Diseño e Implementación del Sistema Inteligente

**HaloLights**

Julián González Dos Reis - bk0136

Tabla de contenido

[Introducción 2](#_Toc535332578)

[Actividades Realizadas 4](#_Toc535332579)

[Pasos a Seguir para un Correcto Funcionamiento 8](#_Toc535332580)

[Conclusiones y Futuras Mejoras 9](#_Toc535332581)

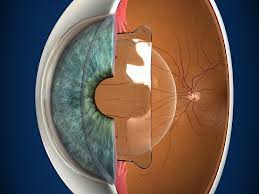
[Bibliografía y Referencias 10](#_Toc535332582)

# Introducción

Hoy en día hay una infinidad de sistemas inteligentes basados en la percepción computacional por todo el mundo, ya sea para el entretenimiento, como pueden ser los bien conocidos filtros de Snapchat, o para la seguridad y ayuda de la sociedad.

*HaloLights* es un sistema capaz de mostrar los halos nocturnos en los focos de luz durante la noche debido a una reciente operación oftalmológica.

La elección de este proyecto nace de intentar explicar y dar un sentido más técnico a la visión temporal que tengo ahora mismo y la dificultad que conlleva expresarlo con palabras. Hace unos meses me operé de miopía a través de unas lentes intraoculares comúnmente conocidas como ICL debido a que tenía un número elevado de dioptrías y la operación con láser no era viable.

El doctor me comentó los posibles efectos secundarios de la operación y como se irían reduciendo estos a lo largo del tiempo (el ojo no es más que otro sistema inteligente que al meterle un “cuerpo extraño” tiene que aprender a convivir con este nuevo objeto).

Los halos nocturnos aparecen en momentos de mucha oscuridad y con focos de luz alrededor de la visión. Cuando esas fuentes de luz inciden de forma oblicua sobre el ojo ocurre este efecto que es debido a un pequeño agujero que tiene las lentes justo en el centro (50 micras) que dispersa la luz por dentro de la cavidad ocular.

El momento donde más halos aparecen es durante la conducción de noche y es ahí donde el sistema analizará cada una de las imágenes o frames de un vídeo donde obteniendo los diferentes focos de luz dibujaré un círculo alrededor de cada uno de ellos con unas características concretas dando una muy cercana aproximación de cómo es la visión con estos halos.

Otro motivo por el que me decidí a realizar este proyecto es porque cuando estos halos empezaron a aparecer en mi día a día intenté consultarlo en internet y ver si a otras personas les pasaba lo mismo. La falta de información era tan grande que pensé que realizar este sistema inteligente también serviría de aportación para el resto de la sociedad. En la imagen inferior, de las cuatro imágenes que aparecen, la de la parte superior izquierda estaría vinculada a una visión correcta y normal y el resto posibles alteraciones debido a los halos. En mi caso lo más exacto sería una combinación entre ambas imágenes inferiores.



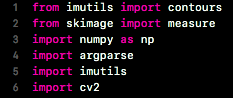
*Posible ejemplo de una visión con halos*

# Actividades Realizadas

Se abordará el proyecto desde el terminal de MAC, que a diferencia de un terminal de Windows se tiene que crear un entorno virtual para poder trabajar cómodamente con Python. La idea principal es trabajar frame a frame con un vídeo determinado que se pasa por parámetro. Se realizan unos toques característicos a la imagen y se van guardando cada una de las posibles luces o focos a tratar.

Los vídeos para tratar el sistema han sido grabados por mí mismo ya que como se comentó antes no hay mucha información referente a los halos nocturnos en internet y encontrar uno bueno ha sido bastante difícil. En los primeros pasos de creación del sistema se trabajaba con imágenes y los resultados eran bastante exitosos, en cambio, en los vídeos debido a cualquier alteración durante la grabación de un foco de luz, un cartel muy luminoso u otros objetos se empezaban a ver los primeros fallos y debilidades del sistema.

