YART: Una herramienta interactiva para rehabilitación cognitiva

Martín Menchón

Universidad Nacional del Centro de la Provincia de Buenos Aires (UNCPBA)

Resumen La rehabilitación cognitiva, es un método terapéutico destinado a mejorar o compensar los déficits neurocognitivos producidos por procesos que afectan el normal funcionamiento cerebral. Diversas enfermedades neurológicas o afecciones psicológicas pueden acarrear dificultades en las capacidades de atención, memoria, lenguaje, razonamiento, organización, entre otras. A través de este tipo de terapias se procura restaurar esas funciones o compensarlas a través del aprendizaje de otras habilidades.
YART es una herramienta de *software* que permite realizar el seguimiento de pacientes con dificultades de coordinación visuomotora y visuoespacial a través de ejercicios interactivos. Además como funcionalidad complementaria se desarrolló e integró una aplicación de procesamiento de imágenes que, a partir de una cámara monocular de baja resolución, registra una estimación de hacia donde esta mirando el paciente y la correlaciona con el movimiento del puntero en ese momento. Los resultados obtenidos muestran que la herramienta es capaz de estimar el cuadrante hacia donde mira el paciente con una eficacia del 96 %.

Keywords: Rehabilitación Cognitiva, Head Pose Estimation

1. Introducción

La rehabilitación cognitiva, es un método terapéutico destinado a mejorar o compensar los déficits neurocognitivos producidos por procesos que afectan el normal funcionamiento cerebral. A través de la terapia de rehabilitación cognitiva se procura restaurar las capacidades de atención, memoria, lenguaje, razonamiento y organización, entre otras; o compensarlas a través del aprendizaje de otras habilitades [1].
Desde el punto de vista médico existen numerosos trabajos que muestran resultados beneficiosos en la aplicación de diferentes herramientas de *software* para el problema de rehabilitación cognitiva en pacientes neurológicos [2,3,4,5,6,7,8].
La idea central de este trabajo se basa en el desarrollo de una herramienta de tratamiento, rehabilitación y estimulación cognitiva para las personas que han sido afectadas por lesiones neurológicas (accidentes cerebrovasculares, anoxias, traumatismo craneoencefálico, tumores, etc.); enfermedades neurodegenerativas (Alzheimer; Parkinson, Esclerosis múltiple, etc.); o actividades de promoción y

prevención para personas con declinación cognitiva debido a la edad.
(Yet Another Rehabilitation Tool) YART permite contribuir a mejorar el seguimiento de tratamientos de rehabilitación cognitiva de forma personalizada, intensiva y con monitorización continua. La misma comprende dos componentes principales: el primero de ellos es una plataforma web con características de usabilidad acordes a pacientes con problemas cognitivos que permita obtener métricas útiles para el seguimiento de la rehabilitación por parte del profesional médico. El segundo componente comprende un módulo de estimación de la mirada que apunta a complementar las métricas anteriores.

2. Ejercicios de rehabilitación cognitiva

En este trabajo se implementaron ejercicios de rehabilitación diseñados por los responsables del servicio de Neurología Clínica del Hospital Italiano. En los mismos se divide la pantalla en cuatro cuadrantes definidos por dos líneas perpendiculares y se muestra cinco veces por cuadrante una figura de distinta forma y color en lugares aleatorios de la pantalla. Estas figuras aparecen durante un tiempo determinado y el usuario debe hacerle click ni bien las observa. Ver Fig. 1.
En estos ejercicios existen ciertos parámetros variables, como la transparencia de los cuadrantes, el tamaño de las figuras, el color de fondo, el color de las figuras, el tipo de figura (abstracta o real), el tiempo de exposición, la cantidad de elementos distractores (figuras que aparecen al comienzo del ejercicio y no desaparecen), la velocidad de aparición del objeto y una trayectoria (es decir, si se mueve o no).
Después de la realización del ejercicio se obtiene información relevante como el porcentaje de aciertos, el tiempo que demoró en resolver todos los ejercicios y la cantidad de clicks que realizó. En la Fig. 1 se pueden observar capturas de pantalla de la herramienta, donde se diferencian las distintas configuraciones posibles.

