TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN BIOTECNOLOGÍA Y BIOINGENIERÍA

CURSO 2023-24

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

DISEÑO Y EVALUACIÓN DE UNA PLATAFORMA AVANZADA PARA LA SIMULACIÓN DE VISIÓN ARTIFICIAL

Autor: Maria del Mar Ayuso Arroyave

Tutor académico: Prof. Dr Eduardo Fernández Jover

D. EDUARDO FERNÁNDEZ JOVER, Doctor y Profesor Titular del área de Biología Molecular del Departamento de Histología y Anatomía de la Universidad Miguel Hernández de Elche.

CERTIFICA

Que el presente trabajo titulado:

"DISEÑO Y EVALUACIÓN DE UNA PLATAFORMA AVANZADA PARA LA SIMULACIÓN DE VISIÓN ARTIFICIAL"

y que constituye la Memoria del Trabajo Fin de Máster en Biotecnología y Bioingeniería, que presenta:

D. MARIA DEL MAR AYUSO ARROYAVE

ha sido realizado bajo su supervisión en el Instituto de Bioingeniería, cumpliendo todos los requisitos necesarios.

Y para que así conste, se expide y firma el presente certificado en Elche a 26 de junio de 2024

Fdo.: Prof. Eduardo Fernández Jover

FERNANDEZ JOVER EDUARDO - 74180965P

Firmado digitalmente por FERNANDEZ JOVER EDUARDO - 74180965P
Fecha: 2024.06.21 07:20:09 +02'00'

Resumen

Las neuroprótesis visuales son dispositivos que se pueden colocar en la retina, en la corteza visual o en el nervio óptico de pacientes con algún tipo de enfermedad visual y tienen como objetivo realizar una estimulación eléctrica en las células que aún son funcionales para así generar una percepción visual o visión artificial en los pacientes con la finalidad de poder hacer en el futuro tareas tales como la navegación, detección y reconocimiento de objetos o lectura.

Para estudiar el funcionamiento de dichas neuroprótesis se plantea en este Trabajo de Fin de Máster crear una plataforma avanzada para desarrollar una simulación a tiempo real de lo que una persona con neuroprótesis visual percibe, lo que se conoce como visión artificial. Esto es útil para los profesionales del ámbito de la salud, ya que facilita el estudio del funcionamiento o las limitaciones de las neuroprótesis.

Esta plataforma permite seleccionar un modelo, que puede ser de retina o de corteza visual, la neuroprótesis que se va a utilizar y también el estímulo que se va a usar de forma sencilla gracias a que es modular; y además también se pueden diseñar desde cero las neuroprótesis, los modelos y los estímulos que va a recibir la neuroprótesis gracias a su flexibilidad.

La finalidad que tiene este trabajo es que los resultados obtenidos sean de ayuda para las futuras investigaciones enfocadas en prótesis visuales.

Palabras Clave

Neuroprótesis Visual, Modelos Visuales, Visión Artificial, Simuladores, Realidad Aumentada

Abstract

Visual neuroprostheses are devices that can be placed either on the retina, in the visual cortex or in the optic nerve of patients with some type of visual impairment. Their aim is to electrically stimulate the cells that are still functional in order to generate visual perception or artificial vision in patients, with the goal of enabling future tasks such as navigation, object detection and recognition, or reading.

To study the functioning of such prostheses, this Master's Thesis proposes to create an advanced plataform to develop a real-time simulation of what a person with a visual prosthesis perceives, known as artificial vision. This is useful for healthcare professionals as it facilitates the study and limitations of the implants.

This platform allows for the selection of a model that can be either retinal or visual cortex, the neuroprostheses to be used, and also the stimulus to be used in a simple way thanks to its modularity. Additionally, implants, models, and stimuli to be received by the implant can also be designed from scratch due to its flexibility.

The purpose of this work is for the results obtained to be helpful for future research focused on visual prostheses.

Keywords

Visual Neuroprosthesis, Visual Models, Artificial Vision, Simulators, Augmented Reality

Índice

1) Introducción y antecedentes

1.1) Neuroprótesis visuales

Actualmente hay millones de personas alrededor del mundo que se ven afectadas por enfermedades que generan baja visión y el número va en aumento debido a, por ejemplo, el aumento de la edad de la población (ref [11]). Algunos ejemplos de dichas enfermedades son la Degeneración Macular debida a la Edad (AMD) o la Retinitis Pigmentosa (RP), los cuales tienen como efecto la degeneración de los fotoreceptores localizados en la retina de forma progresiva y generan pérdida de agudeza visual severa y ceguera (ref [1]).

Debido a esto, investigadores han estudiado diferentes opciones para poder restaurar la visión de las personas afectadas por la baja visión. Algunos ejemplos de esto son los trasplantes de células de retina, la terapia génica o los factores de crecimiento (ref [11]). En este caso particular, este Trabajo de Fin de Máster se va a centrar en la solución consistente en las neuroprótesis visuales (ref [3]), las cuales se encargan de estimular eléctricamente las células sanas de la retina en el caso de las neuroprótesis de retina (ref [1], ref [12]), la corteza visual en el caso de las neuroprótesis de corteza visual (ref [15],)o incluso en el propio nervio óptico (que está compuesto de millones de axones de las células ganglionares) en el caso de las neuroprótesis de nervio óptico (ref [19]), pudiendo así generar visión artificial o percepciones visuales, que no son más que una serie de patrones de 'flashes de luz', lo que se llama fosfeno (ref [5]).

1.1.1) Neuroprótesis de retina

Para entender las neuroprótesis de retina, lo primero que hay que tener claro es cómo funciona la retina (ref [1]): Los fotorreceptores (bastones y conos) son los encargados de recolectar la luz que llega y de transformarla en señales electroquímicas que van a tener la información de la intensidad de luz , luego estas señales pasan a unas células más especializadas que detectan diferentes características dependiendo de la célula (células bipolares, amacrinas, horizontales o de Muller), después estas características se codifican en paralelo en las células ganglionares, que pasan las señales eléctricas al cerebro gracias a los axones de dichas células que conectan al cuerpo de las células ganglionares al nervio óptico. Una vez dicho esto, las neuroprótesis de retina tienen como objetivo estimular las células que aún quedan sanas gracias a los electrodos para así poder generar una respuesta neuronal y, por consiguiente, una percepción o fosfeno.

Las neuroprótesis de retina se pueden clasificar de varias maneras, siendo una de ellas la posición donde se implantan:

- Neuroprótesis epiretinal: La neuroprótesis se coloca en la superficie de la retina.

- Neuroprótesis subretinal: La neuroprótesis se implanta cerca de las células bipolares, donde estarían los fotoreceptores que se han perdido.
- Neuroprótesis supracoroidal: Se coloca la neuroprótesis entre la coroides y la esclera.

Esto se puede visualizar mejor gracias a la siguiente imagen (Figura[1]):

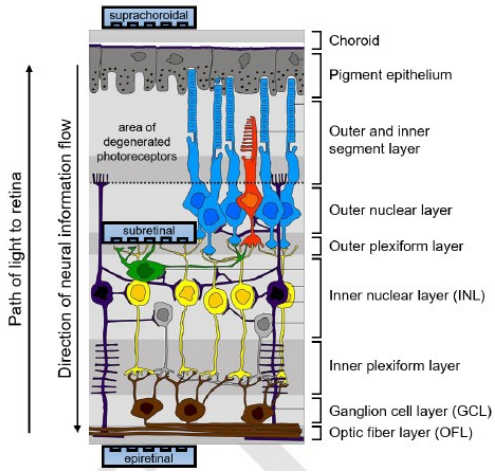

Figura 1: Esquema de la retina y los tipos de neuroprótesis según su posición (ref [1]).

En este momento se va a proceder a exponer una serie de ejemplos de neuroprótesis de cada uno de los tipos mencionados anteriormente:

- Argus II (ref [8]): Consiste en una neuroprótesis epiretinal cuyo fabricante es Second Sight Medical Products Inc (ahora llevado por Cortigent). Éste contiene una antena (bobina receptora) y un array de polímero de 60 electrodos de platino que se colocan en una red de 6x10 y que tienen 200 µm de diámetro y 525 µm de distancia entre electrodos (vertical y horizontal). Esto se puede ver en las siguientes figuras (Figura [2]):

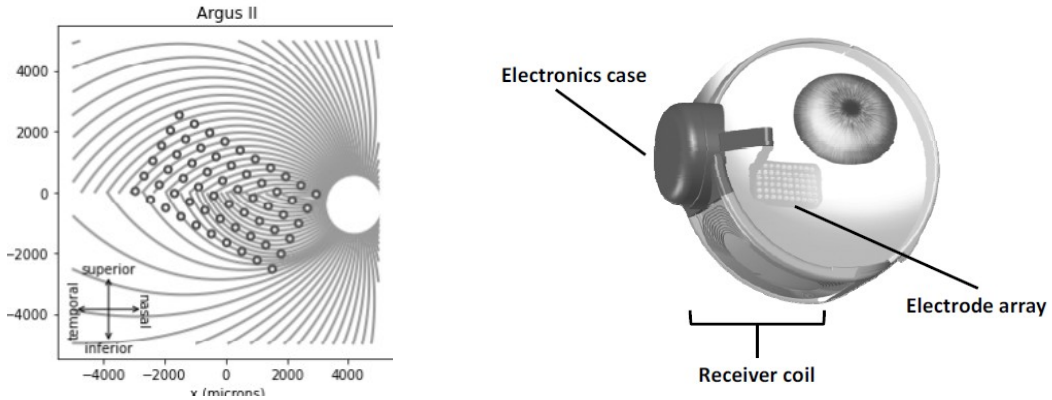

Figura 2: Representación (izda) y componentes (dcha, ref [8]) de la neuroprótesis Argus II.

Existe también su versión anterior: Argus I, el cual consiste en 16 electrodos en un array de 4x4 electrodos, separación entre electrodos de 800 µm y dos distintos diámetros (250 µm y 500 µm) colocados en forma de tablero de ajedrez.

- Alpha AMS (ref [9]): Consiste en una neuroprótesis subretinal fabricada por Retina Implant AG que consta de un chip CMOS de tamaño 4mmx3.2mmx70µm donde se encuentran 1600 “píxeles” de 70µmx70µm y contienen un fotodiodo, un amplificador y un electrodo de estimulación. Éste chip está unido a una película de poliimida distal. Ésto se puede observar en las siguientes figuras (Figura [3]):

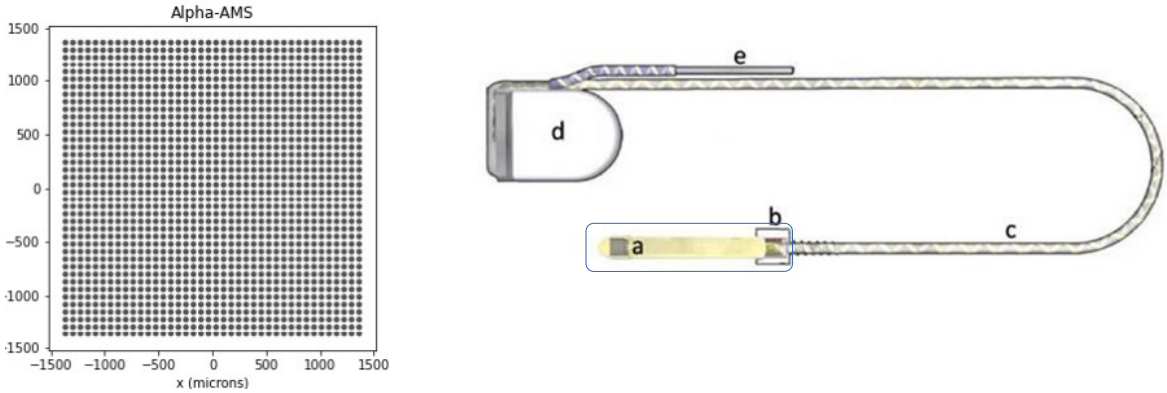

Figura 3: Representación (izda) y componentes (dcha, ref [9]) de la neuroprótesis Alpha-AMS.

- BVT-24 (ref [10]): Consiste en una neuroprótesis supracoroidal fabricada por Bionic Vision Technologies. Éste consta de un sustrato de silicona con 33 electrodos de estimulación (30 de ellos de 600 µm de diámetro y 3 de 400 µm de diámetro) de platino y 2 electrodos de retorno de 2 mm de diámetro. Además, se coloca un electrodo de retorno más detrás de la oreja de forma subcutánea. En este caso hay 24 canales, uno para los 13 electrodos de estimulación exteriores, 20 para cada uno de los demás electrodos de estimulación y 3 para cada uno de los electrodos de retorno. Ésto se puede observar en las siguientes imágenes (figura [4]):

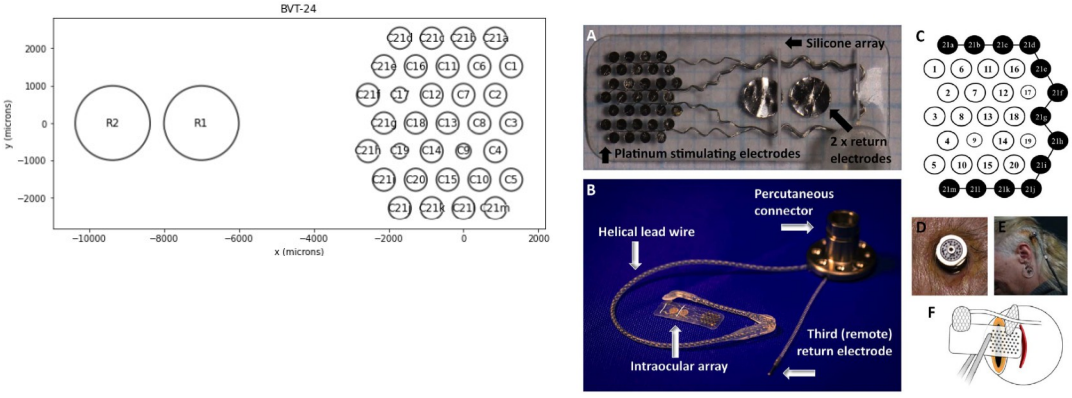

Figura 4: Representación (izda) y colocación (dcha, ref [10]) de la neuroprótesis BVT-24.

Existen otros tipos de neuroprótesis, tales como las neuroprótesis PRIMA y sus respectivas variantes, Alpha IMS que es una variante de Alpha AMS e IMIE.

1.1.2) Neuroprótesis de corteza visual

La corteza visual (ref [18]) es la zona del cerebro que principalmente se encarga de la información que le proporciona la retina, y se encuentra en el lóbulo occipital de la corteza cerebral primaria (la parte posterior del cerebro). La corteza visual se divide en 5 regiones (de V1 a V5). La información visual primero pasa por el tálamo y hace sinapsis en el núcleo geniculado lateral antes de llegar a V1, también conocido como la corteza visual primaria, que se centra alrededor del surco calcarino. Cada hemisferio del cerebro tiene su propia corteza visual que procesa la información del campo visual contralateral, es decir, el área cortical derecha procesa la información del ojo izquierdo y viceversa. El propósito principal de la corteza visual es recibir, segmentar e integrar la información visual, la cual luego es enviada a otras regiones del cerebro para ser analizada y utilizada. Esta especialización permite que otras regiones corticales se dediquen a funciones como la toma de decisiones y el funcionamiento ejecutivo. Sin embargo, dado que el procesamiento visual es mayormente inconsciente, puede llevar a hacer una interpretación errónea de los datos visuales, como se demuestra en la eficacia de las ilusiones visuales.

Una vez dicho esto, las neuroprótesis de corteza visual son aquellas que se encargan de estimular eléctricamente la corteza visual. Cabe destacar que, como se ha visto en la referencia [5], se obtienen muy buenos resultados al colocar una neuroprótesis en la zona de la corteza visual primaria (V1) tanto de forma superficial o intracortical (ver Figura [5]).

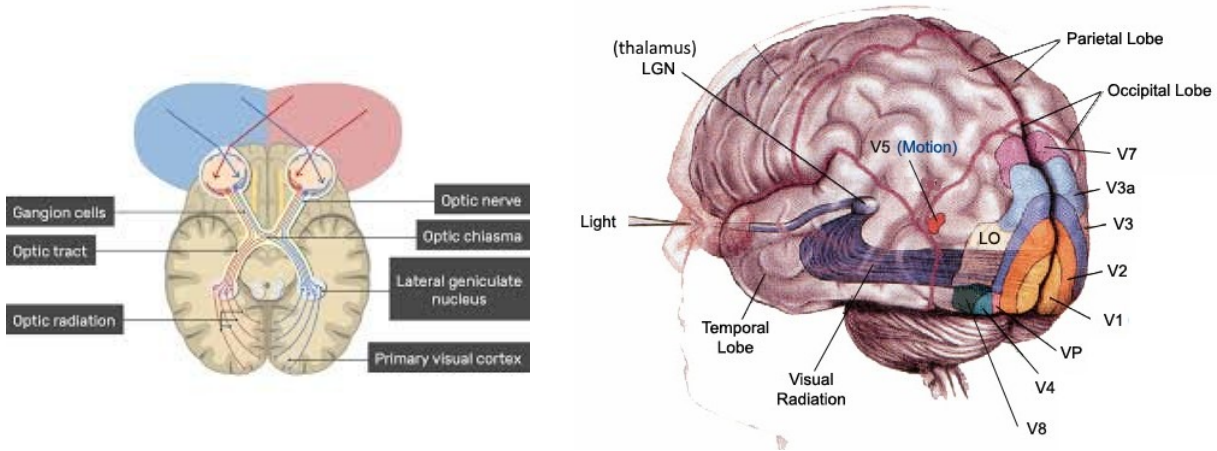

Figura 5: Zonas de la corteza cerebral (ref [15] y ref []).

