinner
function MinSpinnerValueChanged(app, event)
    % Se escala los valores de intensidad de la imagen en el nuevo rango
    global archivo;
    global tipoArchivo;
    global nFrame;

    if tipoArchivo ==".mp4" || tipoArchivo == ".avi"
        video1=archivo;
        framesVideo = video1;
        frame = read(framesVideo,nFrame);
        low = (app.MinSpinner.Value)/255;
        high = (app.MaxSpinner.Value)/255;
        frame = imadjust(frame,[low high]);
        imshow(frame,'Parent',app.UIAxes);

    elseif tipoArchivo == ".jpg" || tipoArchivo == ".jpeg" || tipoArchivo == ".bmp" || tipoArchivo == ".png" || tipoArchivo == ".tif"
        low = (app.MinSpinner.Value)/255;
```

```
            high = (app.MaxSpinner.Value)/255;
            archivo = imadjust(archivo,[low high]);
            imshow(archivo,'Parent',app.UIAxes);
        end


    end

    % Value changed function: MaxSpinner
    function MaxSpinnerValueChanged(app, event)
        % Se escala los valores de intensidad de la imagen en el nuevo rango
        global archivo;
        global tipoArchivo;
        global nFrame;

        if tipoArchivo ==".mp4" || tipoArchivo == ".avi"
            video1=archivo;
            framesVideo = video1;
            frame = read(framesVideo,nFrame);
            low = (app.MinSpinner.Value)/255;
            high = (app.MaxSpinner.Value)/255;
            frame = imadjust(frame,[low high]);
            imshow(frame,'Parent',app.UIAxes);

        elseif tipoArchivo == ".jpg" || tipoArchivo == ".jpeg" || tipoArchivo == ".bmp" || tipoArchivo == ".png" || tipoArchivo == ".tif"
            low = (app.MinSpinner.Value)/255;
            high = (app.MaxSpinner.Value)/255;
            archivo = imadjust(archivo,[low high]);
            imshow(archivo,'Parent',app.UIAxes);
        end
    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Get the file path for locating images
        pathToMLAPP = fileparts(mfilename('fullpath'));

        % Create TOPGRAFODECRNEAUIFigure and hide until all components are created
        app.TOPGRAFODECRNEAUIFigure = uifigure('Visible', 'off');
        app.TOPGRAFODECRNEAUIFigure.Color = [1 1 1];
        app.TOPGRAFODECRNEAUIFigure.Position = [100 100 753 613];
        app.TOPGRAFODECRNEAUIFigure.Name = 'TOPÓGRAFO DE CÓRNEA';
        app.TOPGRAFODECRNEAUIFigure.Resize = 'off';
        app.TOPGRAFODECRNEAUIFigure.CloseRequestFcn = createCallbackFcn(app, @TOPGRAFODECRNEAUIFigureCloseRequest, true);

        % Create UIAxes
        app.UIAxes = uiaxes(app.TOPGRAFODECRNEAUIFigure);
        app.UIAxes.Toolbar.Visible = 'off';
        app.UIAxes.XAxisLocation = 'origin';
        app.UIAxes.XColor = 'none';
        app.UIAxes.XTickLabelRotation = 0;
        app.UIAxes.YAxisLocation = 'origin';
        app.UIAxes.YColor = 'none';
        app.UIAxes.YTickLabelRotation = 0;
        app.UIAxes.ZColor = 'none';
        app.UIAxes.ZTickLabelRotation = 0;
        app.UIAxes.XGrid = 'on';
        app.UIAxes.YGrid = 'on';
        app.UIAxes.GridAlpha = 0.2;
        app.UIAxes.Box = 'on';
        app.UIAxes.Position = [134 94 460 324];

        % Create Panel
        app.Panel = uipanel(app.TOPGRAFODECRNEAUIFigure);
        app.Panel.BackgroundColor = [0 0.4471 0.7412];
        app.Panel.Position = [1 1 134 613];

        % Create Image
        app.Image = uiimage(app.Panel);
        app.Image.Position = [18 488 100 100];
        app.Image.ImageSource = fullfile(pathToMLAPP, 'Recurso 1.png');