2.1. Decisiones de implementación

Un requisito de gran relevancia de este sistema es que los usuarios son pacientes que presentan capacidades diferentes, tanto motrices como cognitivas. Como se mencionó anteriormente, este requisito implica que el software debe tener la característica de ser fácilmente instalable y configurable.
Luego de considerar varias alternativas basadas en diversas tecnologías (ver apéndice sección B), se decidió implementar esta etapa como una aplicación web lo cual posee varias ventajas:
Al utilizar el navegador web como cliente ligero se logra la independencia del sistema operativo, así como la facilidad para actualizar y mantener aplicaciones sin distribuir e instalar *software* a un gran número de usuarios potenciales. De esta manera sólo con ingresar a la página web se está accediendo a la aplicación deseada. Cualquier cambio requerido se actualiza por única vez en el servidor

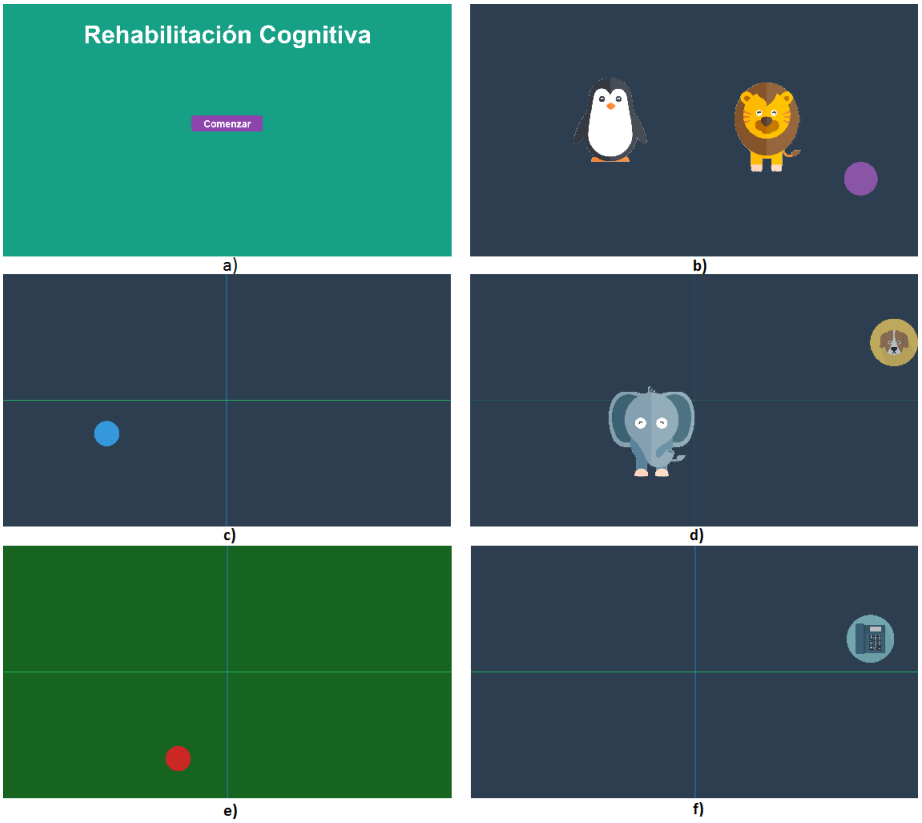

Figura 1. Capturas de pantalla de la herramienta en las cuales es posible apreciar las diferentes configuraciones posibles.

y se evita la complejidad de capacitar al paciente para que instale el *software* desarrollado.

3. Diseño del sistema

Teniendo en cuenta los objetivos planteados en el punto anterior, se propuso dividir el trabajo en cuatro módulos menores, detallados a continuación. El primero consiste en el desarrollo de una herramienta de rehabilitación cognitiva interactiva al estilo de un video-juego online. El segundo se enfoca en la implementación de una solución para la detección de la mirada a través de la estimación de la posición de la cabeza. El objetivo de este módulo es brindar información a los terapeutas sobre el cuadrante hacia donde está mirando el paciente, y en base a todos estos datos analizar la evolución del mismo durante el tratamiento. El tercer módulo comprende el desarrollo de la interfaz de configuración web para los ejercicios. Por último, el cuarto módulo incorpora un servicio web que

almacena y retorna los datos y configuraciones de los usuarios a la herramienta. Todos los módulos de *software* correspondientes interaccionan entre sí, enviando y recibiendo información según lo solicitan. En la Fig. 2 se muestra un diagrama de los mismos y la forma de intercambiar datos.

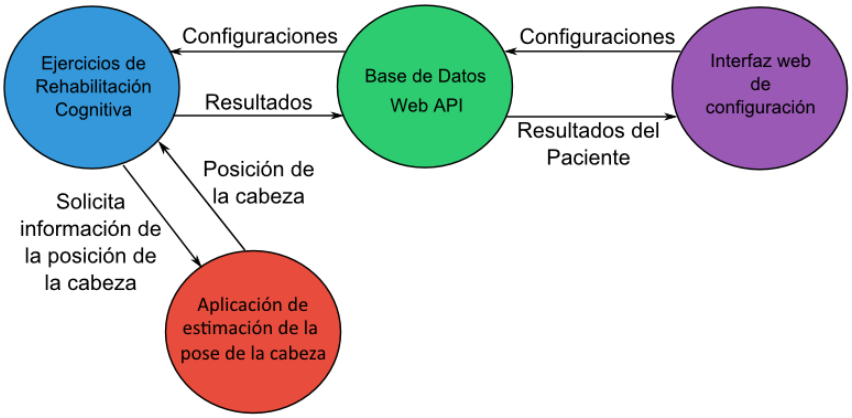

Figura 2. Diagrama que muestra las diferentes secciones y su comunicación.