En este momento se van exponer una serie de ejemplos de neuroprótesis de corteza visual:

- CORTIVIS (ref [2], ref [11]): Esta neuroprótesis, que surgió a raíz una colaboración europea entre 6 universidades, una organización investigadora técnica y una compañía de dispositivos biomédicos, contiene un array de tamaño 10x10 llamado Utah Electrode Array (UEA) que

tiene 96 electrodos, cada uno con 80 µm de diámetro, 400 µm de espaciado entre electrodos y una profundidad de entre 1 y 1.5 mm. Esto se puede ver en la siguiente figura (Figura [6]):

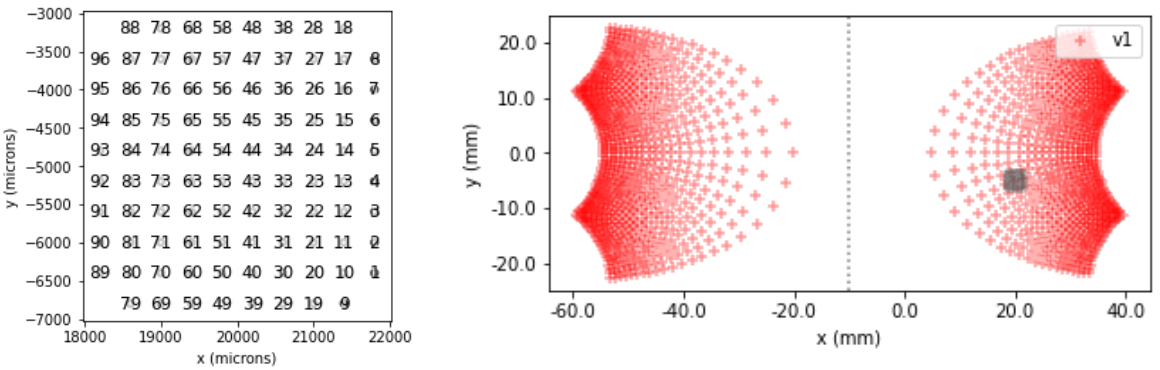

Figura 6: Representación (izda) y colocación en la corteza visual (dcha) de la neuroprótesis CORTIVIS.

- ORION (ref [15]): Este dispositivo, el cual lleva la misma empresa que Argus I y Argus II, consiste en un array de 60 electrodos dispuestos de forma hexagonal y también una bobina la cual también es la misma que los de Argus. Se coloca en el lóbulo occipital medio. Se puede ver en la siguiente figura (Figura 7):

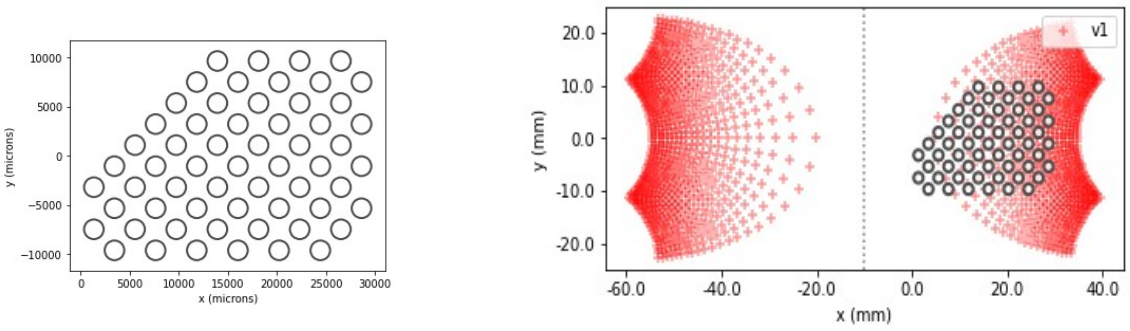

Figura 7: Representación (izda) y colocación en la corteza visual (dcha) de la neuroprótesis ORION.

Además de esta neuroprótesis, existen otras tales como ICVP o Neuralink.

1.1.3) Neuroprótesis de nervio óptico

En el caso de las neuroprótesis de nervio óptico (ref [19-25]), lo que se busca es hacer una estimulación directa gracias a neuroprótesis que contienen electrodos de manguito en espiral o microelectrodos penetrantes. El primer tipo de electrodo mencionado contiene varios contactos colocados en forma de espiral alrededor del nervio óptico, pudiendo hacer así una estimulación selectiva de las fibras nerviosas para así poder generar percepciones de forma más controlada y precisa y además sin que los electrodos penetren en el nervio, mientras que en el segundo caso los

electrodos se insertan directamente. La ventaja de este último es que tiene más resolución espacial. Esto se puede observar mejor en la siguiente imagen (Figura [8]):

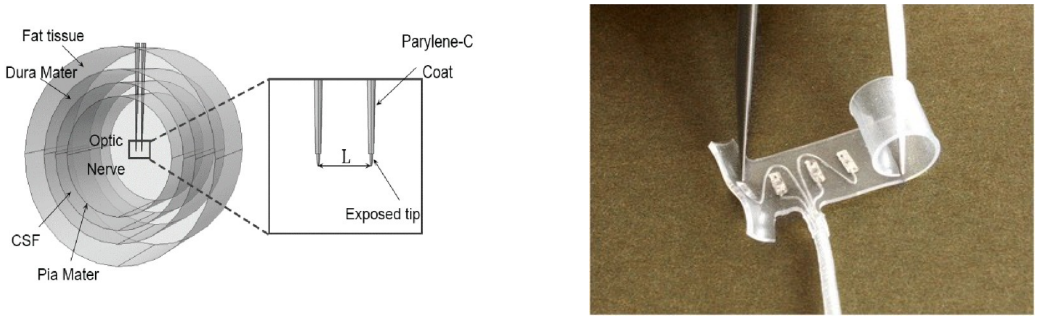

Figura 8: Ejemplos de neuroprótesis de nervio óptico penetrante (izda) y de manguito en espiral (dcha) (ref [25] y Long-term stability of stimulating spiral nerve cuff electrodes on human peripheral nerves")

En este punto se va a proceder a exponer un ejemplo de cada tipo de neuroprótesis de nervio óptico mencionado:

- Neuroprótesis de manguito en espiral MiViP (ref []19): Esta neuroprótesis contiene 4 electrodos que están rodeando el nervio óptico y cada uno tiene una posición angular con respecto al nervio óptico: 0^{o}, 90^{o}, 180^{o} y 270^{o}. Contiene también el neuroestimulador, una antena y un conector. Esto se puede observar mejor en la siguiente figura (Figura 9):

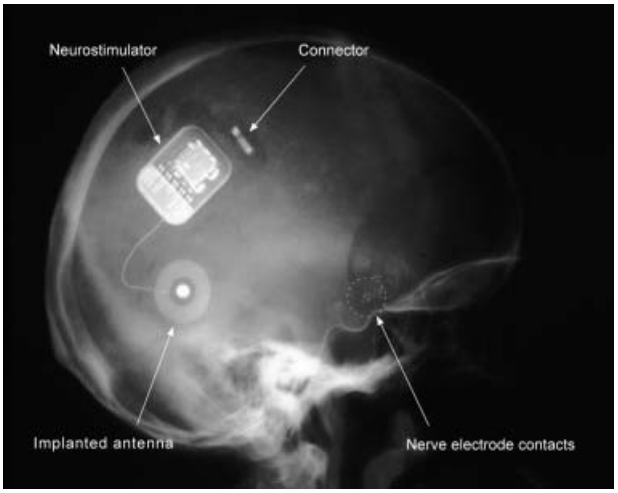

Figura 9: Neuroprótesis MiViP (ref [19])

- Neuroprótesis penetrante (ref [27]): En la referencia 27 se usa una neuroprótesis con 7 electrodos , un electrodo de referencia, una varilla de manipulación y fijación y una placa de silicona cilíndrica. Tienen Todos los electrodos están hechos de alambres de platino-iridio recubiertos de politetrafluoroetileno (PTFE) con un diámetro de 50 µm. La varilla de

manipulación es una barra de platino-iridio con un diámetro de 100 µm, a placa de silicona cilíndrica tiene un diámetro de 2.0 mm, cada electrodo de estimulación tiene 1 mm de longitud expuesta y 3 mm de la varilla de manipulación sobresaliendo a través de la placa de silicona. Esto se puede ver bien en la siguiente figura (Figura 10):

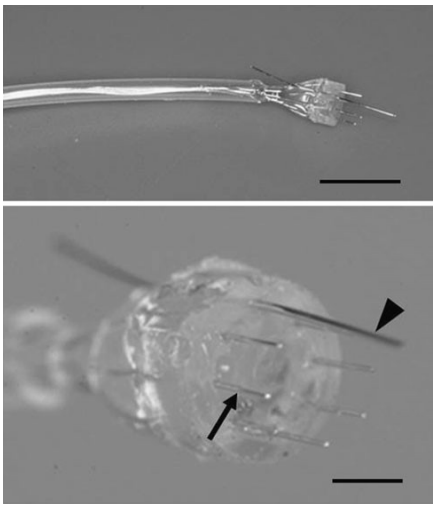

Figura 10: Neuroprótesis de estimulación directa (ref [27])

1.2) Modelos

Hay que tener en cuenta que a la hora de estimular los tejidos con un electrodo de una neuroprótesis cualquiera no se va a obtener un punto perfecto, sino que aparecen distorsiones tanto espaciales como temporales que van a hacer que la forma del fosfeno varíe o que el fosfeno no perdure en el tiempo (en el caso de que el estímulo que se esté mandando sea un tren de pulsos) (ref [1]). Para hacer un buen estudio de estos efectos se han llevado a cabo distintos modelos que tienen en cuenta dichas distorsiones tanto para el caso de las neuroprótesis de retina como el de corteza visual (V1).

1.2.1) Modelo de retina

En este ejemplo se ha desarrollado un modelo llamado el *"modelo cascada"* (ref [1], ref[17]) el cual consiste en una serie de operaciones lineales y no lineales para describir las distorsiones espaciales y lineales debidos al contacto entre la electrónica de la neuroprótesis y el tejido de la retina. Como ejemplo, se tiene el siguiente esquema (Figura [11]):

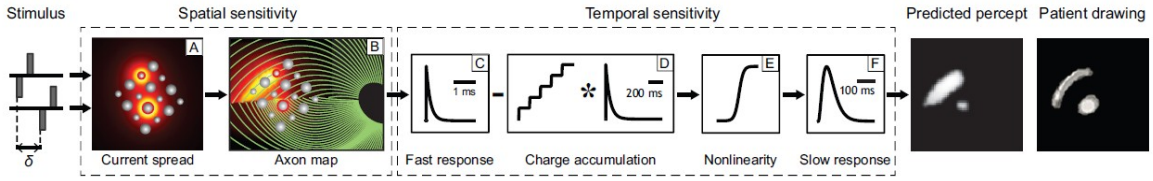

Figura 11: Modelo de Cascada usado para la estimulación del neuroprótesis Argus I (de ref[1]).

La figura describe cómo se obtiene la percepción con el modelo cascada dado la neuroprótesis Argus I. Se puede observar que el primer paso es la entrada de estímulos, que pueden ser pulsos, trenes de pulsos o imágenes (en este caso se puede observar que son dos trenes pulsos bifásicos y desfasados, cada uno yendo a un electrodo). Después de esto se puede ver que hay dos grandes bloques, uno que modela las distorsiones espaciales y otro que desarrolla las distorsiones temporales. En este momento se van a desarrollar ambas partes:

- Bloque de distorsiones espaciales, que contiene 2 bloques:
  - Bloque A: Define cómo la densidad de corriente generada por la estimulación eléctrica se va disipando conforme se va alejando de los bordes del electrodo (se conoce como modelo Scoreboard). La densidad de corriente viene dada por la siguiente ecuación:

$$c(d)=\frac{\alpha}{\alpha+d^n}$$

    donde d es la distancia 3D euclídea al borde del electrodo, α es 14000 y n es 1.69.
  - Bloque B: En el caso de las neuroprótesis epiretinales, al estar cerca del nervio óptico pueden llegar a estimular los axones de las células ganglionares aparte del propio cuerpo de las células, generando dichos axones también percepciones haciendo que se genere la forma de cometa.
- Bloque de distorsiones temporales, que contiene 4 bloques, 3 de ellos lineales (bloques C, D y F, que se modelan como filtros de paso baja y usan funciones gamma de orden n: $\delta(t,n,\tau)=\frac{\exp(-t/\tau)}{\tau(n-1)!}\left(\frac{t}{\tau}\right)^{n-1}$, donde t es tiempo, n es el número de estados de cascada idénticos y τ es la constante de tiempo del filtro) y una no linealidad (bloque E):
  - Bloque C: Describe la respuesta rápida de las células ganglionares y se modela convolucionando la función $f(t)$, que representa la serie temporal de las intensidades de activación de los tejidos en una determinada localización, con una función gamma con $n=1$ y $\tau_1=0.42$ ms, teniendo la siguiente respuesta al impulso:

$$r_1(t)=f(t)*\delta(t,1,\tau_1)$$

- Bloque D: Describe la sensibilidad del tejido a la carga acumulada, disminuyendo la sensibilidad en función de la carga acumulada. Esto se modela haciendo un cálculo de la carga acumulada en cada instante de tiempo ($c(t)$), convolucionando esto con un filtro gamma de $n=1$ y $\tau_2=45.3$ ms, luego el resultado de la convolución se escala con un valor $\varepsilon_1=8.3$ y al final a todo esto se le sustrae a $r_1(t)$, obteniendo el siguiente resultado:

$$r_2(t)=r_1(t)-\varepsilon_1\left(c(t)*\delta(t,1,\tau_2)\right)$$

- Bloque E: Este es un bloque no lineal que modela la relación de entrada salida que hay en la generación de spikes en las células ganglionares, obteniendo lo siguiente a la hora de hacer la convolución de la función no lineal con el resultado del bloque anterior:

$$r_3(t)=r_2(t)\frac{\alpha}{1+\exp\dfrac{\delta-max_t\, r_2(t)}{s}}$$

donde $\alpha=14$ (que es la asíntota de la función no lineal), $s=3$ (que es la pendiente de la función no lineal), y $\delta=16$ (desplazamiento de la función).

- Bloque F: Este último bloque modela la respuesta lenta a la hora de obtener la percepción. Esto se hace convolucionando $r_3(t)$ con un filtro paso baja de $n=3$ y $\tau_3=26.3$ ms, obteniendo la siguiente respuesta:

$$r_4(t)=\varepsilon_2 r_3(t)*\delta(t,3,\tau_3)$$

donde $\varepsilon_2=1000$, siendo un factor de escala para obtener una salida en valores de brillo que va a estar entre 0 y 100.

1.2.2) Modelo de corteza visual

En el caso de la corteza visual se han desarrollado modelos bastante simplificados en los cuales todos los fosfenos tienen el mismo tamaño y están distribuidos de manera uniforme en el campo visual (modelo de Scoreboard), pero en este trabajo se va a estudiar un modelo para obtener fosfenos a partir de la estimulación que recibe la corteza visual. En el caso de este trabajo se va a tratar el modelo Dynaphos (ref [5]).

Este modelo tiene como entrada los parámetros de los estímulos (amplitud (I_{stim}), frecuencia (f), y ancho del pulso (Pw)) de cada electrodo y además la situación de los electrodos en mapa visuotópico de la corteza visual, que no es más que la representación de la relación que hay entre las coordenadas en el campo visual y en la corteza visual primaria (V1). En el caso del mapa visuotópico se nos pueden dar directamente las coordenadas polares de los electrodos en el campo visual o se nos pueden dar las coordenadas de los electrodos en la corteza visual, que en ese caso habría que

obtener su equivalente en el campo visual siguiendo un modelo de dipolo inverso que tiene la siguiente ecuación:

$$w = k\left(\log\left(r^{i\alpha\theta} + a\right) - \log\left(r^{i\alpha\theta} + b\right)\right)$$

donde w es la localización del electrodo en la corteza visual, $r^{i\theta} = z$ es la coordenada polar del electrodo en el campo visual, r es el distancia desde el punto de fijación (generalmente el centro de la mirada) hasta un punto específico en el campo visual, $-\frac{\pi}{2} \leqslant \theta \leqslant \frac{\pi}{2}$ la posición angular de un objeto en el campo visual respecto a un punto de referencia, típicamente el centro de la mirada, k es un factor de escala que ajusta el mapeo a proporciones realistas en distancia de la corteza visual (se da en mm), a, b controlan las singularidades del modelo de dipolo y α controla la distorsión del mapeo. Para obtener las coordenadas en el campo visual dadas las coordenadas en la corteza visual se hace la operación inversa:

$$z = \Lambda^{-1}\left(\frac{ab\left(e^{\frac{w}{k}} - 1\right)}{b - ae^{\frac{w}{k}}}\right)$$

donde $\Lambda^{-1}\left(re^{i\theta}\right) = re^{i\frac{\theta}{\alpha}}$ es el operador de cizallamiento inverso. Además de esto, se puede obtener la magnificación de la corteza, es decir, la cantidad de tejido que se usa en el proceso:

$$M = \frac{k(b-a)}{(r+a)(r+b)}$$

Una vez se tienen los parámetros de entrada, se pueden obtener las siguientes características de los fosfenos:

- Tamaño del fosfeno:

  El tamaño del fosfeno (en grados) viene dado por la siguiente ecuación

  $$P = \frac{D}{M}$$

  donde M es la magnificación de la corteza, $D = 2\sqrt{\frac{I_{stim}}{K}}$ es el diámetro del tejido de la corteza visual activa (en mm), I_{stim} es la corriente de estimulación (en μA) y K es la constante de excitabilidad (en $\mu A / mm^2$).