        % Create ControlButton
        app.ControlButton = uibutton(app.Panel, 'push');
        app.ControlButton.ButtonPushedFcn = createCallbackFcn(app, @ControlButtonPushed, true);
        app.ControlButton.Position = [32 27 72 22];
        app.ControlButton.Text = 'Control';

        % Create CapturaButton
        app.CapturaButton = uibutton(app.Panel, 'push');
        app.CapturaButton.ButtonPushedFcn = createCallbackFcn(app, @CapturaButtonPushed, true);
        app.CapturaButton.Position = [33 71 71 23];
        app.CapturaButton.Text = 'Captura';

        % Create AnalisisLabel
        app.AnalisisLabel = uilabel(app.TOPGRAFODECRNEAUIFigure);
        app.AnalisisLabel.FontSize = 26;
        app.AnalisisLabel.FontWeight = 'bold';
        app.AnalisisLabel.Position = [350 522 105 34];
        app.AnalisisLabel.Text = 'Analisis';

        % Create AtrasButton
```

```
app.AtrasButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.AtrasButton.ButtonPushedFcn = createCallbackFcn(app, @AtrasButtonPushed, true);
app.AtrasButton.Position = [620 28 100 22];
app.AtrasButton.Text = 'Atras';

% Create Atras1Button
app.Atras1Button = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.Atras1Button.ButtonPushedFcn = createCallbackFcn(app, @Atras1ButtonPushed, true);
app.Atras1Button.Visible = 'off';
app.Atras1Button.Position = [305 72 56 23];
app.Atras1Button.Text = '<';

% Create Adelante1Button
app.Adelante1Button = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.Adelante1Button.ButtonPushedFcn = createCallbackFcn(app, @Adelante1ButtonPushed, true);
app.Adelante1Button.Visible = 'off';
app.Adelante1Button.Position = [383 72 59 23];
app.Adelante1Button.Text = '>';

% Create GuardarframeButton
app.GuardarframeButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.GuardarframeButton.ButtonPushedFcn = createCallbackFcn(app, @GuardarframeButtonPushed, true);
app.GuardarframeButton.Visible = 'off';
app.GuardarframeButton.Position = [623 137 100 23];
app.GuardarframeButton.Text = 'Guardar frame';

% Create Atras2Button
app.Atras2Button = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.Atras2Button.ButtonPushedFcn = createCallbackFcn(app, @Atras2ButtonPushed, true);
app.Atras2Button.Visible = 'off';
app.Atras2Button.Position = [225 72 57 23];
app.Atras2Button.Text = '<<';

% Create Adelante2Button
app.Adelante2Button = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.Adelante2Button.ButtonPushedFcn = createCallbackFcn(app, @Adelante2ButtonPushed, true);
app.Adelante2Button.Visible = 'off';
app.Adelante2Button.Position = [462 72 58 23];
app.Adelante2Button.Text = '>>';

% Create SubirarchivoButton
app.SubirarchivoButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.SubirarchivoButton.ButtonPushedFcn = createCallbackFcn(app, @SubirarchivoButtonPushed, true);
app.SubirarchivoButton.Position = [195 417 100 23];
app.SubirarchivoButton.Text = 'Subir archivo';

% Create MirillaCheckBox
app.MirillaCheckBox = uicheckbox(app.TOPGRAFODECRNEAUIFigure);
app.MirillaCheckBox.ValueChangedFcn = createCallbackFcn(app, @MirillaCheckBoxValueChanged, true);
app.MirillaCheckBox.Visible = 'off';
app.MirillaCheckBox.Text = 'Mirilla';
app.MirillaCheckBox.Position = [624 199 53 22];

% Create ImageMira
app.ImageMira = uiimage(app.TOPGRAFODECRNEAUIFigure);
app.ImageMira.Visible = 'off';
app.ImageMira.Position = [618 364 100 100];
app.ImageMira.ImageSource = fullfile(pathToMLAPP, 'MiraBlanca.png');