4. Aplicación de estimación de la mirada a través de la posición de la cabeza

Considerando las características del ejercicio descripto en la sección 2, resulta de gran interés para los profesionales médicos determinar si el paciente está mirando el cuadrante correcto durante el tiempo en que aparece la figura. Por lo tanto, con respecto a estos requerimientos, la exactitud de la posición de la mirada no es tan crítica, lo que significa que no es necesario emplear un método de alta precisión para estimar la mirada del paciente sino que, debido al error tolerado por el problema, la misma puede ser inferida a partir de la posición de la cabeza [9].

Se decidió entonces, para este trabajo desarrollar una aplicación propia para estimar la posición de la cabeza. La idea fue intentar lograr una estimación de la pose de la cabeza rápida, eficiente y robusta, que soporte oclusiones, sea fácilmente integrable a la aplicación y que sea configurable en todos sus parámetros. Por ejemplo que sea posible incorporar optimizaciones en el reconocimiento de rostros, etc. Por otro lado, a nivel académico y pedagógico por lo general se obtiene mas conocimiento construyendo una herramienta ad-hoc que incorpore algunas funcionalidades desde bibliotecas de código abierto.

Como el objetivo es que la herramienta sea de amplio acceso al público, se decidió utilizar una cámara web de bajo costo y gran disponibilidad. Esto agregó un factor limitante ya que al utilizar una cámara de gama baja se reduce la calidad de la imagen capturada por la misma. Hay que tener en cuenta que por lo general,

la mayoría de los sistemas de oculografía con video requieren cámaras de alta resolución.
El método en que se basó para la realización de este trabajo fue el de Matsumoto *et al.* [10], un sistema en el cuál una cámara monocular es usada para calcular y estimar la posición 3D de la cabeza y los globos oculares, pero combinándolo con la propuesta de emplear una cámara de baja resolución. Al utilizar imágenes de menor resolución el contorno de los ojos, el iris y la pupila se difuminan, provocando esto que sea más difícil poder segmentar cada uno con precisión [11]. Matsumoto aplica un modelo 3D flexible. Para esta tesis se decidió utilizar un modelo 3D rígido ya que un modelo flexible añade la complejidad extra de tener que deformar la malla para representar los gestos de la cara. La implementación de un modelo flexible puede ser considerada adecuada si el objetivo de la detección es su utilización en animaciones [12,13], pero en el caso de estimación de la mirada para el problema de rehabilitación cognitiva carece de sentido ya que la representación de los gestos no es buscada en este problema.

El método detallado a continuación esta conformado por las siguientes subetapas:

1. Obtención de *frames* y detección del conjunto de puntos faciales.
2. Conversión del conjunto de puntos 2D a 3D.
3. Estimación de la posición de la cabeza.
4. Archivo de configuración del modelo y de los puntos.

4.1. Elección del entorno de desarrollo

Al trabajar con grandes volúmenes de datos (en este caso imágenes) uno de los requisitos para la estimación de la posición de la cabeza es que la misma se calcule rápidamente. Es por esto que se prefirió el uso de C++ por sobre otros, ya que es un lenguaje de programación de bajo nivel con el que es posible lograr aplicaciones eficientes y además soporta el paradigma de Orientación a Objetos. Para implementar los algoritmos de estimación de la posición de la cabeza se utilizó la biblioteca Open Source Computer Visión, (OpenCV) ya que está optimizada para procesamiento de imágenes [14]. Como entorno de desarrollo integrado (IDE) fue elegido Microsoft® Visual Studio® 2013 ya que facilita la integración con OpenCV sin necesidad de recompilar la biblioteca.

4.2. Obtención de *frames* y detección del conjunto de puntos faciales

Considerando que OpenCV provee una API para capturar video desde cámaras o archivos, esto desliga de las responsabilidades de tener que conocer el hardware o entrar en contacto con el driver de la cámara, lo cual permite que la aplicación pueda ser utilizada en cualquier computadora con diferentes cámaras.
A partir de un frame se detecta el rostro del usuario utilizando la función HaarCascade de biblioteca OpenCV que implementa el *framework* Viola-Jones