- Brillo del fosfeno:

  El primer paso es obtener la corriente efectiva de la estimulación, que se hace de la siguiente manera:

  $$I_{eff} = max\left(0, \left(I_{stim} - I_0 - Q\right) Pw \cdot f\right)$$

donde I_{stim} el la amplitud de la corriente, I_0 es la corriente de fuga, Q es la alteración estructural de las células cerebrales después del aprendizaje (registro de memoria), f y Pw son la frecuencia y el ancho del pulso, formando ambos parámetros el ciclo de trabajo.

Una vez se ha calculado la corriente efectiva, hay que calcular la activación de la corteza visual en cada instante de tiempo t de la siguiente manera:

$$A_t = A_{t-\Delta t} + \Delta A$$

donde Δt es la duración de un frame, $\Delta A = \left(\frac{-A_{t-\Delta t}}{\tau_{act}} + I_{eff} d \right)$, τ_{act} es la constante de decrecimiento de la activación de la corteza (en s) y d es la escala de la duración de la estimulación relativa a la duración del frame (está entre 0 y 1).

Ya con la activación de la corteza visual calculada, ya solo queda calcular el brillo:

$$B = \frac{1}{1 + e^{-\lambda (A - A_{50})}}$$

donde se sigue una función sigmoidal con λ la pendiente de la curva sigmoide y A_{50} el valor de la activación del tejido de la corteza en el que el fosfeno tiene el 50% del máximo del brillo.

En el caso de este modelo solo va a haber fosfeno cuando se supere el umbral de activación de la corteza, cuyo valor se obtiene gracias a la distribución normal:

$$A_{th} = N\left(Th_{50}, \sigma^2 \right)$$

donde Th_{50} el un umbral de probabilidad del 50% y σ es la desviación estándar.

- Dinámica temporal del fosfeno:

  En este caso se quiere ver el efecto de la acomodación (adaptación del ojo a la luz continua) del brillo en tiempos prolongados o de forma repetida. Para eso se cuenta con el registro de memoria, que se puede calcular de la siguiente manera (para cada frame):

  $$Q_t = Q_{t-\Delta t} - \Delta Q$$

  donde $\Delta Q = \left(\frac{-Q_{t-\Delta t}}{\tau_{trace}} + I_{eff} \kappa \right) \Delta t$, τ_{trace} es una constante de tiempo del decrecimiento del registro (en s) y κ controla la entrada.

Una vez obtenidos estos parámetros para cada fosfeno se hace un renderizado en el campo visual y para finalizar se suma cada renderizado para así obtener la visión artificial completa. Todo lo que se acaba de explicar se puede observar mejor en la siguiente figura (figura [12]):

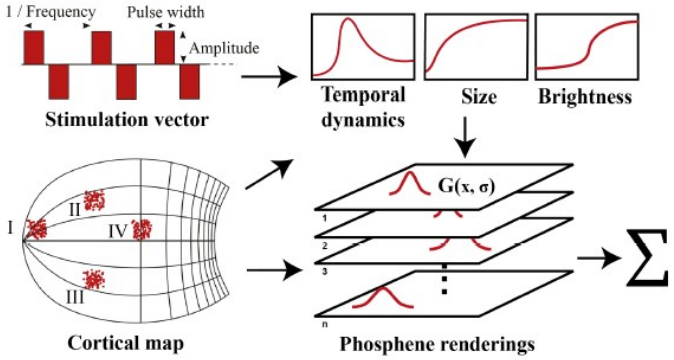

Figura 12: Modelo de Dynaphos (de ref[5]).

Una vez dicho esto, se va a proceder a proporcionar los valores usados para algunos de los parámetros mencionados anteriormente:

Parámetro	**Valor**
a	0.75
k	17.3
b	120
α	0.95
K	$675\,\mu A/mm$
I_0	$23.9\,\mu A$
d	1
τ_{act}	$0.111\,s$
λ	$19.2 \cdot 10^{7}$
A_{50}	$1.06 \cdot 10^{-6}$
Th_{50}	$9.14 \cdot 10^{-8}$
σ	$6.72 \cdot 10^{-8}$
τ_{trace}	$1.97 \cdot 10^{3}\,s$
κ	14

Tabla 1: Valores de algunos parámetros del modelo Dynaphos (de ref[5])

1.3) Simuladores

En el contexto de poder hacer un estudio de cómo va a ser el comportamiento de las neuroprótesis a la hora de usar diferentes tipos de estímulos, de estimular distintos electrodos y poder predecir cuál va a ser su correspondiente percepción de forma aproximada antes incluso de desarrollar dicha neuroprótesis se han descrito distintos simuladores con diferentes características. En este caso se van a destacar 3 de ellos: Pulse2Percept (ref[1] y ref[13]), Dynaphos (ref[5]) y OpticStim (ref [22+]) . Estos paquetes se desarrollaron para el entorno de programación Python y tienen capacidades avanzadas para la simulación de fosfenos. En este punto se va a hablar de cada uno de los paquetes y después se va a hacer una comparación entre ellos.

1.3.1) Pulse2Percept

El paquete de Python Pulse2Percept es un marco de simulación de código abierto diseñado para modelar la experiencia perceptual de los pacientes con neuroprótesis de retina en una amplia gama de configuraciones de neuroprótesis de retina, aunque actualmente se ha ampliado esto a neuroprótesis de corteza visual. Algunas de sus características son las siguientes:

- Este paquete contiene modelos computacionales avanzados para así poder predecir cuál va a ser la percepción que se va a producir, dados una neuroprótesis y los estímulos a utilizar. Gracias a esto los investigadores y desarrolladores pueden optimizar el diseño de las neuroprótesis.
- Pulse2Percept está dividido en distintos módulos (estímulos, modelos, neuroprótesis…), y gracias a esto se pueden usar distintas neuroprótesis (de corteza y de retina), estímulos (ya sean pulsos, trenes de pulsos, imágenes o vídeos) y modelos (ya sean de retina (modelo de Axones o de Scoreboard) o de corteza visual (modelos Scoreboard o Dynaphos)) y además se pueden diseñar nuevos, pudiendo hacer distintas adaptaciones a distintas necesidades.
- Es de fácil uso, por lo que es accesible tanto para investigadores con experiencia en programación como para médicos; además es flexible para poder hacer distintas configuraciones.

En resumen, Pulse2Percept es una herramienta poderosa y versátil para simular las percepciones de los pacientes con neuroprótesis tanto de retina como de corteza visual, y su disponibilidad como paquete de Python de código abierto lo hace accesible para la comunidad científica y médica interesada en el desarrollo de tecnologías de neuroprótesis visuales.

Habiendo hablado de las características positivas de Pulse2Percept, se van a enumerar también sus puntos negativos:

- Algunas personas con poca experiencia programando o en el campo de las neuroprótesis visuales se puede enfrentar a una curva de aprendizaje a la hora de usar el paquete, pudiendo requerir bastante tiempo a comprender todas las funcionalidades y opciones.
- Dado que Pulse2Percept utiliza un modelo computacional detallado para simular la experiencia perceptual de los pacientes con neuroprótesis visual, la complejidad del modelado y la interpretación de los resultados podrían representar un desafío para usuarios menos familiarizados con conceptos de neurociencia computacional.
- Aunque este paquete es bastante preciso en cuanto a simulación de experiencia perceptual, hay que tener en cuenta que cualquier modelo computacional tiene limitaciones en su capacidad para representar completamente la complejidad y variabilidad de la percepción visual humana. Por lo tanto, los resultados de las simulaciones realizadas con Pulse2Percept deben interpretarse con precaución.
- Dependiendo de la complejidad de las simulaciones y del tamaño de los datos a utilizar, Pulse2Percept podría requerir recursos computacionales importantes como capacidad de procesamiento y memoria, lo que podría limitar su uso en sistemas con recursos limitados.

Es importante tener en cuenta que estos posibles inconvenientes son hipotéticos y pueden variar según las necesidades y habilidades de los usuarios. En general, Pulse2Percept sigue siendo una herramienta valiosa para la simulación de la experiencia perceptual de los pacientes con neuroprótesis visuales, con sus ventajas superando posibles inconvenientes.

1.3.2) Dynaphos

El paquete de Python Dynaphos es una herramienta que se utiliza para simular la percepción de fosfenos cuando se tienen neuroprótesis de corteza visual en tiempo real. Este paquete proporciona un entorno de simulación que integra modelos de corteza visual basados en principios neurofisiológicos y empíricos para generar representaciones precisas de la percepción visual inducida por la estimulación eléctrica en la corteza visual. Algunas características a destacar son las siguientes:

- El modelo Dynaphos considera todo lo que se dijo en el apartado 1.2.2, como puede ser el mapeo visuotópico, la magnificación cortical, la propagación de la corriente por el tejido de la corteza visual o las dinámicas temporales, haciendo la simulación de fosfenos bastante realista.
- Gracias a este paquete se puede hacer una simulación del tamaño, brillo y ubicación de los fosfenos que se generan por una estimulación que se hace en la corteza visual, permitiéndonos comprender y optimizar las estrategias de codificación visual en neuroprótesis de corteza visual.

- En este paquete de Python se usa la biblioteca de deep learning PyTorch, que nos permite hacer operaciones diferenciables para poder hacer la optimización basada en gradientes de los modelos de codificación visual.
- Esta herramienta va a permitir a los investigadores, clínicos o desarrolladores de neuroprótesis de corteza visual evaluar y optimizar distintas estrategias de estimulación para mejorar la percepción visual en personas con discapacidades visuales.

En resumen, Dynaphos es un paquete de Python especializado en la simulación de fosfenos en neuroprótesis de corteza visual, que integra modelos de corteza visual y ofrece una plataforma flexible y poderosa para investigar y desarrollar tecnologías de asistencia visual.

Aunque el paquete Dynaphos ofrece numerosas ventaja, también puede presentar algunos inconvenientes potenciales que vale la pena considerar:

- Debido a que el modelo Dynaphos y la generación de fosfenos son bastante complejos, se puede necesitar muchos recursos computacionales tales como memoria y capacidad de procesamiento, limitando la ejecución en ciertos sistemas.
- Como en el paquete Pulse2Percept, para ciertos usuarios sin experiencia en programación esta herramienta puede ser complicada de usar ya que integra modelos neurofisiológicos y de deep learning.
- La capacidad de optimización basada en gradientes es una ventaja clave de Dynaphos, pero también puede requerir un proceso de ajuste y optimización de parámetros bastante detallado para lograr unos buenos resultados. Esto puede llevar a esfuerzo adicional por parte de los usuarios para calibrar adecuadamente los modelos y los parámetros de estimulación.
- Como ocurre también en el paquete Pulse2Percept, Dynaphos aún puede tener limitaciones en la representación precisa de la percepción visual inducida por la estimulación eléctrica. La complejidad del sistema visual humano puede implicar que algunos aspectos de la simulación sean simplificados o no sean completamente representativos de la realidad.

En resumen, Dynaphos ofrece una plataforma poderosa y flexible para la simulación de fosfenos en neuroprótesis de corteza visual, pero es importante tener en cuenta estos posibles inconvenientes relacionados con los requisitos de hardware, la curva de aprendizaje, la optimización de parámetros y las limitaciones en la representación biológica al utilizar esta herramienta en investigaciones y desarrollos en neuro neuroprótesis de corteza visual.

1.3.3) OpticStim

Este último paquete (ref [22]) está basado en Machine Learning y se usa para llevar a cabo experimentos psicofísicos y simular el proceso de optimización de la estimulación del nervio óptico. Este simulador funciona mediante la implementación de un modelo computacional del sistema visual primate que simula la respuesta neuronal a la estimulación del nervio óptico, y además también hace uso de un algoritmo de optimización basado en aprendizaje automático para evolucionar patrones de estimulación óptima del nervio óptico, buscando encontrar configuraciones de estimulación que generen activaciones corticales que correspondan a estímulos visuales naturales específicos. Más concretamente. Algunas de sus ventajas son las siguientes:

- Permite a los investigadores llevar a cabo experimentos simulados para probar y comparar diferentes estrategias de estimulación del nervio óptico en un entorno controlado y reproducible.
- Utiliza un algoritmo de optimización basado en aprendizaje automático para desarrollar patrones de estimulación personalizados que se puedan ajustar a las respuestas de la corteza visual esperadas.
- Incorpora un modelo computacional detallado del sistema visual primate, lo que permite simular de manera realista la respuesta neuronal a la estimulación del nervio óptico y comprender mejor los mecanismos que contiene.

Una vez se han visto los aspectos positivos de esta herramienta, se van a enumerar los negativos:

- la ejecución de este paquete de Python puede requerir recursos computacionales significativos y tiempo de procesamiento.
- A pesar de ser detallado, el modelo del sistema visual utilizado en el simulador puede tener suposiciones que pueden llegar a limitar su capacidad para capturar completamente la complejidad del sistema visual real.
- En este caso no se pueden definir las distintas neuroprótesis a utilizar, y solo está limitado a estimulación del nervio óptico.

1.3.4) Comparación

Una vez habiendo puesto en valor las características y desventajas de ambos paquetes de Python, se va a proceder a hacer una comparación para así justificar la opción que se ha escogido para realizar las simulaciones:

- En el caso de Pulse2Percept se pueden hacer simulaciones de neuroprótesis y modelos de retina y de corteza visual, mientras que en el caso del paquete Dynaphos se centra en la corteza visual y con OpticStim solo se puede hacer estimulación del nervio óptico.

- Tanto Pulse2Percept como Dynaphos son paquetes modulares, pero Pulse2Percept es más flexible ya que permite la rápida integración y prototipado con otros sistemas mientras que Dynaphos tiene un enfoque específico en la precisión del modelo visuotópico y la eficiencia computacional. En cambio, OpticStim no es un paquete modular, lo que dificulta su uso.
- Dynaphos y OpticStim pueden requerir muchos recursos a nivel computacional ya que todo el proceso que se lleva a cabo es complejo, limitando su ejecución en sistemas que no tengan los recursos necesarios, mientras que en el caso de Pulse2Percep es un paquete mucho más accesible.
- En el caso de Dynaphos se requiere cierto nivel de familiaridad con conceptos relacionados con la neurociencia computacional y con Python, lo que podría resultar complejo para los usuarios que no tengan conocimientos previos, mientras que en el caso de Pulse2Percept se puede encontrar toda la documentación detallada de cada una de las partes del código, lo cual facilita la tarea de desarrollo al usuario.
- Tanto Pulse2Percept como Dynaphos requieren un ajuste de parámetros, pero en el caso de Pulse2Percept, al no basarse en gradientes como en el caso de Dynaphos, es un ajuste más sencillo. En el caso de OpticStim solo hay que hacer un entrenamiento con imágenes.

Dicho todo esto, debido a que abarca más modelos y neuroprótesis, requiere menos recursos a nivel computacional, es más fácil de usar y el ajuste de parámetros es más sencillo, el paquete elegido para este Trabajo de Fin de Máster es Pulse2Percept.

2) Objetivos

En el siguiente punto se van a exponer los objetivos de este Trabajo de Fin de Máster, los cuales son los siguientes:

1) Adaptar el paquete de Python Pulse2Percept para su utilización con diferentes modelos de visión artificial.
2) Desarrollar una plataforma en lenguaje Python para generar visión artificial en tiempo real.
3) Evaluar las capacidades y velocidad del procesamiento de la nueva plataforma en entornos reales.
4) Analizar la integración de la plataforma desarrollada en diferentes dispositivos de realidad virtual y realidad aumentada.

3) Procedimiento experimental

3.1) Materiales

En esta memoria, el material a destacar serían las gafas de Realidad Virtual PICO 4 Enterprise, las cuales tienen las siguientes características:

- Tienen 6 grados de libertad (6DoF) gracias al uso de varias cámaras ubicadas en torno a las gafas para ofrecer seguimiento 6DoF inside-out, lo que te permite moverte de forma libre y orgánica en 360°.
- Se puede hacer seguimiento ocular (puede ser de utilidad en futuros proyectos) y facial.
- Cámara RGB frontal que te permite obtener imágenes del entorno de forma nítida y hacer desarrollo para realidad mixta.
- Mandos HyperSense.
- Contiene altavoces estereo dobles por si en un futuro se quiere incluir audio en el simulador.
- CPU Qualcomm Snapdragon XR2 Con la potencia del chipset Snapdragon™ XR2, PICO 4 Enterprise brinda una experiencia de VR ultrafluida con 256 GB de almacenamiento y 8 GB DDR5 de RAM.
- Campo de visión de 105°.
- Resolución máxima de 4320 × 2160 (1200 ppp) y una frecuencia de actualización de 90 Hz
- Lentes Pancake, que son delgadas y ligeras.
- Diseño delgado y Ligero.