% Create AnalizarButton
app.AnalizarButton = uibutton(app.TOPGRAFODECRNEAUIFigure, 'push');
app.AnalizarButton.ButtonPushedFcn = createCallbackFcn(app, @AnalizarButtonPushed, true);
app.AnalizarButton.Visible = 'off';
app.AnalizarButton.Position = [442 417 100 23];
app.AnalizarButton.Text = 'Analizar';

% Create IntensidadPanel
app.IntensidadPanel = uipanel(app.TOPGRAFODECRNEAUIFigure);
app.IntensidadPanel.BorderType = 'none';
app.IntensidadPanel.Title = 'Intensidad';
app.IntensidadPanel.Visible = 'off';
app.IntensidadPanel.BackgroundColor = [1 1 1];
app.IntensidadPanel.FontSize = 16;
app.IntensidadPanel.Position = [618 258 123 99];

% Create MinSpinnerLabel
app.MinSpinnerLabel = uilabel(app.IntensidadPanel);
app.MinSpinnerLabel.HorizontalAlignment = 'right';
app.MinSpinnerLabel.FontSize = 14;
app.MinSpinnerLabel.Position = [3 39 28 22];
app.MinSpinnerLabel.Text = 'Min';

% Create MinSpinner
app.MinSpinner = uispinner(app.IntensidadPanel);
app.MinSpinner.Limits = [0 255];
app.MinSpinner.ValueChangedFcn = createCallbackFcn(app, @MinSpinnerValueChanged, true);
app.MinSpinner.FontSize = 14;
app.MinSpinner.Position = [48 39 76 22];

% Create MaxSpinnerLabel
app.MaxSpinnerLabel = uilabel(app.IntensidadPanel);
app.MaxSpinnerLabel.HorizontalAlignment = 'right';
app.MaxSpinnerLabel.FontSize = 14;
app.MaxSpinnerLabel.Position = [2 1 32 22];
app.MaxSpinnerLabel.Text = 'Max';

% Create MaxSpinner
```

```
            app.MaxSpinner = uispinner(app.IntensidadPanel);
            app.MaxSpinner.Limits = [0 255];
            app.MaxSpinner.ValueChangedFcn = createCallbackFcn(app, @MaxSpinnerValueChanged, true);
            app.MaxSpinner.FontSize = 14;
            app.MaxSpinner.Position = [46 1 77 22];
            app.MaxSpinner.Value = 255;

            % Show the figure after all components are created
            app.TOPGRAFODECRNEAUIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = VenAnalisis