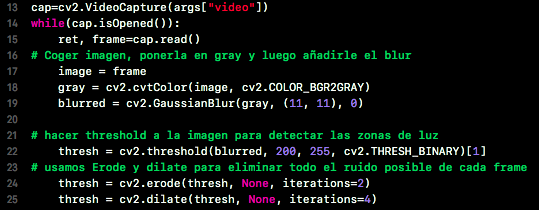
A continuación, se comenta uno a uno los fragmentos más relevantes del código en Python:

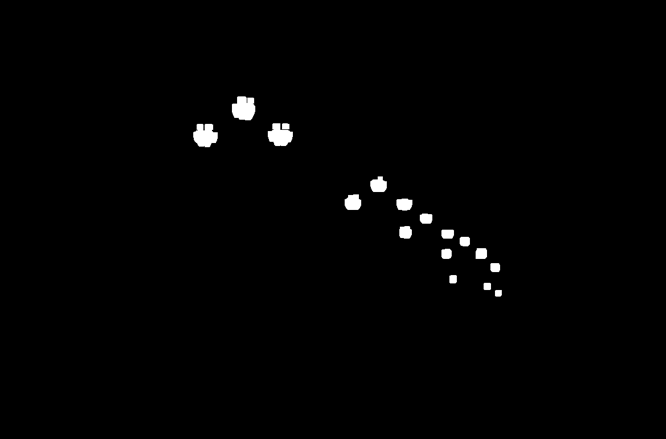


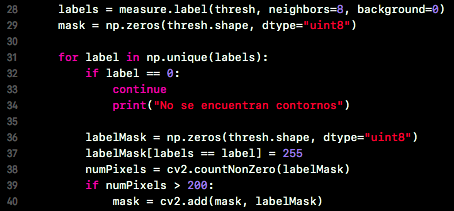
Se utilizan librerías externas como es el caso de *imutils*, que es una serie de funciones de conveniencia para facilitar las operaciones básicas de procesamiento de imágenes, como la traducción, la rotación, el cambio de tamaño, el esqueleto y la visualización de imágenes de Matplotlib con OpenCV y Python y *Scikit-Image* (skimage) que es una colección de algoritmos de procesamiento de imágenes. En las líneas 3-6 se importan otras librerías de uso común en Python.



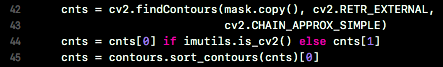
En las líneas 9-11 se hace uso del argparse para poder pasarle el vídeo que nosotros queramos en vez de tener que modificar el código para ir viendo vídeo a vídeo. Escribiendo simplemente --v nombreVideo tras la llamada a Python funcionará correctamente.



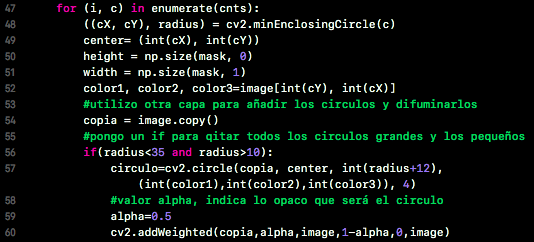
En la línea 13 se coge el vídeo que pasamos por parámetro y lo guardamos en la variable “cap” para en la próxima línea abrir el bucle por toda la duración del vídeo. Lo más interesante de estas líneas es el tratamiento que le hacemos a la imagen: le aplicamos gray, blur y threshold ya que así será mucho más sencillo obtener los focos de luz de cada frame. Además de usar las funciones propias de la biblioteca *imutils* erode y dilate para así conseguir eliminar esos pequeños puntos y ruido innecesario que podría quedar. Se puede observar un posible resultado en la imagen inferior.



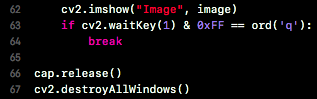
La línea 28 se realiza el análisis real de los componentes conectados utilizando la biblioteca de imágenes *SciKit*. La variable de etiquetas devuelta desde measure.label tiene exactamente las mismas dimensiones que nuestra imagen, la única diferencia es que las etiquetas almacenan un único número entero para cada contorno del thresh. En la línea 29 se inicializa la máscara que guardará cada uno de los contornos. En las líneas 31-40 entramos en el bucle y si el label es 0 lo ignoramos pues es el fondo, si no lo vamos añadiendo si cumple las condiciones.