[15]. Este algoritmo devuelve un rectángulo donde esta contenida la cara y será la entrada de para una biblioteca de reconocimiento de *facial landmarks* [16] o puntos característicos del rostro.
Los *facial landmarks* se definen como un subconjunto de puntos que son utilizados en problemas como reconocimiento de rostros, detección de mirada, seguimiento facial, reconocimiento de expresiones y gestos, etc.
Luego de realizar un relevamiento de las bibliotecas disponibles, se seleccionó Dlib para lograr la detección de *landmarks* (sección F del apéndice). La ventaja de esta biblioteca frente a otras analizadas es que a juzgar por las pruebas realizadas en este trabajo, resultó ser estable y responder muy bien frente a las oclusiones, estimando con gran aproximación donde deberían estar los puntos no visibles. Por todas estas razones fue elegida para desarrollar la aplicación de estimación de la mirada.
Del total de los 68 puntos resultantes de aplicar Dlib, solo se seleccionaron 8 que permanecen fijos sin importar la expresión del rostro. Dos puntos indican el borde de los ojos y la boca, el extremo y el intermedio de la nariz. En la siguiente etapa se explicará la razón de seleccionar estos puntos.

4.3. Conversión del conjunto de puntos 2D a 3D

En esta etapa, se realiza un mapeo del conjunto de puntos obtenidos en la etapa anterior con un conjunto de 8 puntos seleccionados sobre una malla 3D de un modelo que representa una cabeza humana (ver Fig. 3). Este mapeo entre los *facial landmarks* 2D detectados y los puntos 3D de la malla se realiza mediante el algoritmo *solve PnP* [17] de OpenCV.
Este algoritmo se basa en el concepto conocido como modelo de cámara estenopeica (*pinhole-camera*). En este modelo, una escena se forma mediante la proyección de puntos en tres dimensiones sobre el plano de la imagen. Esta proyección se realiza utilizando una transformación de perspectiva. La imagen se forma como resultado de aplicar la expresión (1):

$$m' = A[R|t]M' \tag{1}$$

Donde A es la matriz de parámetros intrínsecos formada por las distancias focales, las cuales no dependen de la escena y por lo tanto puede ser re-utilizada siempre que no cambie la cámara y la resolución. [R/t] es la matriz de rotaciones y traslaciones de la cámara sobre una escena fija. La misma fórmula puede ser expresada de forma más completa según la expresión (2):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

Dónde:

- (X, Y, Z) son las coordenadas de un punto 3D

- (u, v) son las coordenadas de un punto 2D en píxeles
- (c_x, c_y) es el punto que es usado como centro de la imagen
- f_x, f_y son las distancias focales en píxeles

Dados los puntos 2D y sus correspondientes en el espacio 3D, el algoritmo *solve PnP* calcula la matriz de rotación y el vector de traslación descriptos anteriormente. Con esta información se debe armar la matriz $[R|t]$ que luego se multiplica por la matriz de parámetros intrínsecos obteniendo, de esta forma, $A[R|t]$. Por último se multiplican todos los puntos de la malla 3D por esta matriz, quedando la malla rotada y trasladada en el espacio. Este algoritmo se debe aplicar por cada *frame* en el que se reconocen *landmarks*.
Como resultado de esta etapa, cuando los puntos 2D varían de posición, es decir, si el usuario mueve la cabeza, la malla imita el movimiento. En la Fig. 3 se observa una captura de una posición específica.
En pruebas preliminares se observó que si el usuario gesticula o habla durante los ejercicios, muchos *landmarks* varían de posición en un instante, afectando la correcta ubicación de la malla en el espacio. La selección de los 8 puntos elegidos sobre la malla se realizó de forma tal de minimizar la variación de la posición de los *landmarks* frente a gestos o expresiones.

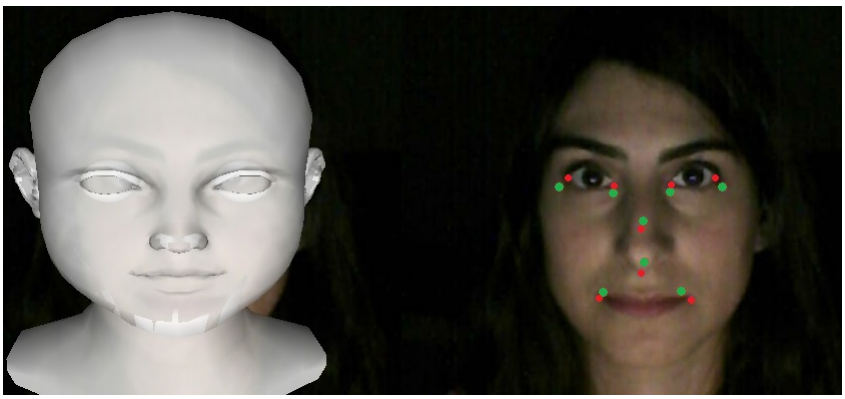

Figura 3. Representación de la estimación de los puntos 3D de la malla (puntos verdes, izquierda) correspondiente a cada *facial-landmark* (puntos rojos, derecha).