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.TOPGRAFODECRNEAUIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.TOPGRAFODECRNEAUIFigure)
        end
    end
end
```

D. Anexo: Plano óptico del topógrafo

A continuación, se muestra el plano del topógrafo, el cual fue desarrollado por el profesor Yobani Mejia dando inicio a la construcción del topógrafo de córnea. Dicho plano está conformado por dos partes, las cuales cuentan con el elemento denotado por el código G56-263, en este se encuentra la ubicación del divisor de haz, el cual permite la formación de la imagen de la denominada mirilla en frente del objeto a realizar la topografía.

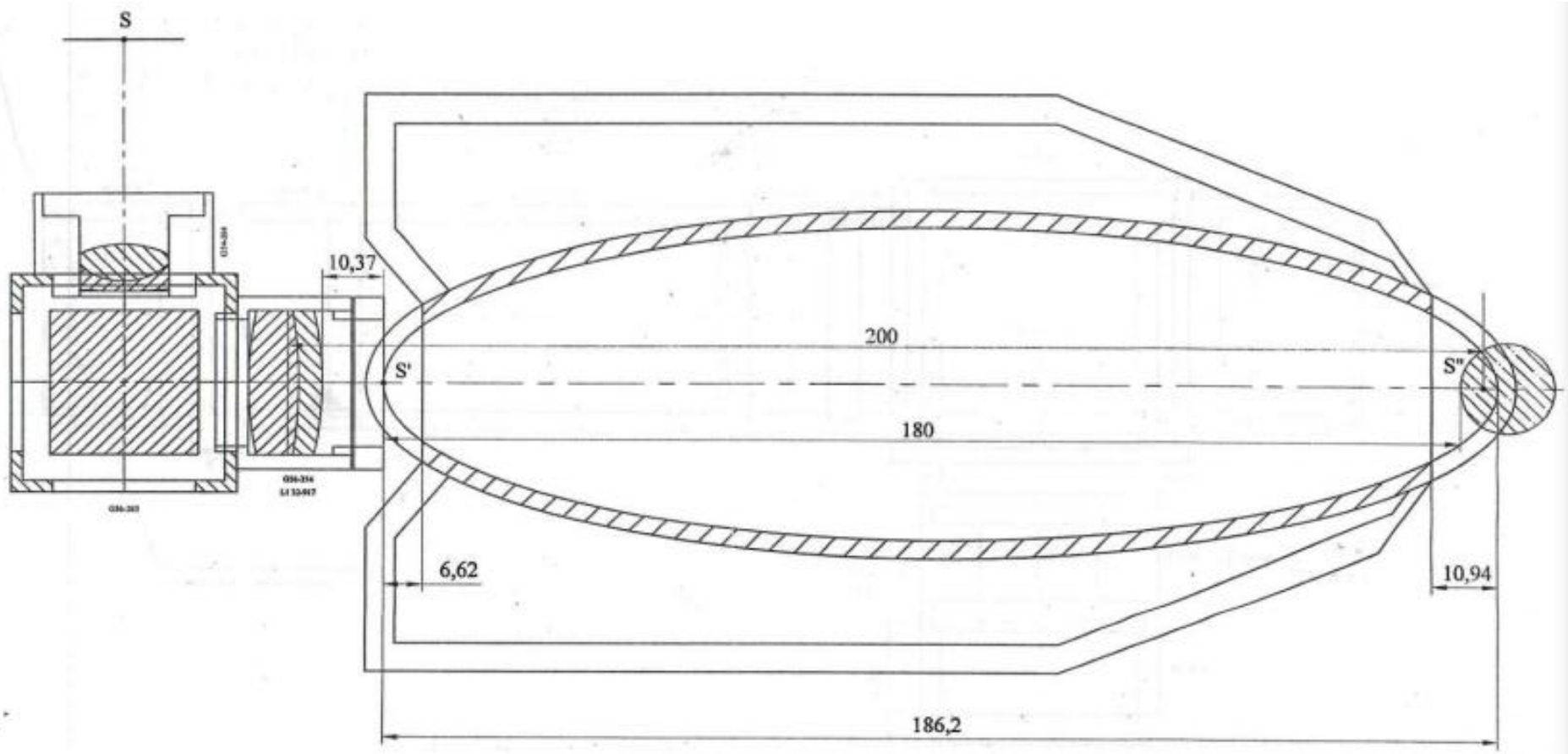
En la primera página del plano se observa el divisor de haz junto con la pantalla de Hartman, en donde se puede evidenciar como dicha pantalla junto a la imagen S' forma una imagen al respaldo de la córnea o lente, la cual se denota con S" y corresponde al arreglo de puntos que será captada por el sensor que se puede apreciar en la segunda página del plano, junto con la lente encargada de formar la imagen para dicho sensor. Cabe aclarar que las distancias que se observan en el plano están dadas en milímetros, y los códigos de lentes se pueden apreciar en la siguiente tabla donde se especifica la distancia focal, diámetro y código de catálogo de cada lente.

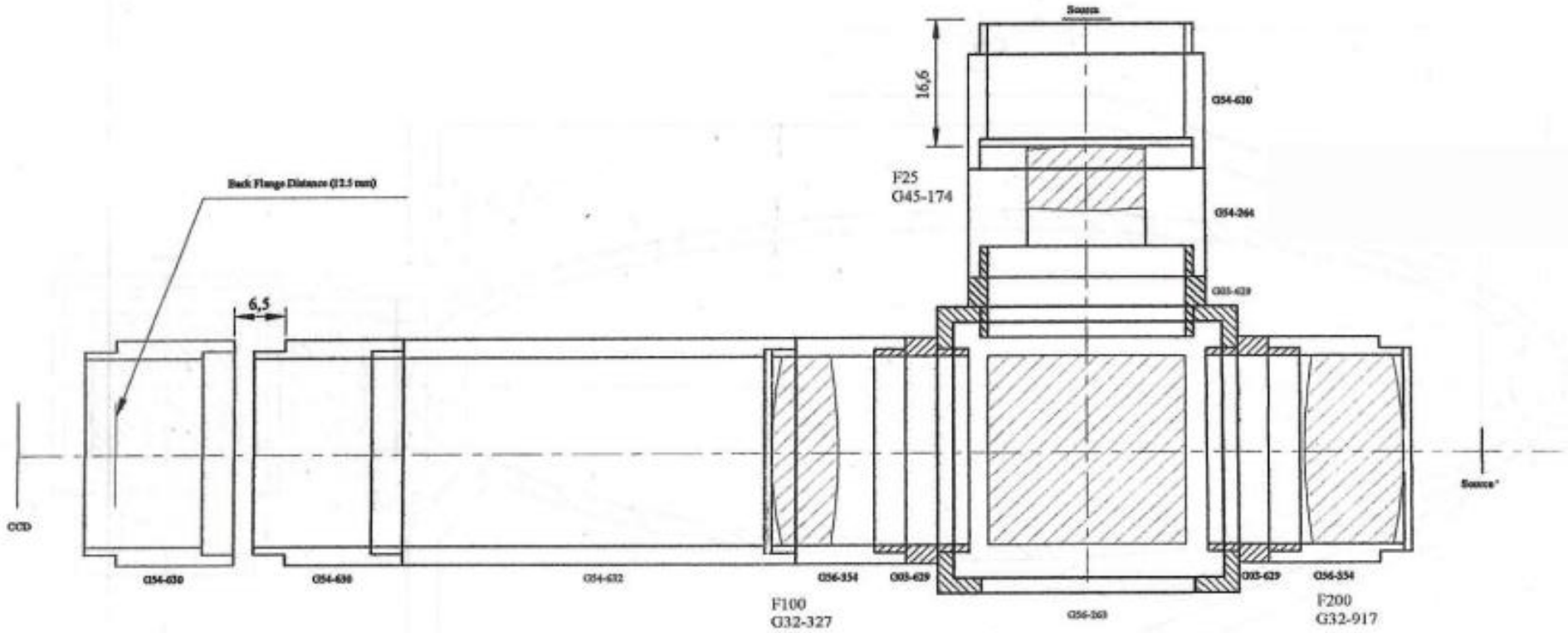
Código	Distancia focal (mm)	Diámetro (mm)	Código de catalogo
G45-265	100	12.5	NT54-623
G45-214	225	25.0	NT56-354
G32-327	100	25.0	NT56-354
G32-917	200	25.0	NT56-354
G45-266	125	18.0	NT54-625
G45-174	25	15.0	NT54-264

Tabla D-1.: Información de lentes utilizados en el topógrafo.

La imagen S o "Source" que se observa en el plano, es formada en una pantalla la cual es un vidrio blanco cubierto de tal manera que se forma la figura de una mirilla rotada, está la podemos observar en la Figura D-1.

Figura D-1.: Pantalla formadora de la imagen S, denotada como “mirilla”.


S
10,37
S'
200
S"
180
6,62
10,94
186,2


Source
16,6
G54-630
F25
G45-174
G54-264
G03-629
Back Flange Distance (12.5 mm)
6,5
CCD
G54-630
G54-630
G54-632
G56-554
G03-629
F100
G32-327
G56-263
G03-629
G56-554
F200
G32-917
Source'

Bibliografía

[Abdulin et al., 2019] Abdulin, E., Friedman, L., and Komogortsev, O. (2019). Custom video-oculography device and its application to fourth purkinje image detection during saccades. *ArXiv*, 1904.07361.