En la línea 42 se van detectando los contornos en la imagen de la máscara y se ordenan de izquierda a derecha como se muestra en la función vista en clase findContours.



Una vez los contornos están organizados, se accede a ellos uno a uno (línea 47). Para cada uno de estos contornos se utiliza la función de minEnclosingCircle que lo que hace es rodear con el menor radio posible el contorno que se le pasa.



En estas últimas líneas mostramos la imagen una vez hemos realizado todas las operaciones y ponemos la condición de que si se pulsa la letra “q” salimos del sistema.

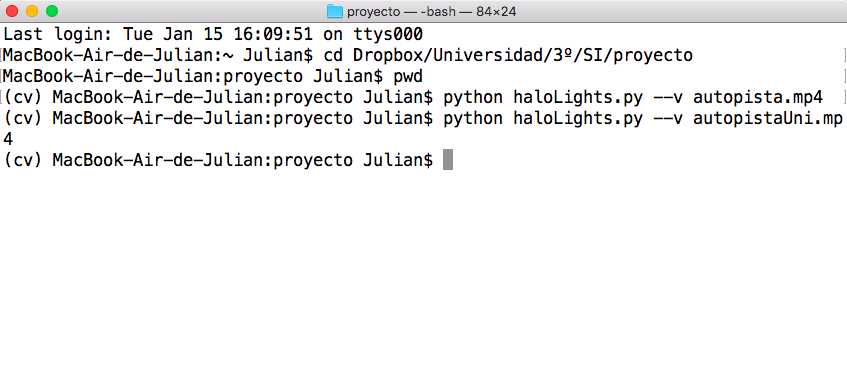
# Pasos a Seguir para un Correcto Funcionamiento

La práctica está compuesta por un conjunto de vídeos y un archivo tipo Python donde se recoge todo el código comentado anteriormente. Para un correcto funcionamiento del sistema es necesario tener instaladas dos bibliotecas externas ya comentadas anteriormente que son *Imutils* y *SciKit-Image*, para su instalación solo es necesario escribir en el terminal los siguientes comandos:

* pip install imutils
* conda install –c conda–forge scikit-image (para Anaconda, Miniconda)
* pip install –U scikit-image (para Linux y OSX)

Una vez instaladas las librerías simplemente es necesario situarse en el directorio donde se encuentren nuestros archivos y lanzar el comando para abrir el archivo de Python. Como le pasamos por parámetro los vídeos habrá que añadir el nombre del vídeo en cuestión.

* python haloLights.py --v nombreVideo



# Conclusiones y Futuras Mejoras

En definitiva, estoy contento con el funcionamiento general del proyecto, desde su etapa inicial de captura de focos de luz en imágenes hasta este último punto de detección mejorada en los videos. Me ha servido para hacer ver a la gente un ápice de lo que se siente y se ve tras una operación oftalmológica, que era mi objetivo principal con este trabajo. También mi conocimiento en la percepción computacional ha mejorado drásticamente, me ha hecho ver el funcionamiento de diversos programas desde un punto de vista más técnico, como puede ser el reconocimiento facial de cualquier sistema de seguridad y su futuro prometedor en nuestra sociedad.

Considero que *HaloLights* se encuentra en una etapa inicial y se le pueden aplicar grandes mejoras para un mejor funcionamiento. Un posible ejemplo es que cuando estoy detectando los contornos crear un filtro para analizar si realmente ese foco de luz debe ser rodeado con un círculo o ignorarlo.

# Bibliografía y Referencias

[1] <https://github.com/jrosebr1/imutils> - Librería *Imutils*

[2] <https://scikit-image.org/> - Librería *SciKit-Image*

[3] <https://docs.opencv.org/3.3.1/d4/d73/tutorial_py_contours_begin.html> – OpenCV funciones

[4] [https://www.vista-laser.com/halos-destellos-deslumbramiento- tras-cirugia-laser-ocular/](https://www.vista-laser.com/halos-destellos-deslumbramiento-%20%20tras-cirugia-laser-ocular/) - Información Halos Nocturnos