Las gafas se pueden ver en la siguiente figura (Figura [13]):

Figura 13: Gafas de Realidad Virtual PICO 4 Enterprise

3.2) Métodos

3.2.1) Plataforma en tiempo real

Este primer apartado consiste en obtener en tiempo real los frames que se captan mediante la cámara de un portátil, hacer el procesado de dichos frames para conseguir la visión artificial y por último presentar por pantalla todo el proceso por el que pasa el frame original hasta llegar a la percepción, guardando los frames y los vídeos generados. Además, lo que se va a hacer es medir el tiempo que se tarda en generar una percepción durante un tiempo. Para ello se ha utilizado el programa Spyder para poder realizar un script en lenguaje Python que realiza lo siguiente:

1. Se hace una conexión con la cámara.
2. Se elije tanto el modelo computacional como la neuroprótesis que se van a utilizar, que pueden ser tanto de corteza visual como de retina.
3. Se crea un bucle en el que se lleva a cabo el siguiente proceso, inicializando un contador para ver cuánto dura el bucle:
    1. Se obtiene un frame, que en este caso es una imagen RGB (array de 3 dimensiones: [height(número de píxeles en el eje Y), width (número de píxeles en el eje X), channels (cada color tiene un canal) = 3]).
    2. Se genera el frame fosfenizado gracias a la función create_percept(), cuya entrada va a ser el frame captado, y además se mide la duración de dicho proceso. Dicha función hace lo siguiente:
        1. Se transforman los píxeles del frame en estímulos eléctricos, obteniendo así un vector de tamaño height x width x 3 estímulos.
        2. Esos estímulos se transforman a escala de grises (se pasa a un vector de tamaño height x width estímulos) y se invierte.
        3. Se aplica un filtro para obtener los bordes que contiene el frame. Los filtros a utilizar pueden ser los siguientes:
            1. Sobel: Este filtro tiene como objetivo obtener los bordes que se encuentran en una imagen obteniendo sus gradientes de intensidad, es decir, las diferencias. Concretamente, cuando hay una diferencia muy grande de intensidad en una imagen se dice que hay un borde.
            2. Scharr: Este filtro es muy parecido al anterior, salvo que se usan otro tipo de operadores que hacen que el cálculo del gradiente sea más preciso.
            3. Canny: Este filtro también obtiene los bordes de imágenes y además es bastante robusto al ruido.
            4. Median: Este filtro no lineal tiene el objetivo de eliminar el ruido de las imágenes .
        4. Se aplica un ensanchado de los bordes para que se vean mejor.

5. Se eliminan los bordes negros alrededor del frame y se hace un ajuste del tamaño al tamaño de la neuroprótesis para así añadirlos como entrada a dicha neuroprótesis.
6. Se genera la percepción visual.

4. Una vez obtenida la percepción, lo que se va a hacer es mostrar por pantalla el frame original, en escala de grises invertido, con los bordes dilatados y la percepción para ver así todo el proceso desde que capta el frame hasta que se obtiene la percepción.
5. Se genera una condición en la cual si se pulsa el botón escape se sale del bucle y otra condición en la cual si se supera la duración del bucle se sale del bucle.
6. Una vez se sale del bucle, el último punto es hacer un guardado tanto la grabación de las pantallas como el último frame de las pantallas.

3.2.2) Plataforma en tiempo real con PICO 4 Enterprise

Una vez visto el esquema que sigue la plataforma a tiempo real, se va a proceder a implementarlo en las gafas PICO 4 Enterprise de forma inalámbrica. Para ello, los cambios que habría que hacer serían los siguientes:

- El primer paso es poder tener acceso a la cámara RGB de las gafas PICO 4 Enterprise para poder tener los frames. Para ello nos tuvimos que poner en contacto con la Empresa PICO debido a que, por protección de datos, no se tiene acceso directo a esta información. Una vez proporcionados los permisos, se pueden obtener los datos de la cámara de PICO 4 Enterprise.
- Después de poder obtener los datos de la cámara, es necesario hacer una conexión inalámbrica entre PICO 4 Enterprise y el PC. Para ello se ha recurrido a hacer una conexión TCP/IP, la cual es la identificación del grupo de protocolos de red que hacen posible la transferencia de datos en redes, entre equipos informáticos e internet. Las siglas TCP/IP hacen referencia a este grupo de protocolos:
    - TCP es el Protocolo de Control de Transmisión que permite establecer una conexión y el intercambio de datos entre dos anfitriones. Este protocolo proporciona un transporte fiable de datos.
    - IP o protocolo de internet, utiliza direcciones series de cuatro octetos con formato de punto decimal (como por ejemplo 75.4.160.25). Este protocolo lleva los datos a otras máquinas de la red.

Hay que tener en cuenta que tanto las gafas como el PC tienen que estar conectados a la misma red.
Una vez dicho esto, el esquema de trabajo sería el siguiente:

1. Conexión TCP/IP entre el PC y las gafas.
2. Lo siguiente se hace de forma continua:
    1. Se obtiene un frame de la cámara de las gafas.
    2. Se manda dicho frame al PC dado su respectiva dirección IP y puerto (antes tanto el tamaño del frame como el frame en sí se tiene que codificar en un paquete para mandarlo).
    3. En el PC en python se recibe el paquete y se decodifica para obtener el frame.
    4. Se hace un procesado para obtener la percepción como en el apartado anterior.
    5. Se tiene que codificar la percepción para mandarla a las gafas.
    6. Una vez mandada la percepción se decodifica y se muestra por pantalla.

4) Resultados y discusión

4.1) Simulación en tiempo real

A continuación se van a mostrar imágenes obtenidas después de todo el proceso explicado en el apartado 3.2.1 para distintos casos:

- Distintas neuroprótesis:
  - IMIE: Es una neuroprótesis epiretinal de 256 electrodos.
  - Argus II
  - Cortivis
  - ORION
- Distintos modelos: En este caso se ha decidido para la retina usar los modelo tanto Scoreboard como Axon Map y en el caso de retina Scoreboard. No se aplican modelos que apliquen distorsiones temporales ya que es muy lento para el uso del simulador en tiempo real (caso de corteza visual el tiempo que se emplea en generar la percepción es de 5 segundos y en el de corteza visual)
- Distintos filtros: Se van a usar los filtros Scharr (Sobel no, ya que es muy parecido), Canny y Median.

Primero se va a presentar cuál es la imagen original que se ha capturado (Figura [14]):

Figura 14: Imagen original en cada uno de los casos.

Esta imagen tiene una resolución de 640 x 480 píxeles.

Una vez dicho esto, se va a mostrar el proceso que pasa la imagen una ve se ha obtenido:

1. El primer paso que se sigue es pasar la imagen a escala de grises e invertir los colores para así poder conseguir una mejor distinción de los bordes. Una vez hecho esto, se obtiene la siguiente imagen (Figura [15]):

Figura 15: Imagen en escala de grises invertida

2. El siguiente paso consiste en obtener los bordes de la imagen con el filtro y ensanchar dichos bordes. Como tenemos 3 para comparar, se va a presentar la imagen con cada uno de los filtros (Figuras [16-18]):

Figura 16: Imagen con filtro Scharr

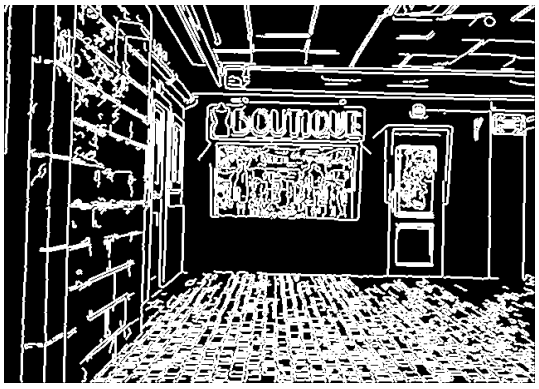

Figura 17: Imagen con filtro Canny

Figura 18: Imagen con filtro Median

Se puede observar cómo con el filtro Canny se tienen bordes más nítidos que en el caso del filtro Scharr y en el caso de Median se ve que es prácticamente igual que la figura 15.

3. Ya cuando se ha filtrado la imagen ya es hora de obtener la percepción y, ya que se tienen diferentes filtros, modelos y neuroprótesis, se tienen distintos resultados (Figuras [19-36]):

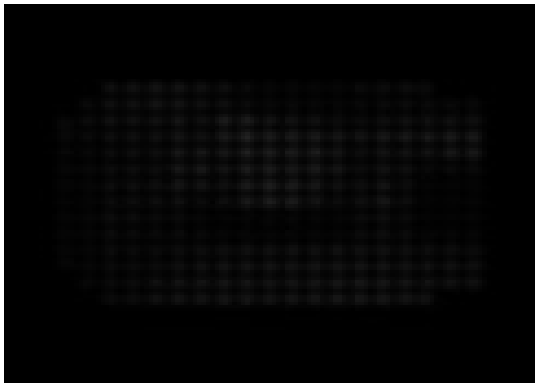

Figura 19: Percepción con la neuroprótesis IMIE, el modelo Scoreboard y el filtro Scharr

Se puede observar en la figura 19 que la imagen es poco nítida debido al poco brillo, y además se puede ver perfectamente la forma del fosfeno generado por el modelo, que al ser ideal es un punto.

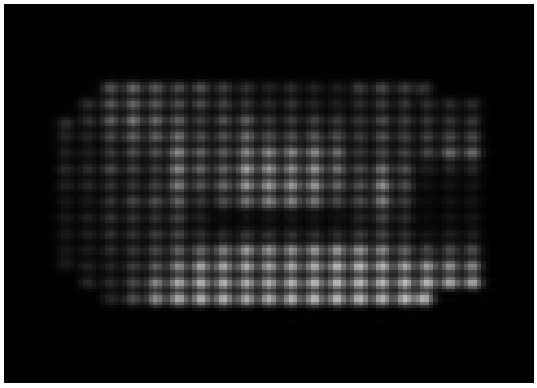

Figura 20: Percepción con la neuroprótesis IMIE, el modelo Scoreboard y el filtro Canny

Se puede observar en la figura 20 que la imagen tiene más brillo que en el caso de la figura anterior, pudiendo ser más nítida y distinguiendo mejor los bordes que contiene la imagen.

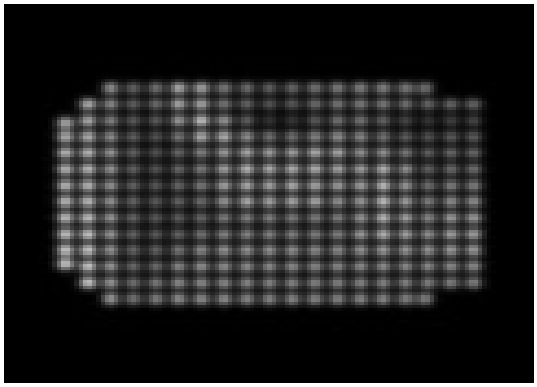

Figura 21: Percepción con la neuroprótesis IMIE , el modelo Scoreboard y el filtro Median

Se puede observar que en la figura 21 hay más brillo pero es menos nítida que la figura anterior.

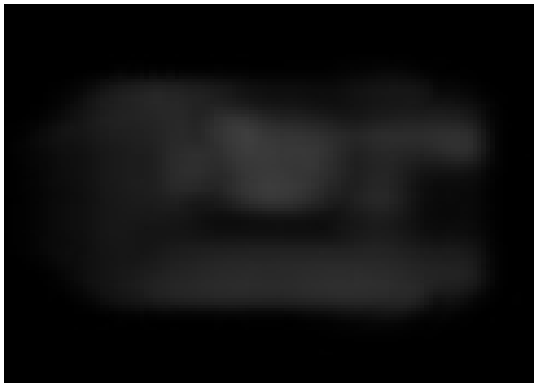

Figura 22: Percepción con la neuroprótesis IMIE, el modelo AxonMap y el filtro Scharr

Se puede observar en la figura 22 que, en comparación con la figura 19, los fosfenos ahora tienen otra forma que hace que la percepción que se forma sea un poco más parecida a la imagen original debido a que ahora se tienen en cuenta los axones que se puedan llegar a estimular.

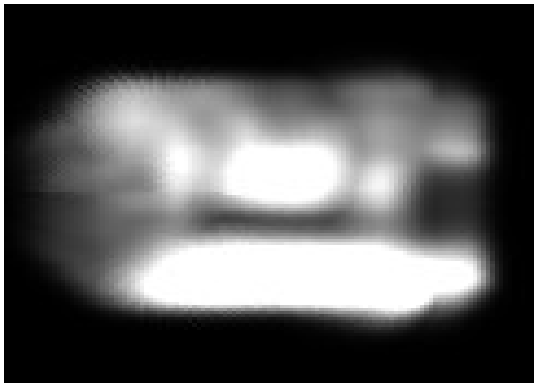

Figura 23: Percepción con la neuroprótesis IMIE, el modelo AxonMap y el filtro Canny

Se puede ver que en la figura 23 que, con respecto a la figura anterior, se ven mejor los bordes de la imagen original.

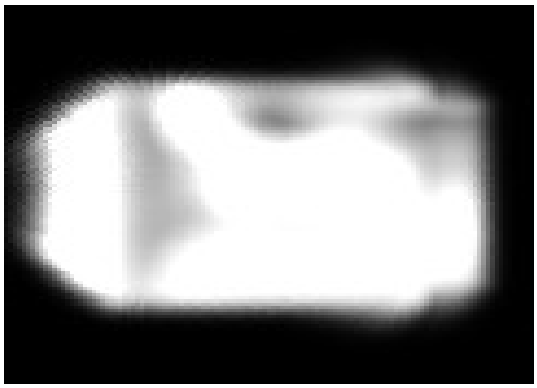

Figura 24: Percepción con la neuroprótesis IMIE, el modelo AxonMap y el filtro Median

Se puede observar en la figura 24 que tiene mucho más brillo que en el caso anterior.

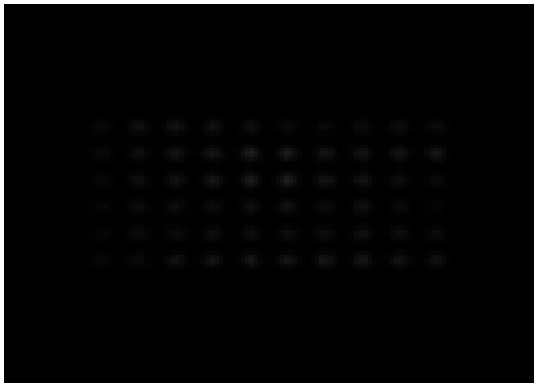

Figura 25: Percepción con la neuroprótesis Argus II, el modelo Scoreboard y el filtro Scharr

Se puede ver en la figura 25 que ahora se tienen muchos menos fosfenos que en el caso de IMIE y están más separados, por lo que la nitidez es peor.

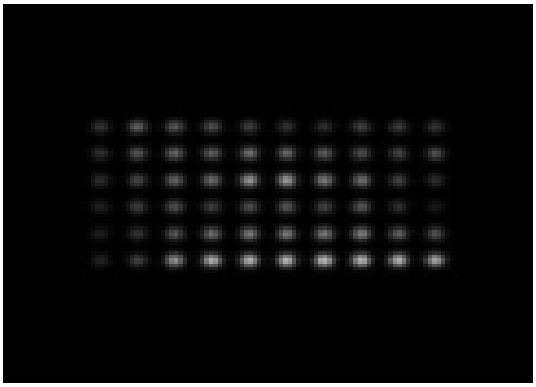

Figura 26: Percepción con la neuroprótesis Argus II, el modelo Scoreboard y el filtro Canny

Se puede ver en la figura 26 que con respecto a la imagen anterior se ve con más brillo.

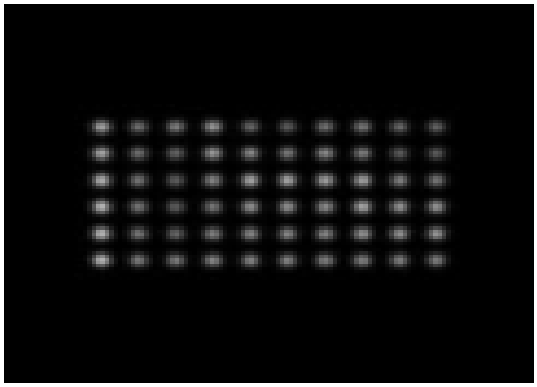

Figura 27: Percepción con la neuroprótesis Argus II, el modelo Scoreboard y el filtro Median

Se puede observar en la figura 27 que aquí hay más brillo aún.

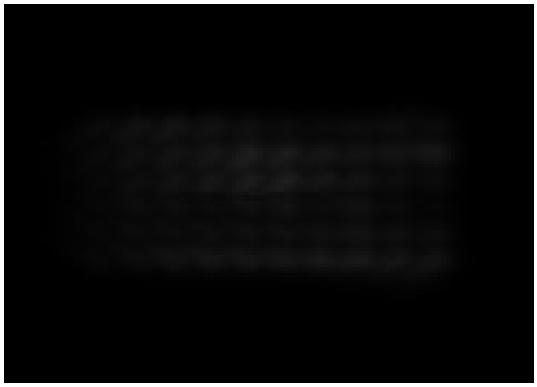

Figura 28: Percepción con la neuroprótesis Argus II, el modelo AxonMap y el filtro Scharr

Se puede observar en la figura 28 que, con respecto a la neuroprótesis IMIE, se puede apreciar mucho mejor la forma que tiene el fosfeno cuando se usa el mapa de axones, pero hay menos nitidez.

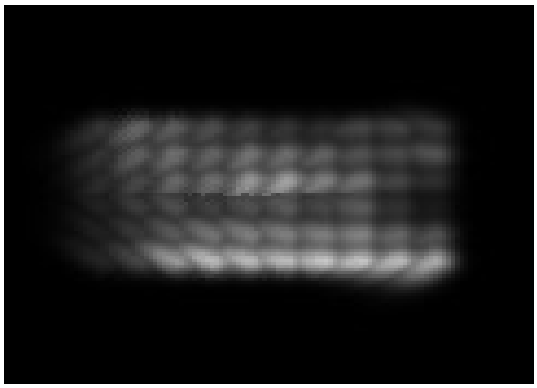

Figura 29: Percepción con la neuroprótesis Argus II, el modelo AxonMap y el filtro Canny

Se puede observar en la figura 29 que, con respecto a la figura anterior, hay más brillo pero no hay una buena nitidez.