4.4. Estimación de la pose de la cabeza

Con el objetivo de estimar la pose de la cabeza a partir de los puntos mapeados en la sección anterior, se empleó el siguiente mecanismo: se toma un punto p situado en el centro de un segmento que conecta ambos ojos de la malla 3D. Luego se prolonga un segmento de distancia l desde este punto hacia el frente sobre el eje de profundidad. La prolongación mencionada constituye un segmento

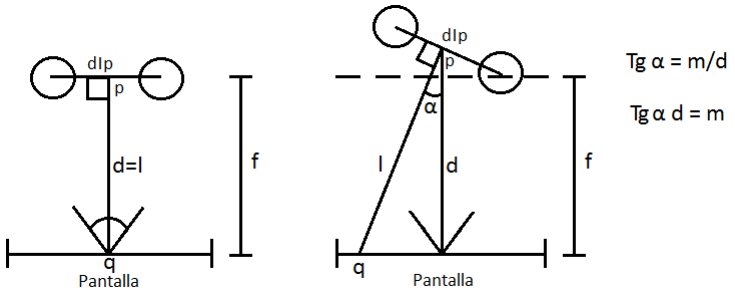

Figura 4. Representación de los ojos de forma cenital. El segmento inferior representa la pantalla, d representa la distancia a la que esta la persona de la cámara, dip es la distancia interpupilar, f es la distancia focal de la cámara, l la distancia que hay que calcular para determinar donde se encuentra q.

formado por el punto seleccionado entre medio de los ojos y el extremo de dicha prolongación, denominado q. Se considera para este trabajo que el usuario está ubicado a una distancia focal entre 50 y 70 cm. Entonces, se calcula l en base a esa distancia y el ángulo de la matriz de rotación (Esto se observa en la Fig. 4). El punto q se calcula como la intersección de entre el segmento mencionado anteriormente y el plano correspondiente a la pantalla. Luego este punto en el espacio 3D (x,y,z) es mapeado al espacio de la cámara 2D (u,v) mediante la fórmula $u = x/z$ y $v = y/z$.
Con este punto 2D, se puede tener una aproximación de la posición en la pantalla de acuerdo a las proporciones de la misma. Este método de aproximación fue basado en Valenti et al. [18].

4.5. Archivo de configuración del modelo y de los puntos

Para el almacenamiento de los datos se prefirió un archivo YAML (ver apéndice sección C). Primero se carga el nombre del archivo de la malla 3D que se quiere utilizar, esto brinda la posibilidad de usar cualquier malla tridimensional que se desee. Luego se almacenan la cantidad de puntos y sus coordenadas tridimensionales en el modelo. Las mismas se corresponderán con los *facial landmarks* detectados en el rostro del usuario. A continuación se guarda el punto que sirve para estimar la mirada. Por último se indican cuales son los *facial landmarks* elegidos. El orden de aparición indica cómo se corresponden los puntos de la malla previamente elegidos.

5. Interfaz de configuración de ejercicio

Mediante la interfaz de configuración de ejercicio, el médico es capaz de configurar cada detalle de los ejercicios de acuerdo a la problemática del paciente y su evolución en el tratamiento. La interfaz de configuración de ejercicio se

realizó empleando las tecnologías HTML5, CSS3, Javascript y Bootstrap. Se pensó un diseño simple, intuitivo y moderno.
Para el envío de las configuraciones se utilizó el formato JSON. Como resultado se traducen las configuraciones a un texto plano con un formato determinado que luego es interpretado tanto por la herramienta como por el servicio web. Todas estas tecnologías están detalladas en las secciones C y D del apéndice.

6. Base de datos y servicio web

Como los ejercicios son implementados mediante una aplicación web, las configuraciones particulares de cada paciente deben ser almacenadas en forma *online* de tal manera de garantizar su disponibilidad en todo momento. Se utilizó una base de datos para almacenar tanto la información de las configuraciones como los resultados de la rehabilitación con los ejercicios particulares de cada paciente. La conexión entre la base de datos, la herramienta y la interfaz web se realizó mediante un *web service* PHP (ver apéndice sección E). Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, puedan utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Como se mencionó anteriormente, el estándar adoptado en este proyecto fue JSON.

7. Resultados

En esta sección se presentan los resultados obtenidos, tanto para la herramienta de rehabilitación cognitiva, como así también para la implementación de las técnicas de seguimiento de mirada y la integración de ambas partes.

7.1. Resultados de la herramienta de rehabilitación cognitiva

Se desarrolló una plataforma online sobre la cual se implementó una herramienta de rehabilitación cognitiva que cumple con lo requerido por los profesionales del servicio de neurología clínica del Hospital Italiano.
Se han utilizado las tecnologías más actualizadas que se encuentran en el estado del arte del desarrollo de videojuegos de forma tal de asegurar una usabilidad adecuada.
La plataforma se desarrolló como una aplicación web, lo permite el acceso a los pacientes desde sus domicilios sin necesidad de descargar e instalar programas locales.
Otro beneficio de las tecnologías aplicadas es la independencia de la resolución del dispositivo, lo cual permite que la plataforma pueda ser utilizada desde computadoras personales, así como desde diferentes dispositivos móviles.