[Aregay et al., 1996] Aregay, J. C., Amigot, B. D., Seijas, J. R. F., Bel, M. F., Valencia, L. G., Poveda, C. H., Contri, C. I., Bas, M. L., Arqués, F. S., Arqués, J. S., Cerspo, M. M. S., and Tenza, M. L. V. (1996). *Tecnología Óptica Lentes Oftálmicas, Diseño y Adaptación*. Edicions UPC.

[Boylestad and Nashelsky, 2009] Boylestad, R. L. and Nashelsky, L. (2009). *Electrónica: teoría de circuitos y dispositivos electrónicos*. Pearson Educación.

[Daniel D. Garcia, 1998] Daniel D. Garcia, Brian A. Barsky, S. A. K. (1998). Cwhatuc : A visual acuity simulator. *Proceedings of SPIE - The International Society for Optical Engineering*, 3246.

[Darryl Meister, 1998] Darryl Meister, A. (1998). Principles of atoric lens design. *SOLA Technical Marketing*, 27(3).

[de la Fuente Arriaga, 2017] de la Fuente Arriaga, J. A. (2017). Topógrafo corneal semiesférico basado en la prueba de hartmann. Master's thesis, Centro de Investigaciones en Óptica, A.C, León, Guanajuato, México.

[en F. Luis Gabriel Valdivieso González, 2014] en F. Luis Gabriel Valdivieso González, M. (2014). *Sistema optimizado para la captura de imágenes de fondo de ojo*. Tesis de doctorado, Instituto Nacional de Astrofísica, Óptica y Electrónica, Tonantzintla, Puebla.

[Evangelos and Konstantinos, 2020] Evangelos, P. and Konstantinos, K. (2020). Corneal placido topography and its correlatio with the best fitting curve for non-astigmatic eyes. *European Journal of Medical and Health Sciences*, 2(1).

[Fernández, 2016] Fernández, J. L. (2016). Reflexión y refracción de la luz.

[Gellrich, 2011] Gellrich, M.-M. (2011). *The Slit Lamp*. Springer.

[González, 2017] González, J. T. B. (2017). *Polinomios de Zernike Para el Estudio de la Aberración de Superficies*. Tesis de licenciatura en matemáticas aplicadas, Benemérita Universidad Autónoma de Puebla, Puebla.

[Guevara, 2018] Guevara, C. O. (2018). Curso de arduino - tutorial 14: Uso del relé ky-019.

[Gwo et al., 2019] Gwo, C.-Y., Zhu, D. C., and Zhang, R. (2019). Brain white matter hyperintensity lesion characterization in t2 fluid-attenuated inversion recovery magnetic resonance images: Shape, texture, and potential growth. *Frontiers in Neuroscience*, 13(1).

[Incorporated, 2004] Incorporated, T. I. (2004). *ULN2803A Darlington Transistor Array*. Texas Instruments Incorporated.