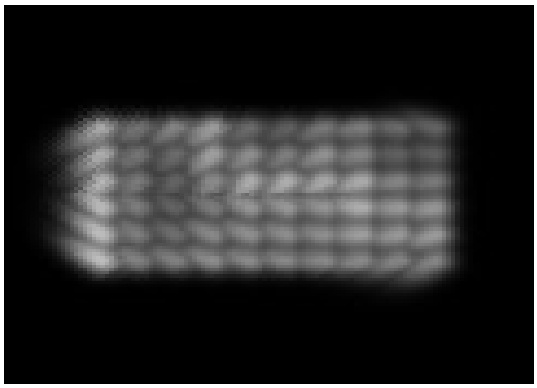

Figura 30: Percepción con la neuroprótesis Argus II, el modelo AxonMap y el filtro Median

Se puede observar en la figura 30 que hay mucho más brillo y no se distingue nada.

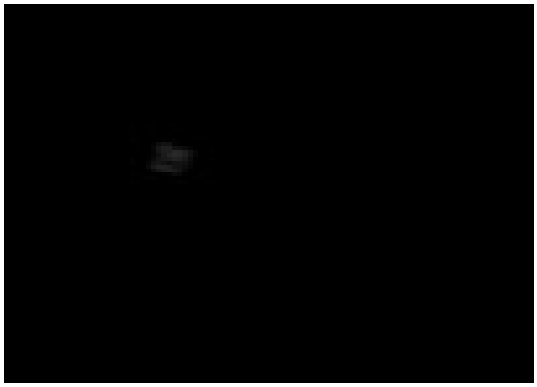

Figura 31: Percepción con la neuroprótesis CORTIVIS, el modelo Scoreboard y el filtro Scharr

Se puede apreciar en la figura 31 que, con respecto a los dos implantes anteriores, la región donde hay fosfenos es mucho menor, no pudiendo hacer distinciones.

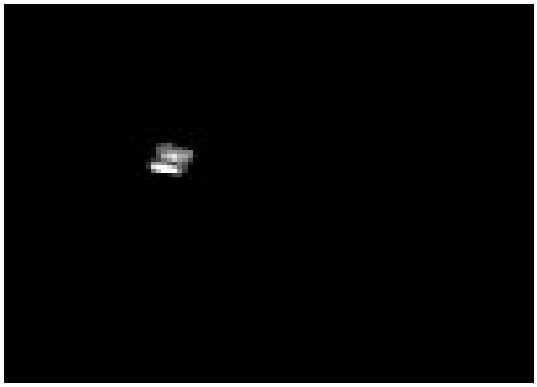

Figura 32: Percepción con la neuroprótesis CORTIVIS, el modelo Scoreboard y el filtro Canny

Se puede observar en la figura 32 que hay más brillo que en la anterior pero sigue sin haber nitidez.

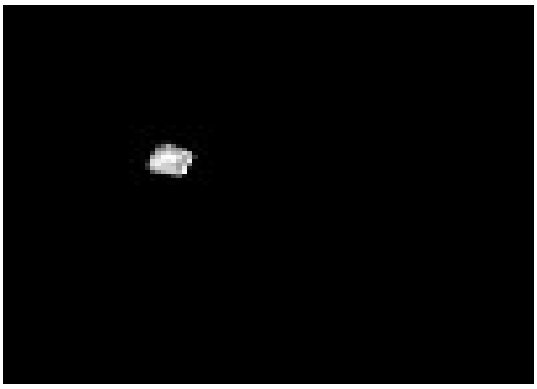

Figura 33: Percepción con la neuroprótesis CORTIVIS, el modelo Scoreboard y el filtro Median

Se puede observar en la figura 33 que el brillo aumenta mucho más con respecto al anterior pero no hay nitidez.

Figura 34: Percepción con la neuroprótesis ORION el modelo Scoreboard y el filtro Scharr

Se puede observar en la figura 34 que con esta neuroprótesis los fosfenos aparecen en forma hexagonal. No se pueden distinguir bordes.

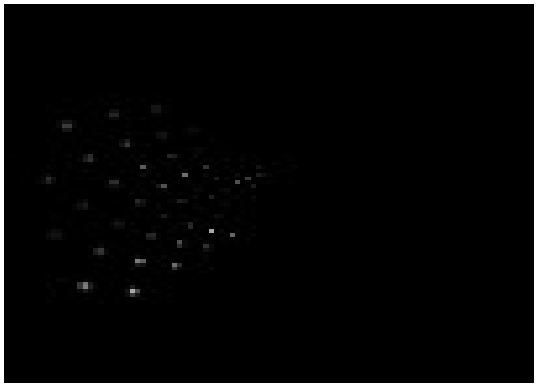

Figura 35: Percepción con la neuroprótesis ORION el modelo Scoreboard y el filtro Canny

Se puede observar en la figura 35 que, con respecto a la anterior, hay más brillo pero no se pueden distinguir figuras.

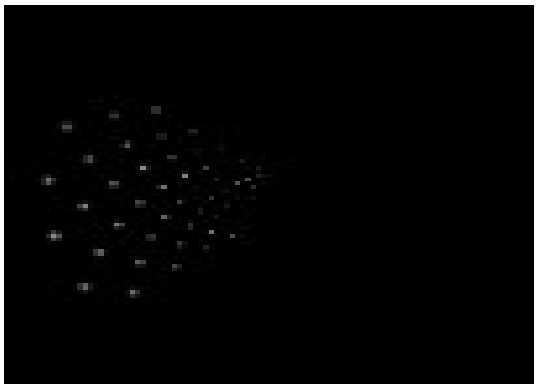

Figura 36: Percepción con la neuroprótesis ORION el modelo Scoreboard y el filtro Median

Se puede observar en la figura 36 que, con respecto a la anterior, hay más brillo pero no se pueden distinguir figuras.

Una vez hecho esto, se ha analizado el tiempo que se tarda en hacer el procesado de la imagen. Más concretamente, se ha medido el tiempo que se tarda en generar una percepción durante 50 frames. Los resultados que se obtuvieron fueron los siguientes (Figuras [37-54]):

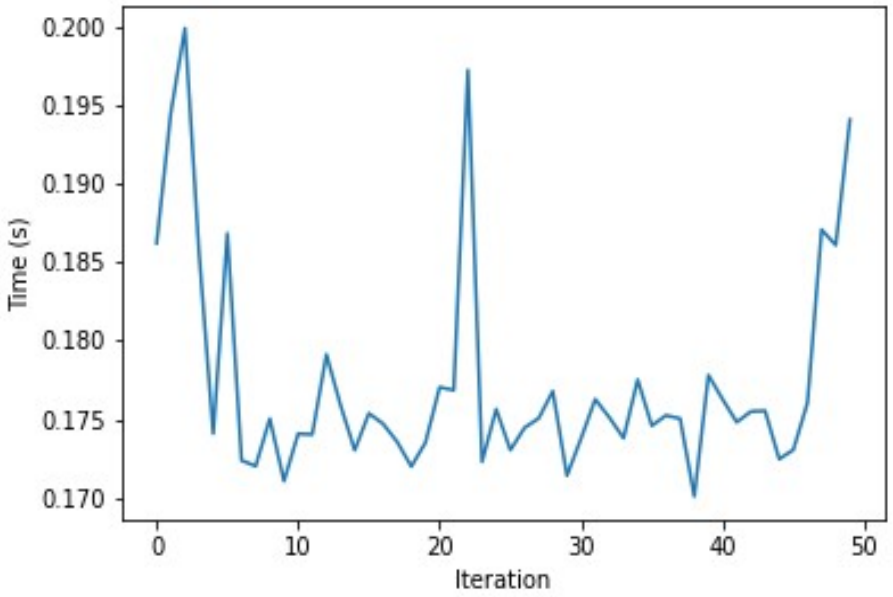

Figura 37: Tiempo de obtención de la percepción con la neuroprótesis IMIE, el modelo Scoreboard y el filtro Scharr

Se puede observar que el tiempo que tarda está en el orden de los milisegundos.

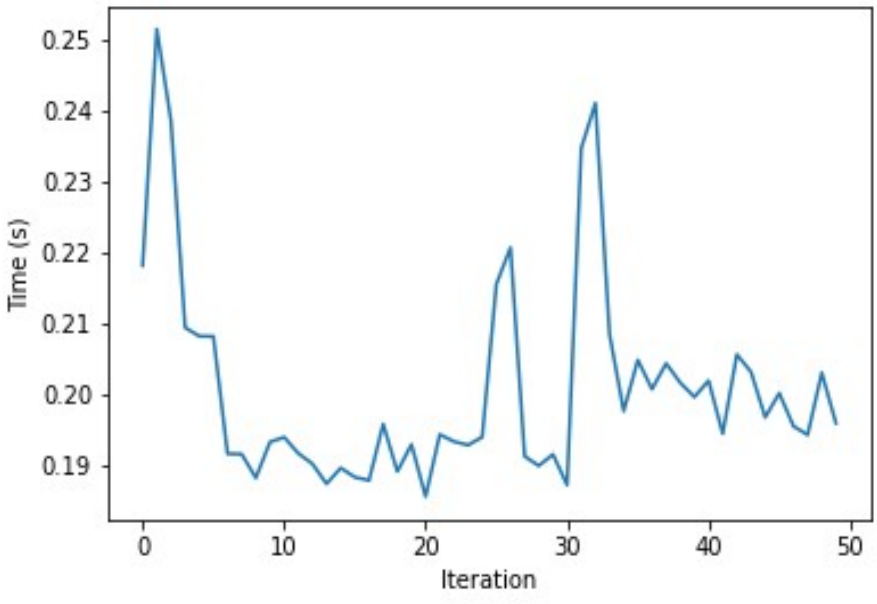

Figura 38: Tiempo de obtención de la percepción con la neuroprótesis IMIE, el modelo Scoreboard y el filtro Canny

Se puede apreciar que en el caso del filtro Canny el tiempo medio a la hora de obtener la percepción aumenta un poco.

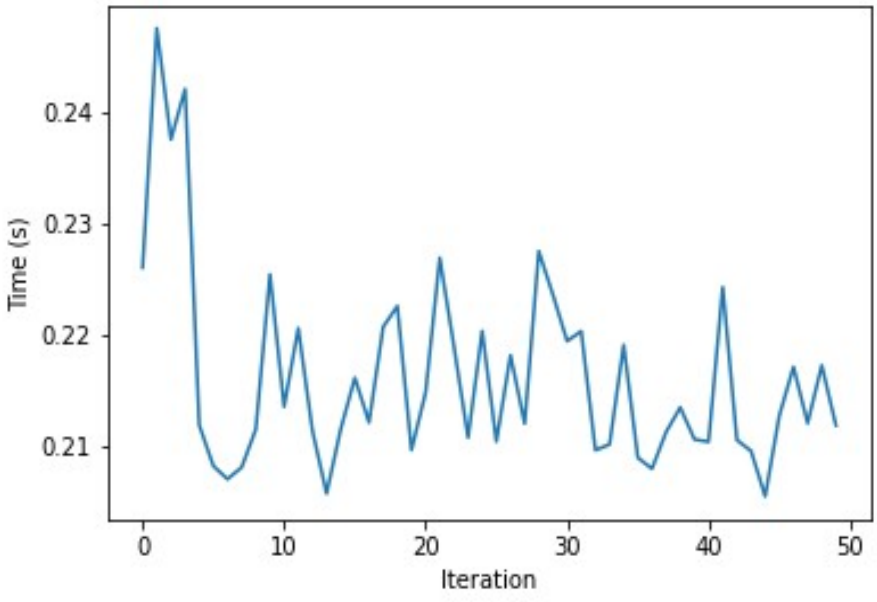

Figura 39: Tiempo de obtención de la percepción con la neuroprótesis IMIE, el modelo Scoreboard y el filtro Median

Se puede observar que en este caso el tiempo medio es mayor en promedio con respecto a los otros dos casos.

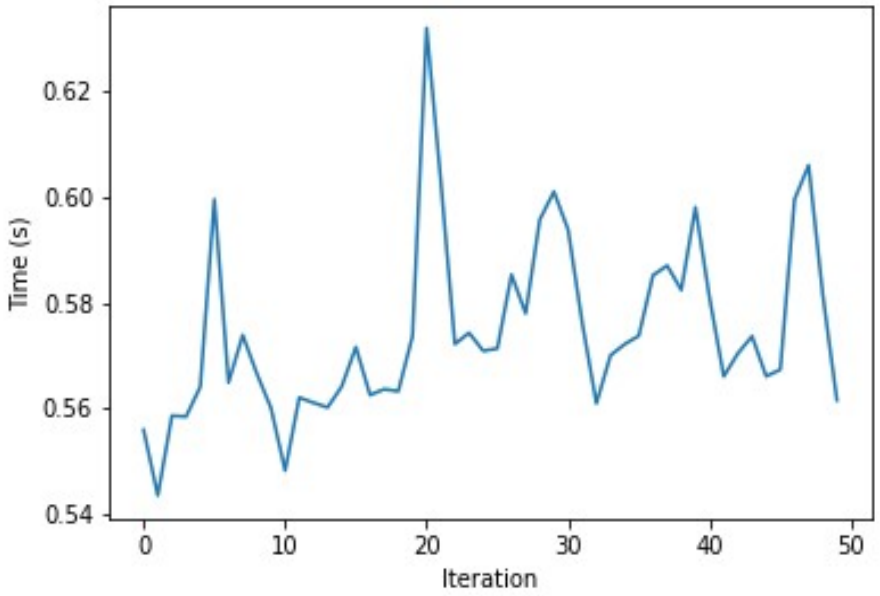

Figura 40: Tiempo de obtención de la percepción con la neuroprótesis IMIE, el modelo AxonMap y el filtro Scharr

Se puede observar que con respecto a las figuras anteriores, el tiempo medio ha aumentado bastante debido al modelo, llegando al medio segundo.

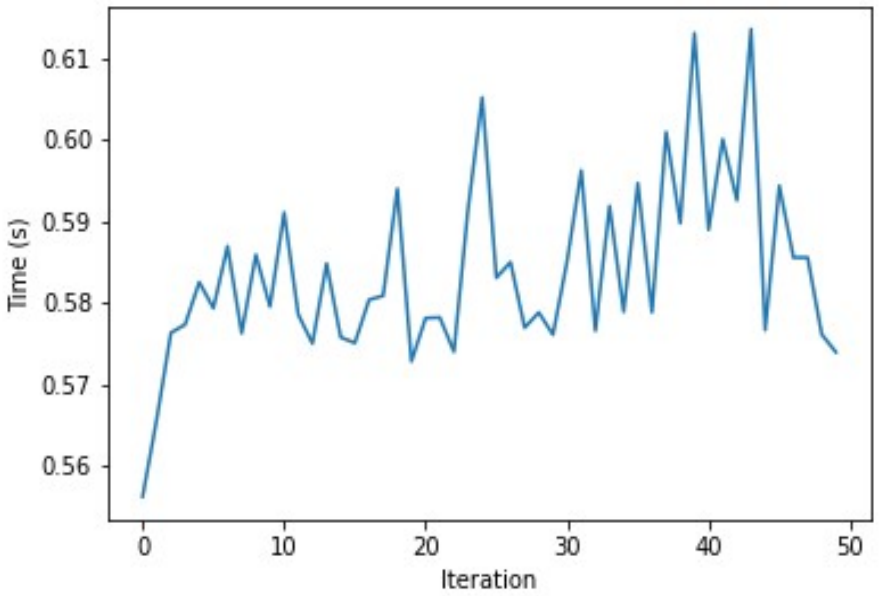

Figura 41: Tiempo de obtención de la percepción con la neuroprótesis IMIE, el modelo AxonMap y el filtro Canny

Se puede observar que el tiempo medio es mayor que en la figura anterior y también mayor que en la figura 38.

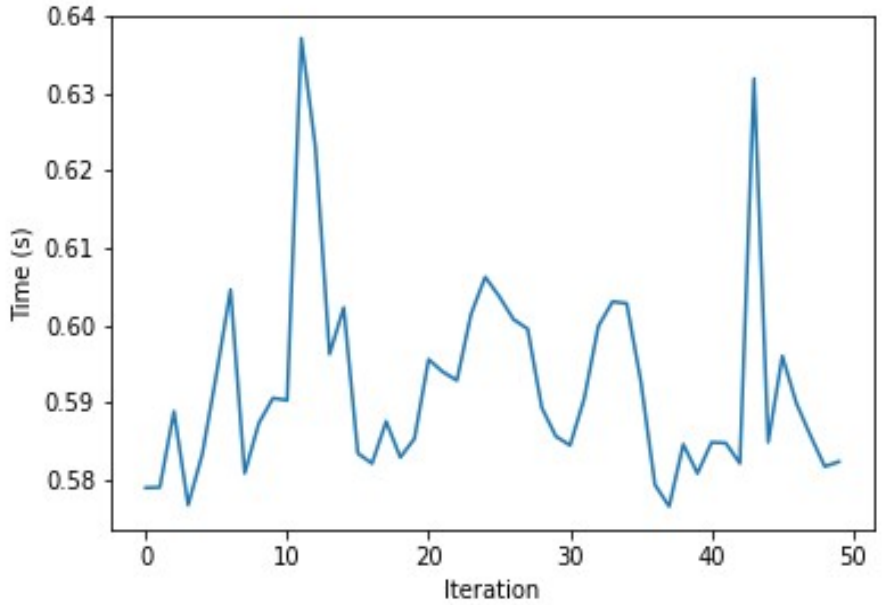

Figura 42: Tiempo de obtención de la percepción con la neuroprótesis IMIE, el modelo AxonMap y el filtro Median

Se puede ver que en este caso el tiempo es mayor que en los otros dos casos anteriores y también mayor que en el caso de la figura 39. Es el de mayor valor.