La herramienta permite que los médicos puedan configurar ejercicios para sus pacientes, estableciendo los parámetros de acuerdo a los requisitos solicitados.
Las métricas de la herramienta, tal como cantidad de clics realizados, cantidad de aciertos en los ejercicios, medición de tiempos, entre otros, se envían a través de un servicio web para ser almacenados en una base de datos para su uso y análisis por parte de los médicos.
Desde el punto de vista del desarrollo se ha testeado la plataforma sobre diferentes dispositivos tales como computadoras personales, tablets y teléfonos móviles para asegurar su funcionamiento de forma independiente del dispositivo y de la resolución.

7.2. Resultados de la implementación del seguimiento de mirada

Esta sección describe los experimentos realizados para evaluar el sistema de seguimiento de mirada. Para el relevamiento de resultados se utilizó la cámara web Genius Ilook 300, una cámara de gama media-baja y resolución de 1.3 mega pixeles en video. La resolución de las imágenes capturadas es de 640x480 píxeles.
Como el objetivo acordado con los médicos es el seguimiento de cuadrantes sobre la pantalla, se numeraron los mismos tomando como referencia la herramienta de rehabilitación cognitiva. Ver Fig. 5
La robustez del algoritmo de seguimiento de mirada se verificó realizando la prueba que se describe a continuación con diferentes escenarios de condiciones de iluminación (luz natural de frente, luz natural indirecta, luz artificial fluorescente e incandescente y condiciones de poca iluminación).
Con un usuario observando las esquinas de la pantalla, se almacenó la posición de la mirada detectada cada 100 milisegundos.
El algoritmo puede seguir la pose de la mirada (gaze tracking) aproximadamente el 90 % del tiempo en condiciones de luz aceptables, en estos casos la mirada estimada coincide con el cuadrante donde aparece la figura el 96 % de las veces.
Los casos observados en donde la herramienta falla al detectar el cuadrante correcto se relacionan con casos de oclusiones parciales de la cara y con condiciones posturales no tenidas en cuenta en la calibración.

7.3. Integración de los datos de la herramienta de rehabilitación con el seguimiento de mirada

Para la evaluación de la integración de los resultados de la herramienta de rehabilitación con el algoritmo de seguimiento de mirada, se definió una sesión donde el usuario utiliza el ejercicio definido por el médico en el módulo de configuración, mientras el módulo de seguimiento de mirada se encuentra activo.
Luego, se correlacionaron las ubicaciones de las figuras junto con los datos de la estimación del cuadrante.
Se realizó un análisis temporal de ambas series de datos cuantificando el error con diferentes técnicas.
Como primera prueba se intentó seguir con la mirada la ubicación de figuras que cambian de cuadrante cada 3 segundos. Si bien el algoritmo de seguimiento de

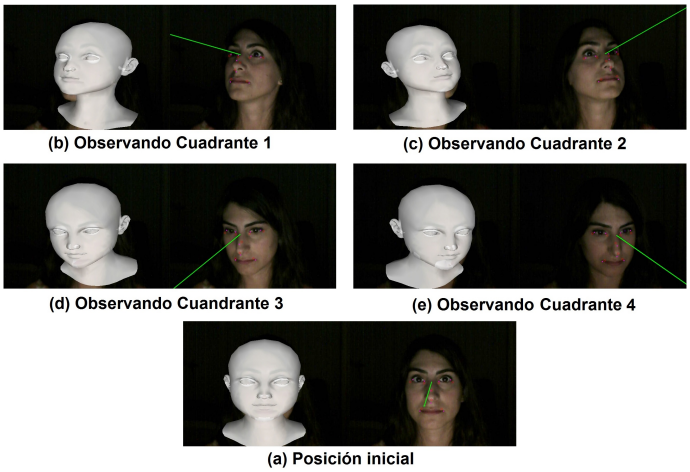

Figura 5. Resultados.

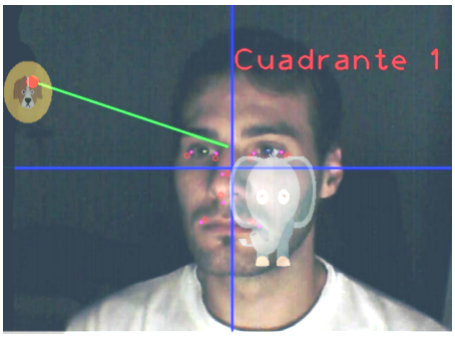

Figura 6. Representación visual de la integración de la herramienta de rehabilitación con la estimación de la mirada.

mirada es robusto y responde rápidamente, la latencia del algoritmo es del orden de la demora del usuario en ubicar la figura en pantalla, por lo que la efectividad del algoritmo no es lo suficientemente buena, llegando a ser del 60 %.
La segunda prueba se realizó con un ejercicio en el que las figuras permanecen durante 7 segundos en cada cuadrante. En este caso la efectividad resulto ser del 95 %. La Fig. 6 es la representación de una captura temporal de un usuario observando una figura mientras el módulo de seguimiento de mirada detecta el cuadrante hacia donde está mirando.