[Jenkins and E.White, 2001] Jenkins, F. A. and E.White, H. (2001). *Fundamentals of Optics*. McGraw-Hill, United States of America.

[Khurana, 2007] Khurana, A. K. (2007). *Comprehensive Ophthalmology*. New Age International (P) Limited.

[Klein and Mandell, 1995] Klein, S. A. and Mandell, R. B. (1995). Shape and refractive powers in corneal topography. *Investigative Ophthalmology & Visual Science*, 36(10).

[Lara, 2021] Lara, J. P. B. (2021). Medición del perfil de la cara convexa de una lente de contacto dura mediante el método de proyección de una línea de luz. Master's thesis, Universidad Nacional de Colombia.

[Martinez, 2019] Martinez, S. R. (2019). Aberraciones oculares inducidas por lentes esclerales en función de su altura sagital en córnea regular. Master's thesis, Universidad Politécnica de Catalunya.

[Marín, 2006] Marín, M. C. P. (2006). *Óptica Fisiológica: El sistema óptico del ojo y la visión binocular*. Universidad Complutense de Madrid, Madrid.

[Matthew, 2017] Matthew (2017). Software publico en github. European Journal of Medical and Health Sciences. https://github.com/mdaddysman/Thorlabs-CMOS-USB-cameras-in-Matlab.

[Mejía, 2011] Mejía, Y. (2011). El frente de onda y su representación con polinomios de zernike. *Ciencia y Tecnología para la Salud Visual y Ocular*, 9(2):145–166.

[Mejía, 2012] Mejía, Y. (2012). La prueba de hartmann en ciencias de la visión. *Ciencia y Tecnología para la Salud Visual y Ocular*, 10(1):149–165.

[Mejía, 2017] Mejía, Y. (2017). Fundamentos de topografía corneal. Departamento de Física, Universidad Nacional de Colombia.

[Mejía, 2021] Mejía, Y. (2021). *Fundamentos de óptica. Curso introductorio*. Facultad de Ciencias, Universidad Nacional de Colombia, Sede Bogotá, Bogotá.

[Mejía et al., 2016] Mejía, Y., Díaz-Uribe, R., Pacheco, A. L., Estrada-Molina, A., and Spors, F. (2016). Measuring conic constant and vertex radius of fast convex conic surfaces from a set of hartmann patterns. *Optics Communications*, 363:166–175.

[Mejía and Galeano, 2009] Mejía, Y. and Galeano, J. C. (2009). Corneal topographer based on the hartmann test. *Optometry and Vision Science*, 86(4):370–381.

[Mejía and Hernández, 2001] Mejía, Y. and Hernández, D. M. (2001). Object surface for applying a modified hartmann test to measure corneal topography. *Applied Optics*, 40(31):5778–5786.