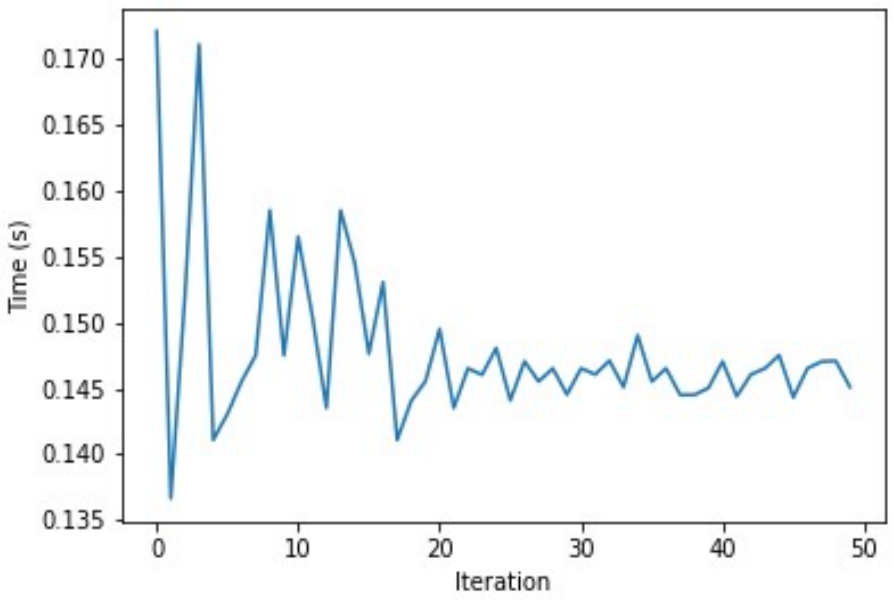

Figura 43: Tiempo de obtención de la percepción con la neuroprótesis Argus II, el modelo Scoreboard y el filtro Scharr

Se puede observar que este es el que tiene menor tiempo de generación de percepciones.

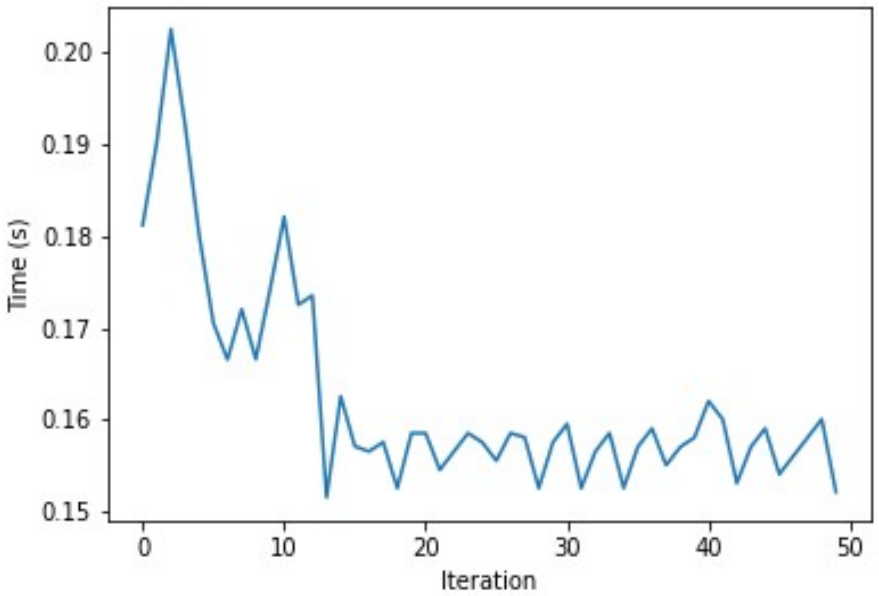

Figura 44: Tiempo de obtención de la percepción con la neuroprótesis Argus II, el modelo Scoreboard y el filtro Canny

Se puede apreciar que el tiempo ha aumentado con respecto a la figura anterior y ha disminuido con respecto a las figuras 41 y 38.

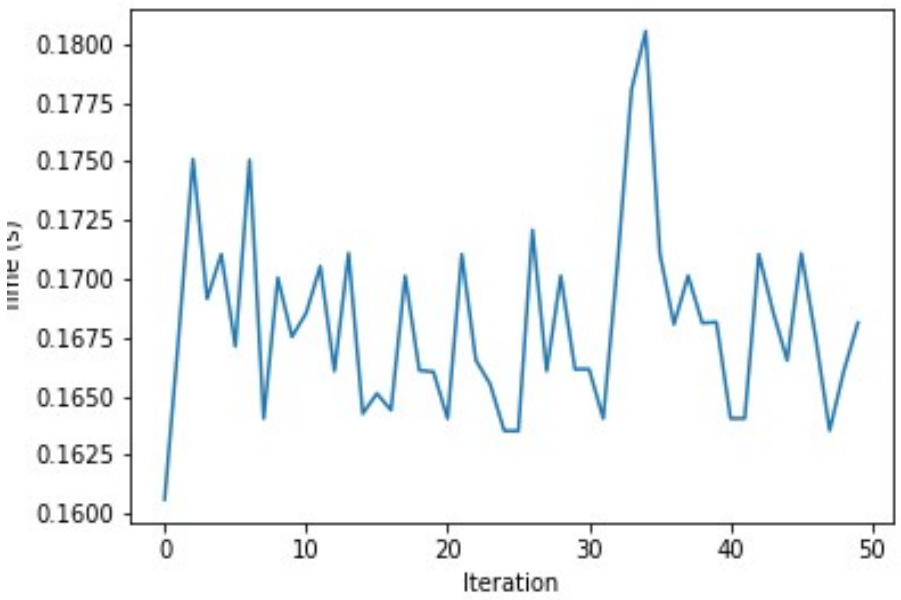

Figura 45: Tiempo de obtención de la percepción con la neuroprótesis Argus II, el modelo Scoreboard y el filtro Median

Se puede apreciar que el tiempo ha aumentado con respecto a la figura anterior y ha disminuido con respecto a las figuras 42 y 39.

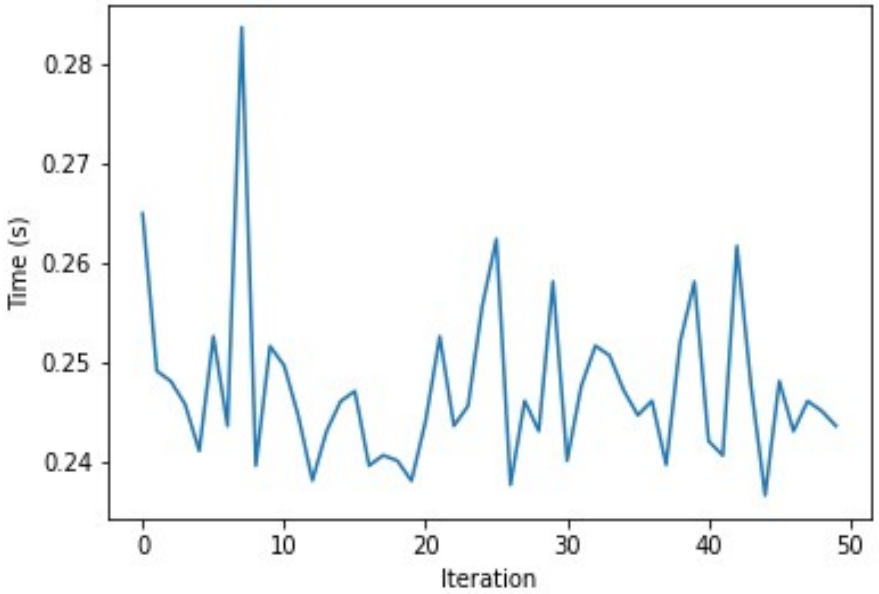

Figura 46: Tiempo de obtención de la percepción con la neuroprótesis Argus II, el modelo AxonMap y el filtro Scharr

Se puede ver en este frame que el tiempo medio ha aumentado con respecto al caso de la figura 43 pero es menor que en el caso de la figura 40.

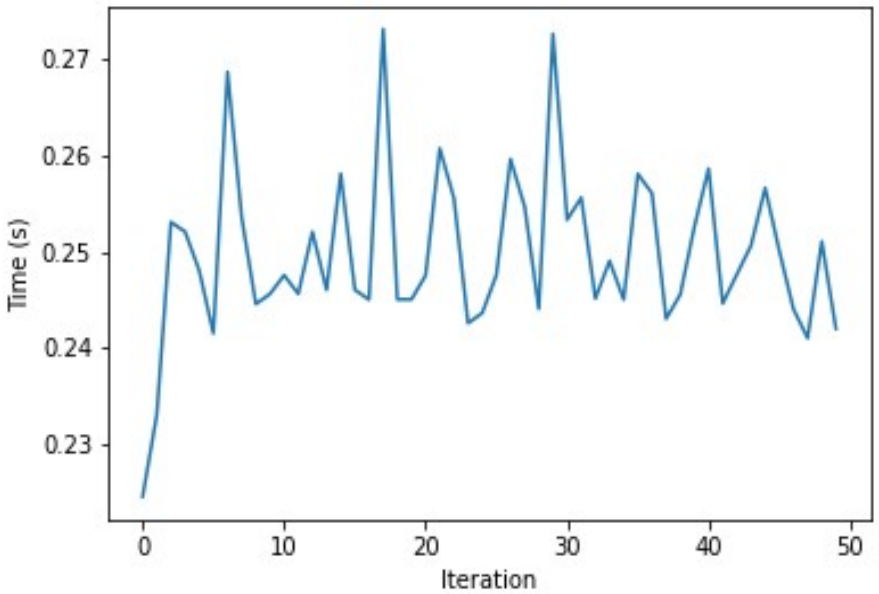

Figura 47: Tiempo de obtención de la percepción con la neuroprótesis Argus II, el modelo AxonMap y el filtro Canny

Se puede ver en este frame que el tiempo medio ha aumentado con respecto al caso de la figura 44 pero es menor que en el caso de la figura 41.

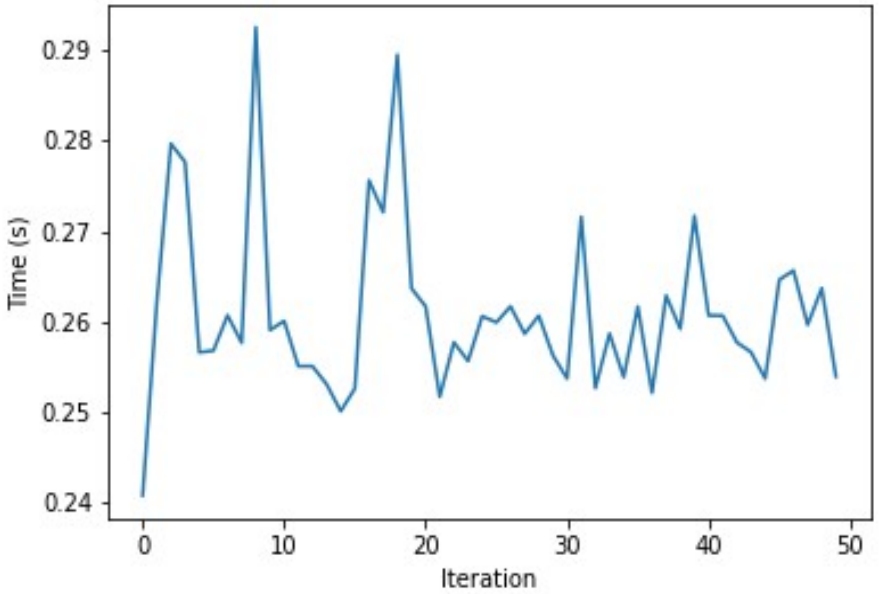

Figura 48: Tiempo de obtención de la percepción con la neuroprótesis Argus II, el modelo AxonMap y el filtro Median

Se puede ver en este frame que el tiempo medio ha aumentado con respecto al caso de la figura 45 pero es menor que en el caso de la figura 42.

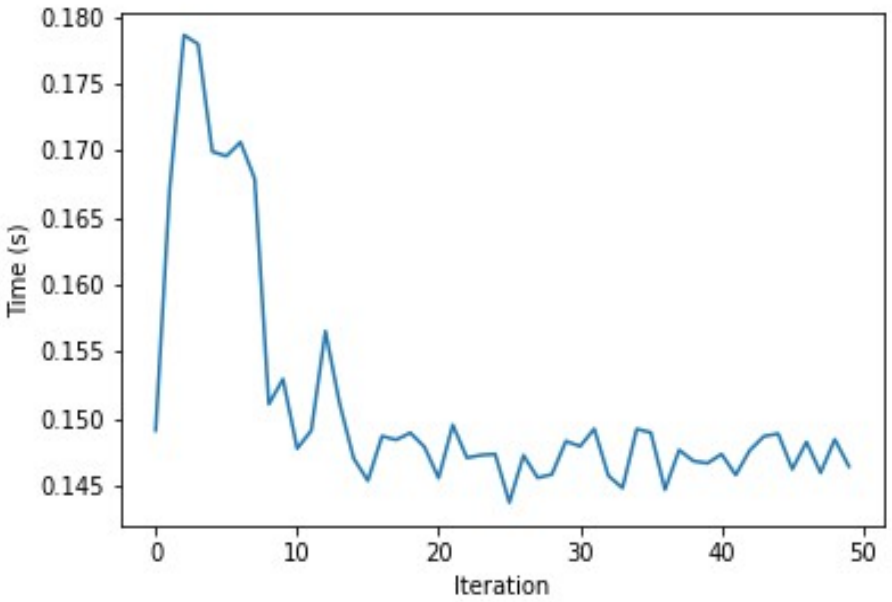

Figura 49: Tiempo de obtención de la percepción con la neuroprótesis CORTIVIS, el modelo Scoreboard y el filtro Scharr

Se puede observar que en este caso el tiempo es menor que en el resto de casos presentados menos el de Argus II con modelo Scoreboard.

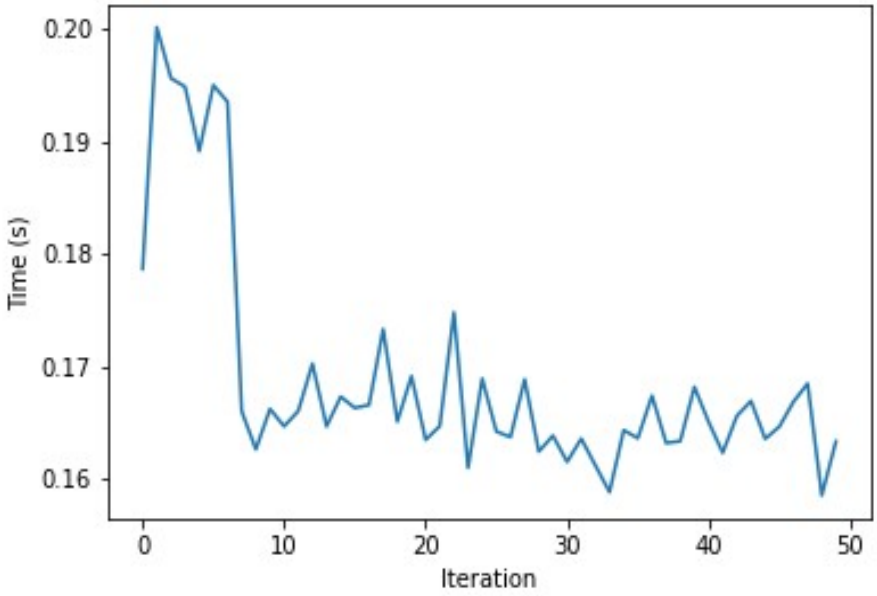

Figura 50: Tiempo de obtención de la percepción con la neuroprótesis CORTIVIS, el modelo Scoreboard y el filtro Canny

Se puede observar que con respecto a la figura anterior el tiempo medio aumenta un poco.

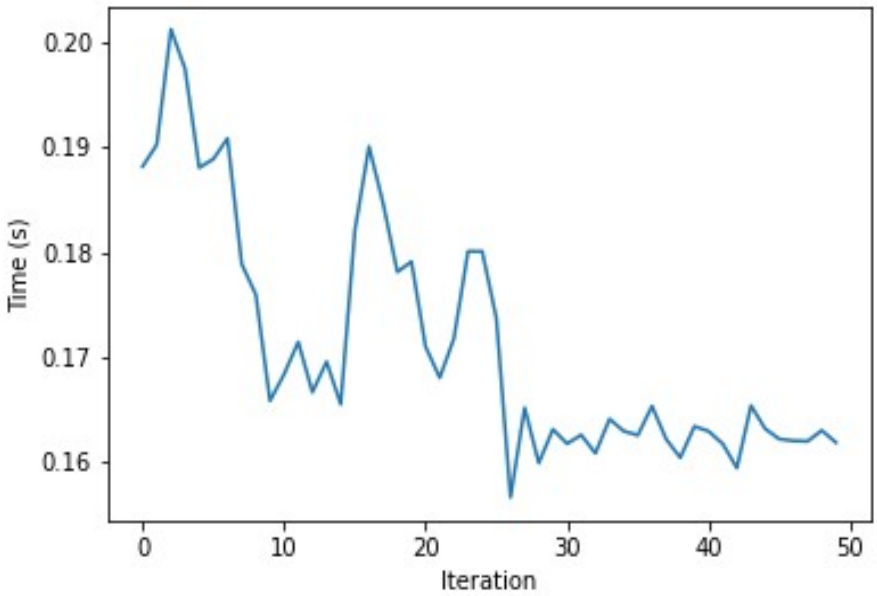

Figura 51: Tiempo de obtención de la percepción con la neuroprótesis CORTIVIS, el modelo Scoreboard y el filtro Median

Se puede observar que el tiempo medio aumenta con respecto a la figura anterior.