En la Fig. 7 se pueden observar los resultados de la integración de la detección de la mirada con la herramienta de rehabililtación cognitiva. Es posible detectar de esta manera un retardo entre el instante en el que aparece la figura en un determinado cuadrante y el instante en el que el paciente ubica la figura con la mirada. Esto está estrechamente relacionado con la velocidad de procesamiento

sensorial o tiempo de reacción. También es posible detectar un retardo entre el instante en que el paciente ubica la figura con la mirada y el instante en el que realiza un clic exitoso, lo cual se relaciona con la velocidad de procesamiento motor. Estos retardos son útiles en medicina para diagnosticar y planear la rehabilitación de entidades clínicas que comprometen funciones cerebrales relacionadas con la coordinación de función psicomotora [19].

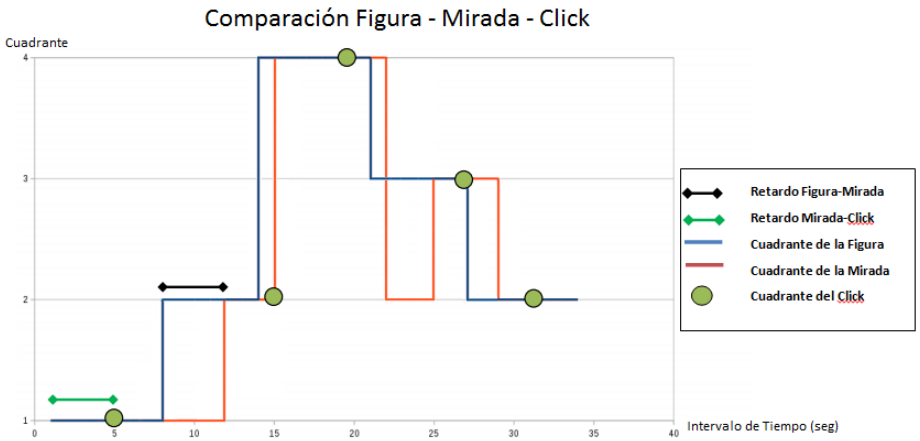

Figura 7. Comparación en el tiempo de la figura, la mirada del usuario y el cuadrante donde se realiza el click.

8. Conclusión

En un tiempo de desarrollo total de 3 meses se diseño y construyó una herramienta on-line independiente del Sistema Operativo, que puede correr en múltiples dispositivos sin la necesidad de que los usuarios posean conocimientos avanzados de informática. Como los usuarios de esta herramienta son pacientes que presentan diferentes discapacidades tanto motrices como cognitivas, en muchos casos su independencia es reducida, generalmente dependiendo de un familiar que los ayude a inicializar el software de rehabilitación.
Como complemento a esta herramienta, se codificó e integró una aplicación de procesamiento de imágenes que se ejecuta en una plataforma de escritorio y que, a partir de una cámara monocular de baja resolución, estima y registra la posición de la cabeza del usuario proporcionando información sobre la mirada del paciente, de forma tal de obtener datos que permitan correlacionar tal zona con la ubicación del cursor durante los ejercicios. Como el objetivo es que la herramienta sea de amplio acceso al público, se decidió utilizar una cámara web de bajo costo y gran disponibilidad. Esto agregó un factor limitante ya que al utilizar una cámara de gama baja se reduce la calidad de la imagen capturada por la misma.

Actualmente se están comenzando a reaizar pruebas con pacientes con afecciones neurológicas en el servicio de Neurología del Hospital Italiano.
Como trabajo futuro se planifica, por una parte implementar nuevos ejercicios de rehabilitación cognitiva de acuerdo al diseño realizado por los profesionales del servicio de neurología del Hospital Italiano y por otra parte, implementar otros algoritmos de seguimiento de mirada que mejoren la precisión de la detección e integrar la aplicación de escritorio por medio de un servicio web con la herramienta on-line.

2

Apéndice: Relevamiento de técnologías de desarrollo

A. Introducción

Debido a las características visuales e interactivas de la herramienta desarrollada, se realizó un relevamiento de los lenguajes, ambientes de desarrollo y *frameworks* disponibles para el desarrollo de videojuegos. Los mas relevantes se enumeran a continuación.

B. Tecnologías de desarrollo de herramientas interactivas

A continuación se describen brevemente las diferentes tecnologías analizadas para desarrollar el *software* de rehabilitación cognitiva.