[ming Dai, 2008] ming Dai, G. (2008). *Wavefront Optics for Vision Correction*. SPIE, Washington.

[MINSALUD, 2016] MINSALUD (2016). Análisis de situación de salud visual en colombia.

[Olarte and Mejía, 2006] Olarte, O. E. and Mejía, Y. (2006). A morphological based method to calculate the centroid spots of hartmann patterns. *Optics communications*, 260(1):87–90.

[Olarte, 2011] Olarte, R. V. (2011). Entendiendo e interpretando las aberraciones ópticas. *Ciencia y Tecnología para la Salud Visual y Ocular*, 9(2):105–122.

[OSHA, 2015] OSHA (2015). Boletín para la industria en general. Administración de Seguridad y Salud Ocupacional. Departamento de Trabajo de los EE. UU.

[Piñera, 1998] Piñera, A. G. (1998). *Calidad Óptica del Ojo Humano en Función de la Edad*. Tesis de doctorado, Universidad de Murcia, Murcia.

[Recarte, 2017] Recarte, M. (2017). Polinomios de zernike y su aplicación en oftalmología. *Revista de la Escuela de Física, UNAH*, 5(1):21 – 25.

[Sigut and Sidha, 2011] Sigut, J. and Sidha, S.-A. (2011). Iris center corneal reflection method for gaze tracking using visible light. *Transactions on Biomedical Engineering*, 58(2):411 – 419.

[Singh and Krishna, 2022] Singh, A. and Krishna, V. (2022). Slit lamp examination techniques. *Digital Journal of Ophthalmology*, 32(4):104–108.

[Stewart, 2012] Stewart, J. (2012). *Multivariable calculus*. McMaster University and University of Toronto, United States.

[Suárez, 2012] Suárez, C. A. C. (2012). Tres modelos de explicación de la refracción: Bacon, pecham, witelo. *Anales del Seminario de Historia de la Filosofía*, 29(2):449–480.

[Thorlabs, 2014] Thorlabs, I. (2014). Thorcam^{TM} software for scientific and compact usb cameras. Software oficial desarrollado por la empresa ThorLabs, para el uso y análisis de imágenes obtenidas con cámaras de la marca línea de cámaras ThorCam.

[Tápias et al., 2014] Tápias, M., Torrents, A., Álvarez, J. L., and Pujol, J. (2014). *Óptica Visual. Teoría*. Universitat Politècnica de Catalunya, Barcelona.

[Vieyra et al., 2018] Vieyra, A. G., Solís, E. M., Hidalgo, J. J. O., Vázquez, K. B. V., Uribe, U. C., and Gómez, G. H. (2018). Diseño de un circuito de control de iluminación para un sistema formador de imágenes de purkinje. *Pistas Educativas*, 39(128).

[Warren J, 2000] Warren J, S. (2000). *Modern Optical Engineering*. McGraw-Hill.