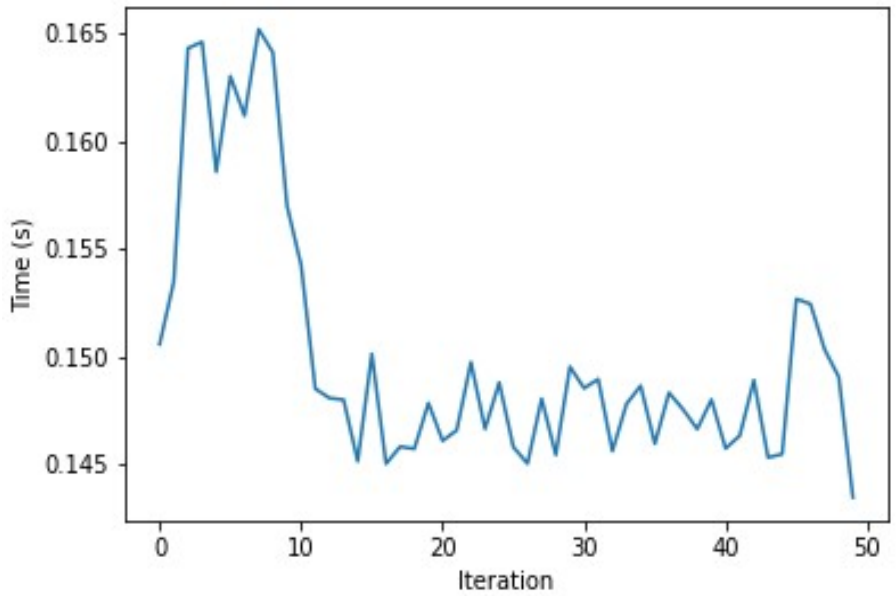

Figura 52: Tiempo de obtención de la percepción con la neuroprótesis ORION, el modelo Scoreboard y el filtro Scharr

Se puede apreciar que el tiempo medio es muy parecido al de la figura 49.

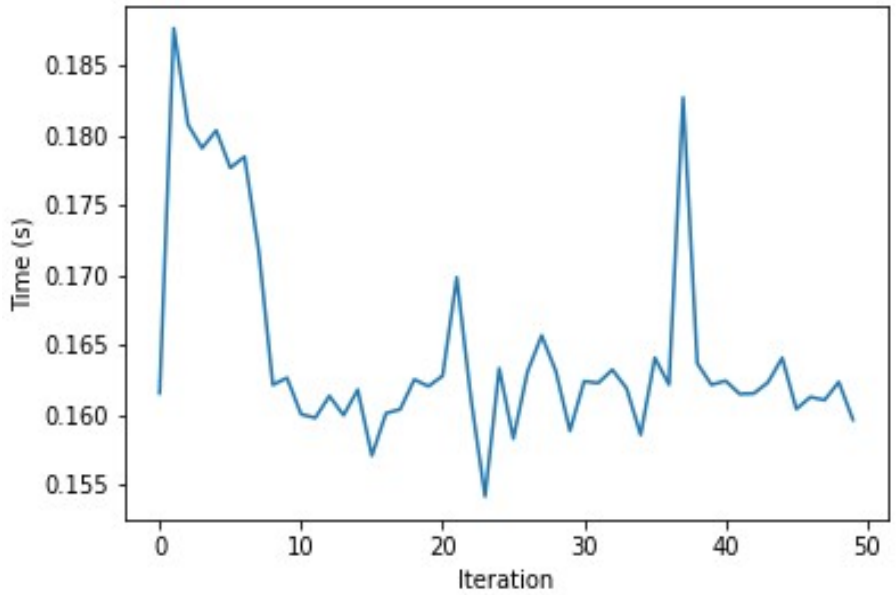

Figura 53: Tiempo de obtención de la percepción con la neuroprótesis ORION, el modelo Scoreboard y el filtro Canny

Se puede observar que el tiempo aumenta con respecto a la figura anterior.

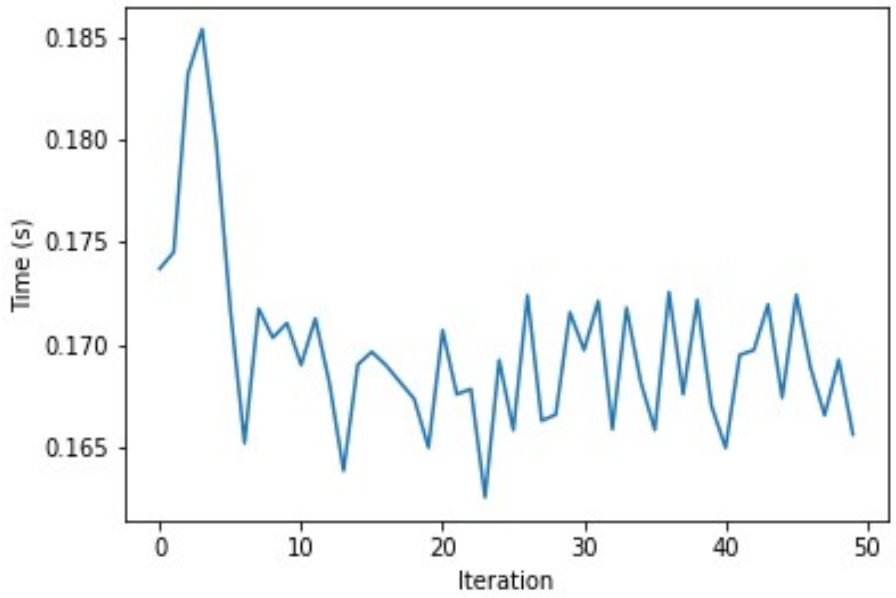

Figura 54: Tiempo de obtención de la percepción con la neuroprótesis ORION, el modelo Scoreboard y el filtro Canny

Se puede observar que el tiempo aumenta con respecto a la figura anterior.

Una vez analizado el tiempo, también se ha procedido a obtener la resolución de las imágenes en cada uno de los casos, obteniendo lo siguiente (figuras [55-56]):

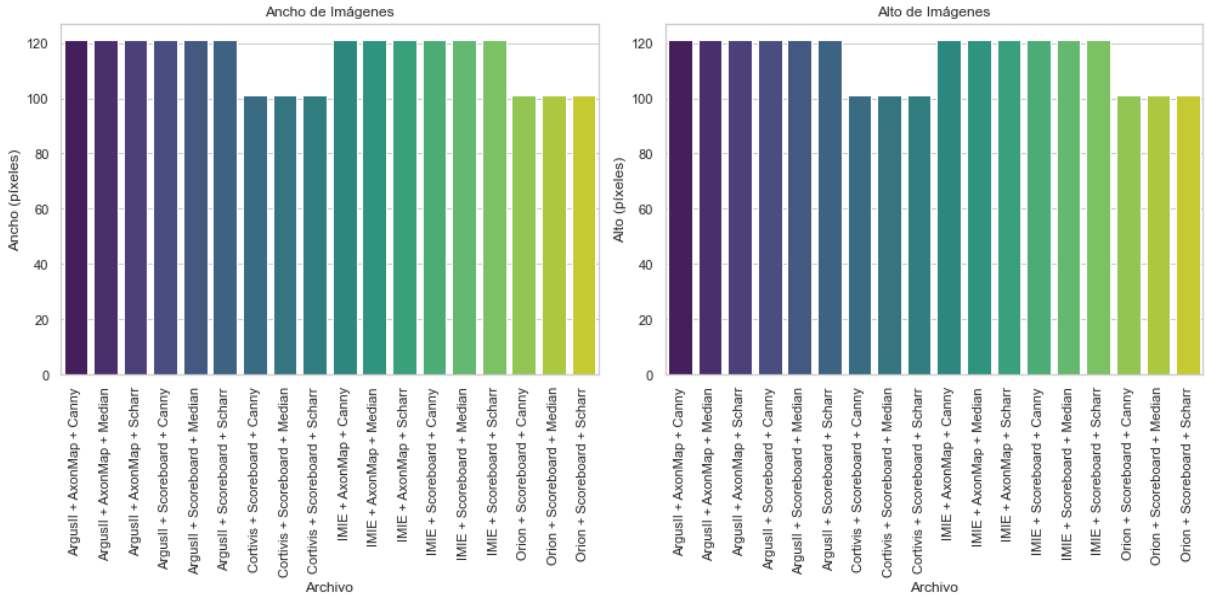

Figura 55: Resolución en píxeles de las percepciones cada una de las distintas combinaciones para el ancho y alto

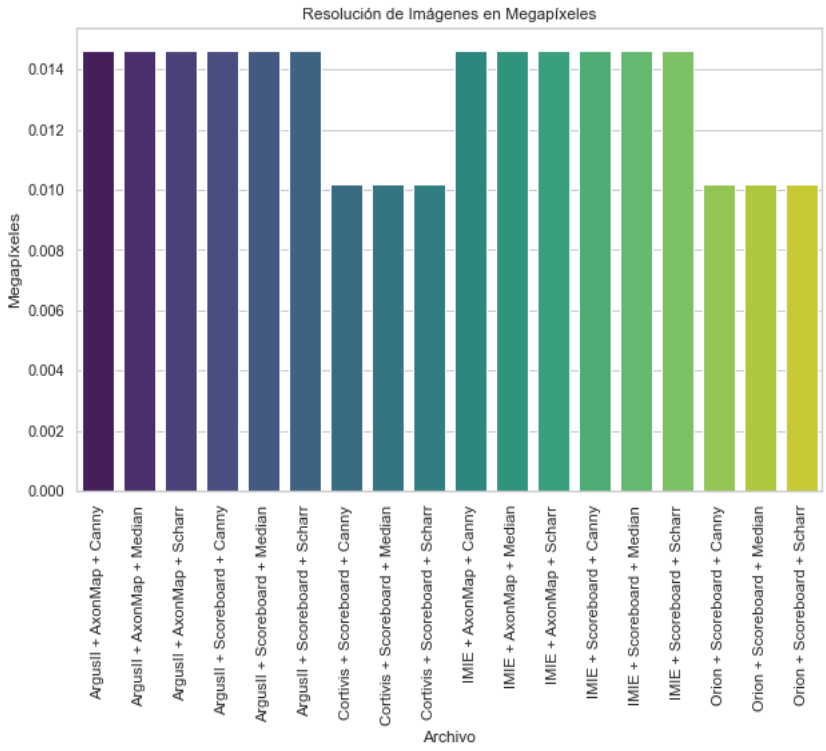

Figura 56: Resolución de las percepciones en megapíxeles de cada una de las distintas combinaciones

Se puede observar que en todos los casos la resolución ha disminuido con respecto al frame original la resolución de las percepciones disminuye, pero en el caso de las neuroprótesis de corteza visual la resolución es menor que en el caso de las neuroprótesis de retina.

A modo de discusión, se van a plantear los siguientes puntos:

- En primer lugar, la plataforma permite hacer un procesado en tiempo real de los frames, cosa que por sí solo por ejemplo Pulse2Percept o OpticStim no implementan. Pulse2Percept en principio es un paquete el cual se usa para imágenes y vídeo, no para imágenes en tiempo real, y OpticStim se encarga de optimizar los patrones de estimulación del nervio óptico. En cambio, en el caso de Dynaphos sí que crea la percepción visual de frames a tiempo real, aunque está limitada a neuroprótesis de corteza visual, mientras que nuestra plataforma puede ser usada para neuroprótesis de retina y de corteza visual, pudiendo en un futuro añadir modelos y neuroprótesis de nervio óptico, ya que al basarse en el paquete Pulse2Percept añadir nuevos modelos e implantes no es una tarea demasiado complicada.
- Un punto que hay que tener en cuenta es el tiempo de computación, ya que, como se ha podido comprobar en el apartado de resultados, dependiendo del modelo, del implante, el tiempo invertido en la generación de las percepciones visuales puede aumentar mucho, cosa que también ocurre en Pulse2Percept y en Dynaphos.

4.2) Simulador en tiempo real en PICO 4 Enterprise

En el caso de la simulación a tiempo real con las gafas PICO 4 Enterprise lo que se ha conseguido hacer es obtener los frames de la cámara de las PICO 4 Enterprise, se ha podido también hacer una conexión TCP/IP entre las gafas y el PC y además se han podido mandar de forma inalámbrica los frames al PC y se han podido presentar por pantalla con una velocidad relativamente buena. El problema surge a la hora de hacer el procesamiento para obtener las percepciones: lo que sucede es que el tamaño de los datos del frame es menor que el tamaño de dicho frame, lo que quiere decir que hay un error a la hora de recibir los paquetes.

6) Conclusiones

En este último apartado se van a exponer las conclusiones a las que se han llegado, además de enfoques futuros a este trabajo.

Para empezar, se ha estudiado de forma profunda el funcionamiento del paquete de Python Pulse2Percept, comprendiendo el uso de los distintos implantes, modelo y filtros y la generación de percepciones visuales dados estímulos provenientes de imágenes.

A continuación, se ha demostrado que es viable hacer un simulador a tiempo real de la visión artificial. Esto se ha probado usando diferentes tipos de neuroprótesis, modelos tanto de retina como de corteza visual, y filtros para poder hacer una evaluación de su efecto a la hora de obtener las percepciones, resultando en que la elección de estas variables es crucial para optimizar la generación de la percepción a tiempo real en términos de calidad, resolución y tiempo para así mejorar la experiencia del usuario y la viabilidad de su implementación en entornos clínicos.

Después de esto, se ha intentado explorar la posible integración del simulador en las gafas de realidad virtual PICO 4 Enterprise y de forma inalámbrica, pudiendo capturar los frames provenientes de la cámara RGB de gafas y pudiendo hacer una conexión TCP/IP inalámbrica para así poder enviar dichos frames al PC, pero sin éxito a la hora de poder realizar el procesamiento y mandar las percepciones visuales de vuelta a las gafas.

En trabajos futuros, como no se ha terminado de llevar a cabo el desarrollo del simulador en las gafas PICO 4 Enterprise, el siguiente paso es terminar de implementarlo, y además también sería interesante añadir la funcionalidad de eyetracking o seguimiento ocular para obtener la información de los movimientos de los ojos, y para terminar también faltaría hacer distintos ensayos de comportamiento con pacientes sanos.

7) Bibliografía

[1] Beyeler, M., Boynton, G. M., Fine, I., & Rokem, A. (2017). Pulse2Percept: A Python-based simulation framework for bionic vision. BioRxiv, 148015.

[2] Romero, S. F., Morillas, C. A., Pelayo, F. J., & Fernández, E. (2008). Computer-Controlled Neurostimulation for a Visual Implant. In BIODEVICES (1) (pp. 84-91).

[3] van Steveninck, J. D. R., Güçlü, U., van Wezel, R., & van Gerven, M. (2022). End-to-end optimization of prosthetic vision. Journal of Vision, 22(2), 20-20.

[4] Rassia, K. E. K., & Pezaris, J. S. (2018). Improvement in reading performance through training with simulated thalamic visual prostheses. Scientific Reports, 8(1), 16310.

[5] Van Der Grinten, M., de Ruyter van Steveninck, J., Lozano, A., Pijnacker, L., Rückauer, B., Roelfsema, P., ... & Güçlütürk, Y. (2022). Biologically plausible phosphene simulation for the differentiable optimization of visual cortical prostheses. BioRxiv, 2022-12.

[6] Chen, S. C., Suaning, G. J., Morley, J. W., & Lovell, N. H. (2009). Simulating prosthetic vision: I. Visual models of phosphenes. Vision research, 49(12), 1493-1506.

[7] Chen, S. C., Suaning, G. J., Morley, J. W., & Lovell, N. H. (2009). Simulating prosthetic vision: II. Measuring functional capacity. Vision research, 49(19), 2329-2343.

[8] Zhou, D. D., Dorn, J. D., & Greenberg, R. J. (2013, July). The Argus® II retinal prosthesis system: An overview. In 2013 IEEE international conference on multimedia and expo workshops (ICMEW) (pp. 1-6). IEEE.

[9] Stingl, K., Schippert, R., Bartz-Schmidt, K. U., Besch, D., Cottriall, C. L., Edwards, T. L., ... & Zrenner, E. (2017). Interim results of a multicenter trial with the new electronic subretinal implant alpha AMS in 15 patients blind from inherited retinal degenerations. Frontiers in neuroscience, 11, 445.

[10] Ayton, L. N., Blamey, P. J., Guymer, R. H., Luu, C. D., Nayagam, D. A., Sinclair, N. C., ... & Bionic Vision Australia Research Consortium. (2014). First-in-human trial of a novel suprachoroidal retinal prosthesis. PloS one, 9(12), e115239.

[11] Fernández, E., & Normann, R. A. (2017). CORTIVIS approach for an intracortical visual prostheses. Artificial vision: A clinical guide, 191-201.

[12] Luo, Y. H. L., & Da Cruz, L. (2014). A review and update on the current status of retinal prostheses (bionic eye). British medical bulletin, 109(1), 31-44.

[13] Pulse2Percept 0.9.0.dev0 documentation — Pulse2Percept 0.9.0.dev0 documentation. (s.f.). https://Pulse2Percept.readthedocs.io/en/latest/index.html

[14] Vicente, J. M. F., Álvarez-Sánchez, J. R., de la Paz López, F., & Adeli, H. (Eds.). (2022). Artificial Intelligence in Neuroscience: Affective Analysis and Health Applications: 9th International Work-

Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2022, Puerto de la Cruz, Tenerife, Spain, May 31–June 3, 2022, Proceedings, Part I (Vol. 13258). Springer Nature.

[15] Liu, X., Chen, P., Ding, X., Liu, A., Li, P., Sun, C., & Guan, H. (2022). A narrative review of cortical visual prosthesis systems: the latest progress and significance of nanotechnology for the future. Annals of Translational Medicine, 10(12).

[16] Devi, S. K., & Subalalitha, C. N. (2022). Deep learning based audio assistive system for visually impaired people. Comput. Mater. Continua, 71(1), 1205-1219.

[17] Beyeler, M., Nanduri, D., Weiland, J. D., Rokem, A., Boynton, G. M., & Fine, I. (2019). A model of ganglion axon pathways accounts for percepts elicited by retinal implants. Scientific reports, 9(1), 9199.

[18] Huff, T., Mahabadi, N., & Tadi, P. (2018). Neuroanatomy, visual cortex.

[19] Veraart, C., Wanet-Defalque, M. C., Gérard, B., Vanlierde, A., & Delbeke, J. (2003). Pattern recognition with the optic nerve visual prosthesis. Artificial organs, 27(11), 996-1004.

[20] Veraart, C., Raftopoulos, C., Mortimer, J. T., Delbeke, J., Pins, D., Michaux, G., ... & Wanet-Defalque, M. C. (1998). Visual sensations produced by optic nerve stimulation using an implanted self-sizing spiral cuff electrode. Brain research, 813(1), 181-186.

[21] Wang, V., & Kuriyan, A. E. (2020). Optoelectronic devices for vision restoration. Current ophthalmology reports, 8, 69-77.

[22] Romeni, S., Zoccolan, D., & Micera, S. (2021). A machine learning framework to optimize optic nerve electrical stimulation for vision restoration. Patterns, 2(7).

[23] Sui, X., Li, L., Chai, X., Wu, K., Zhou, C., Sun, X., ... & Ren, Q. (2009). Visual prosthesis for optic nerve stimulation. Implantable Neural Prostheses 1: Devices and Applications, 43-83.

[24] Fisher, L. E., Tyler, D. J., & Triolo, R. J. (2013). Optimization of selective stimulation parameters for multi-contact electrodes. Journal of neuroengineering and rehabilitation, 10, 1-8.

[25] Wang, J., Cao, P., Li, L., & Li, M. (2014, October). A preliminary simulation study of virtual channel generated by penetrating optic nerve electrical stimulation. In 2014 7th International Conference on Biomedical Engineering and Informatics (pp. 522-526). IEEE.

[26] Ramirez, K. A., Drew-Bear, L. E., Vega-Garces, M., Betancourt-Belandria, H., & Arevalo, J. F. (2023). An update on visual prosthesis. International Journal of Retina and Vitreous, 9(1), 73.

[27] Sakaguchi, H., Kamei, M., Nishida, K., Terasawa, Y., Fujikado, T., Ozawa, M., & Nishida, K. (2012). Implantation of a newly developed direct optic nerve electrode device for artificial vision in rabbits. Journal of Artificial Organs, 15, 295-300.

8) Anexos

TFM_script.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu May 23 12:06:58 2024

@author: maray

Este script se encarga de transformar a tiempo real frames en percepciones dado
un modelo y un implante de retina o de corteza visual.

Esto es para ejemplo.

"""

'--------------------------------------------------------------------------'
'                          Librerías                                       '
'--------------------------------------------------------------------------'

# Para conectarme con la cámara.

import cv2

# Para usar pulse2percept.

import pulse2percept as p2p

# Filtrado.

from skimage.morphology import dilation

# Numpy.
```

```
31
32 import numpy as np
33
34 # Figuras.
35
36 import matplotlib.pyplot as plt
37
38 # Fecha.
39
40 from datetime import datetime
41
42 #Crear un path
43
44 from pathlib import Path
45
46 # Crear carpeta.
47
48 import os
49
50 # Librería de la medida de tiempo.
51
52 import time
53
54 '------------------------------------------------------------------------'
55 '                        Ruta de records                        '
56 '                        ---------------                    '
57 'La función de esta parte del script es crear una carpeta donde se guarden los'
58 'resultados (imágenes y vídeos).                                '
59 '------------------------------------------------------------------------'
60
61 # Ruta donde van a ir los resultados.
62
63
64 # my_implant = "ArgusII"
65 # my_implant = "IMIE"
```

```
66 # my_implant = "Cortivis"
67 my_implant = "Orion"
68 my_model = "ScoreboardModel"
69 # my_model = "AxonMapModel"
70 # my_filter = "Scharr"
71 # my_filter = "Canny"
72 my_filter = "Median"
73
74 configuration = my_implant + " + " + my_model + " + " + my_filter
75
76
77  routelog = r'C:\Documentos Mar\UMH\Segundo cuatrimestre\TFM\pulse2percept-master\
examples\stimuli\results'
78
79 # Fecha de hoy (para el path).
80
81 today = datetime.now().strftime("%Y_%m_%d")
82
83 # Fecha de hoy pero en string y con hora, minuto y segundo (para el nombre de
84 # los archivos.
85
86 dt_string = datetime.now().strftime("%d.%m.%Y.%H.%M.%S")
87
88 # Creación del path donde se van a guardar los logs con fecha la de hoy.
89
90 pth = Path('{}\logs{}\{}'.format(routelog, today, configuration))
91 os.makedirs(pth, exist_ok=True)
92
93 '-------------------------------------------------------------------------'
94 '                    Función para la percepción                    '
95 '                    -------------------------                      '
96 'Esta función tiene como objetivo transforma un frame en percepción visual.   '
97 'Input:                                                              '
98 ' - my_frame: imagen que se va a procesar.                           '
99 'Output:                                                             '
```

```
' - my_image_gray.data: El frame en escala de grises e invertida.          '
' - my_image_edge.data: Los bordes del frame.                               '
' - my_image_dilate.data: Los bordes del frame.                             '
' - my_image_percept.data: La percepción generada.                          '
'--------------------------------------------------------------------------'

def create_percept(my_frame):

    # Creación del estímulo.

    my_stim = p2p.stimuli.ImageStimulus(my_frame)

    # Conversión a escala de grises e invertir.

    my_stim_gray = my_stim.invert().rgb2gray()

    # Usar filtro para obtener los bordes.

    # my_stim_edge = my_stim_gray.filter('scharr')
    # my_stim_edge = my_stim_gray.filter('canny')
    my_stim_edge = my_stim_gray.filter('median')

    # Hacer los bordes más anchos.

    my_stim_dilate = my_stim_edge.apply(dilation)

    # Cambiar tamaño del estímulo al del implante y eliminar bordes negros
    # alrededor de la imagen, para al final colocarlo en el implante.

    implant.stim = my_stim_dilate.trim().resize(implant.shape)

    # Predicción de la percepción dado el modelo y el implante.

    my_stim_percept = model.predict_percept(implant)

```

```
135     # Se devuelven los datos.
136
137     return my_stim_gray.data, my_stim_dilate.data, my_stim_percept.data
138
139 '--------------------------------------------------------------------------'
140 '                    Conexión con la cámara                    '
141 '                    ----------------------                    '
142 'Esta parte del script tiene como objetivo hacer una conexión con una webcam o'
143 'usar un vídeo y preparar las capturas de vídeo de los frames procesados.    '
144 '--------------------------------------------------------------------------'
145
146 # Podemos capturar los frames de la webcam del ordenador o de la webcam de las
147 # gafas de realidad virtual:
148
149 cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
150                 # cap = cv2.VideoCapture('C:/Documentos Mar/UMH/Segundo
cuatrimestre/TFM/streetlab.mp4')
151 # cap = cv2.VideoCapture(1)
152
153 # Lectura de un frame.
154
155 ret, frame = cap.read()
156
157 # Obtención del tamaño del frame.
158
159 h_frame, w_frame = frame.shape[:2]
160
161 # Preparación de los vídeos.
162
163 fourcc = cv2.VideoWriter_fourcc(*'XVID')
164     out_frame = cv2.VideoWriter('{}/video_frame_{}.avi'.format(pth, configuration),fourcc,20.0,
(w_frame,h_frame))
165                 out_frame_gray = cv2.VideoWriter('{}/video_frame_gray_{}.avi'.format(pth,
configuration),fourcc,20.0,(w_frame,h_frame),False)
```

```
166             out_frame_edge      =       cv2.VideoWriter('{}/video_frame_edge_{}.avi'.format(pth,
configuration),fourcc,20.0,(w_frame,h_frame),False)
167         out_frame_percept     =     cv2.VideoWriter('{}/video_frame_percept_{}.avi'.format(pth,
configuration),fourcc,20.0,(w_frame,h_frame),False)
168
169 '------------------------------------------------------------------------'
170 '                 Selección del modelo y el implante                    '
171 '                 ----------------------------------                    '
172 'Esta parte del script tiene como objetivo seleccionar un modelo de corteza   '
173 'visual o de retina y un implante, para después representarlo.              '
174 '------------------------------------------------------------------------'
175
176 # Se construye el modelo que se va a seguir de retina o de corteza visual.
177
178 # model = p2p.models.ScoreboardModel()
179 # model = p2p.models.AxonMapModel()
180 model = p2p.models.cortex.ScoreboardModel()
181 model.build()
182
183 # Trabajo en paralelo para ir más rápido.
184
185 model.engine = 'joblib'
186
187 # Selección del implante que se va a usar:
188
189 # implant = p2p.implants.ArgusI()
190 # implant = p2p.implants.ArgusII()
191 # implant = p2p.implants.PRIMA()
192 # implant = p2p.implants.PRIMA75()
193 # implant = p2p.implants.PRIMA55()
194 # implant = p2p.implants.PRIMA40()
195 # implant = p2p.implants.BVT24()
196 # implant = p2p.implants.AlphaAMS()
197 # implant = p2p.implants.AlphaIMS()
198 # implant = p2p.implants.IMIE()
```

```
199 # implant = p2p.implants.cortex.Cortivis()
200 implant = p2p.implants.cortex.Orion()
201 # implant = p2p.implants.cortex.ICVP()
202
203 # Representación de ambos:
204
205 plt.figure()
206 implant.plot()
207 model.plot()
208
209 # Guardado de la imagen.
210
211 plt.savefig('{}/model+implant_{}.png'.format(pth, configuration))
212
213 '--------------------------------------------------------------------------'
214 '                        Bucle principal                                   '
215 '                        ---------------                                   '
216 'Esta parte del script tiene como objetivo capturar los frames a tiempo real y'
217 'procesarlos para después presentar por pantalla tanto el frame original como '
218 'cada paso del procesado. Esto se hace gracias a un bucle while.           '
219 '--------------------------------------------------------------------------'
220
221 # Array donde se van a guardar los tiempos que se tarda en generar la
222 # percepción.
223
224 tiempos = []
225
226 # Índice que dice el número de frames por el que vamos.
227
228 n_frame = 0
229
230 # Número de frames totales.
231
232 total_frames = 50
233
```

```
234 # Frame a frame (si no supera el límite).
235
236 while n_frame < total_frames:
237
238     # Se aumenta en 1 el índice.
239
240     n_frame = n_frame + 1
241
242     # Lectura del frame de la cámara.
243
244     ret, frame = cap.read()
245
246     # Obtención de los frames procesados (blanco y negro, bordes y
247     # percepción).
248
249     inicio = time.time()    # Inicio generación de la percepción
250     frame_gray, frame_edge, frame_percept = create_percept(frame)
251     fin = time.time()       # Fin generación de la percepción
252
253     # Cálculo de la diferencia de tiempos: Duración.
254
255     tiempos.append(fin-inicio)
256
257
258     # Cambio del tamaño de los frames (blanco y negro y bordes).
259
260     frame_gray = np.reshape(frame_gray,(h_frame,w_frame))
261     frame_edge = np.reshape(frame_edge,(h_frame,w_frame))
262
263     # Creación de las ventanas donde se enseñan los frames.
264
265     cv2.namedWindow('Webcam', cv2.WINDOW_NORMAL)
266     cv2.setWindowProperty('Webcam', cv2.WND_PROP_TOPMOST, 1)
267     cv2.namedWindow('Gray', cv2.WINDOW_NORMAL)
268     cv2.setWindowProperty('Gray', cv2.WND_PROP_TOPMOST, 1)
```

```
    cv2.namedWindow('Edge', cv2.WINDOW_NORMAL)
    cv2.setWindowProperty('Edge', cv2.WND_PROP_TOPMOST, 1)
    cv2.namedWindow('Percept', cv2.WINDOW_NORMAL)
    cv2.setWindowProperty('Percept', cv2.WND_PROP_TOPMOST, 1)

    # Enseñar los frames.

    cv2.imshow('Webcam',frame)
    cv2.imshow('Gray',frame_gray)
    cv2.imshow('Edge',frame_edge)
    cv2.imshow('Percept',frame_percept)

    # Guardar los frames para los vídeos.

    out_frame.write(frame)
    out_frame_gray.write((frame_gray* 255).astype(np.uint8))
    out_frame_edge.write((frame_edge* 255).astype(np.uint8))
    out_frame_percept.write((frame_percept* 255).astype(np.uint8))

    # Leer la tecla que se ha pulsado en la imagen.

    k = cv2.waitKey(20) & 0xFF
    if k == 27:        # Botón escape.

        # El bucle termina.

        break

'--------------------------------------------------------------------------'
'                        Fin del script                                    '
'                        --------------                                    '
'Esta parte del script tiene como objetivo terminar la captura, la grabación  '
'de los vídeos, borrar las ventanas y guardar los vídeos en el path.          '
'--------------------------------------------------------------------------'

```

```
# Guardado de la gráfica de los tiempos.

plt.figure()
plt.plot(tiempos)
plt.xlabel("Iteration")
plt.ylabel("Time (s)")
plt.savefig('{}/tiempo_{}.png'.format(pth, configuration))

# Terminar la captura.

cap.release()

# Terminar la grabación.

out_frame.release()
out_frame_gray.release()
out_frame_edge.release()
out_frame_percept.release()

# Eliminar las pantallas.

cv2.destroyAllWindows()

# Guardar los vídeos.

cv2.imwrite('{}/image_frame_{}.png'.format(pth, configuration), frame)
cv2.imwrite('{}/image_frame_gray_{}.png'.format(pth, configuration), frame_gray*255)
cv2.imwrite('{}/image_frame_edge_{}.png'.format(pth, configuration), frame_edge*255)
cv2.imwrite('{}/image_frame_percept_{}.png'.format(pth, configuration), frame_percept*255)
```

TFM_Resolution.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jun 13 15:52:03 2024
```

```
4
5 @author: maray
6 """
7
8 import os
9 from PIL import Image
10 import pandas as pd
11 import matplotlib.pyplot as plt
12 import seaborn as sns
13
14    routelog = r'C:\Documentos Mar\UMH\Segundo cuatrimestre\TFM\pulse2percept-master\
examples\stimuli\results\logs2024_06_19\Percept'
15
16 resoluciones = []
17
18 for archivo in os.listdir(routelog):
19    if archivo.endswith(('png', 'jpg', 'jpeg', 'bmp', 'gif')):
20          ruta_imagen = os.path.join(routelog, archivo)
21          with Image.open(ruta_imagen) as img:
22             archivo = archivo[0:len(archivo)-4]
23             ancho, alto = img.size
24             print(ancho)
25             resoluciones.append({'archivo': archivo, 'ancho': ancho, 'alto': alto})
26
27 df_resoluciones = pd.DataFrame(resoluciones)
28
29 df_resoluciones['megapixeles'] = (df_resoluciones['ancho'] * df_resoluciones['alto']) / 1e6
30
31 print(df_resoluciones.head())
32
33 # Configurar estilo de Seaborn
34 sns.set(style="whitegrid")
35
36 # Crear gráfico de barras para las resoluciones (ancho y alto por separado)
37 plt.figure(figsize=(14, 7))
```

```
38
39 # Subplot para el ancho
40 plt.subplot(1, 2, 1)
41 sns.barplot(x='archivo', y='ancho', data=df_resoluciones, palette="viridis")
42 plt.xticks(rotation=90)
43 plt.title('Ancho de Imágenes')
44 plt.xlabel('Archivo')
45 plt.ylabel('Ancho (píxeles)')
46
47 # Subplot para el alto
48 plt.subplot(1, 2, 2)
49 sns.barplot(x='archivo', y='alto', data=df_resoluciones, palette="viridis")
50 plt.xticks(rotation=90)
51 plt.title('Alto de Imágenes')
52 plt.xlabel('Archivo')
53 plt.ylabel('Alto (píxeles)')
54
55 plt.tight_layout()
56 plt.show()
57
58 # Crear gráfico de barras para los megapíxeles (opcional)
59 plt.figure(figsize=(10, 6))
60 sns.barplot(x='archivo', y='megapixeles', data=df_resoluciones, palette="viridis")
61 plt.xticks(rotation=90)
62 plt.title('Resolución de Imágenes en Megapíxeles')
63 plt.xlabel('Archivo')
64 plt.ylabel('Megapíxeles')
65 plt.show()
```