Unity es un motor de videojuego multiplataforma. Está disponible como plataforma de desarrollo para Microsoft Windows, OS X y Linux. Gracias al *plugin* web de Unity, también se pueden desarrollar videojuegos de navegador para Windows y Mac.

Phaser.io es un *framework* Javascript para el diseño de juegos. La ventaja de este lenguaje es que no requiere ningún *plugin* adicional, ya que todos los navegadores web actuales poseen un intérprete javascript.

Haxe es un lenguaje de programación multiplataforma de alto nivel y de código abierto, que puede producir programas y código fuente para distintas plataformas desde un único código base. El código escrito en Haxe puede ser compilado en Adobe Flash, Javascript, C++, Java, C# y en aplicaciones de lado del servidor como PHP, Apache CGI y Node.js. Haxe es un lenguaje compilado de propósito general con programación orientada a objetos, excepciones, e inferencia de tipos con parámetros de clase. Clases genéricas, reflectividad, iteradores, y programación funcional están incorporados en la funcionalidad del lenguaje y librerías. A diferencia de otros lenguajes de programación, Haxe contiene un sistema de tipos fuerte pero a su vez también dinámico. Es decir que el compilador comprueba tipos implícitos y da errores de compilación, pero el lenguaje también permite al programador omitir, en ciertos casos, la comprobación de tipos y basarse en el manejo dinámico de tipos de la plataforma de destino [20].
OpenFL es un *framework* libre y gratuito para la creación de *software* y video

juegos que se complementa con haxe agregando soporte para interfaz gráfica multiplataforma.

B.1. Decisiones consideradas para el desarrollo

Para la implementación de estos ejercicios se analizaron un conjunto de tecnologías entre ellas las más destacadas son **Unity 3D engine**, **Phaser.io** y **Haxe**. **Adobe Flash Player** fue descartado ya que la mayoría de los navegadores actuales lo bloquean por defecto debido a la presencia de vulnerabilidades. Tampoco tiene soporte oficial en las nuevas versiones de Android e IOS.
La ejecución web de **Unity** es a través de un *plugin*. El problema de este *plugin* es que se requiere su instalación en el navegador de la computadora lo cual añade una complejidad a los usuarios a la hora de utilizar la aplicación. Debido a esto fue descartado como una opción para este trabajo.
Con **Phaser.io** se realizó un prototipo de los ejercicios pero debido a la imposibilidad de crear clases con Javascript, el código se tornaba demasiado extenso y procedural a medida que aumentaba la complejidad del proyecto, dificultando así su legibilidad y extensibilidad.
Después de atravesar los problemas presentados anteriormente se tomó la decisión de implementar el *software* en el lenguaje **Haxe**. Al ser la sintaxis de **Haxe** inspirada en Java, el tiempo de aprendizaje del lenguaje fue muy corto. Además, las aplicaciones funcionan muy rápido debido a la compilación a código nativo. Por estos motivos fue elegido para desarrollar la aplicación web.

C. Lenguajes para intercambio de datos

A continuación se describen brevemente los diferentes lenguajes analizados para desarrollar el intercambio de datos entre las diferentes aplicaciones que conforman este proyecto.

JSON (JavaScript Object Notation), un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

YAML es un formato de serialización de datos legible por humanos. Fue creado bajo la ideología de que todos los datos pueden ser representados adecuadamente como combinaciones de listas, hashes (mapeos) y datos escalares (valores simples). La sintaxis es relativamente sencilla y fue diseñada teniendo en cuenta que fuera legible pero que a la vez fuese fácilmente mapeable a los tipos de datos más comunes en la mayoría de los lenguajes de alto nivel. Además, YAML utiliza una notación basada en la indentación. Es importante destacar que la biblioteca OpenCV incorpora una función para la lectura y escritura de archivos en formato YAML.

D. Tecnologías para el desarrollo de sitios web

A continuación se describen brevemente las diferentes tecnologías analizadas para desarrollar la interfaz de configuración de la herramienta de rehabilitación cognitiva.

HTML5 siglas de HyperText Markup Language 5, es un lenguaje de marcado para la elaboración de páginas web HTML5 está basado en HTML, considerado el lenguaje mas importante que ha intervenido en la aparición, desarrollo y expansión de la World Wide Web. Actualmente HTML5 es un estándar ampliamente adoptado para la visualización de páginas web y ha sido adoptado por la mayoría de los navegadores actuales.

CSS o Hoja de estilo en cascada es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser definida en un documento separado o en el mismo documento HTML.

Bootstrap es un framework de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML5 y CSS3.

E. Tecnologías para el desarrollo de *web services*

Debido a que la herramienta desarrollada en este trabajo debe conectarse con el repositorio de datos de la Historia Clínica Electrónica (HCE) del Hospital Italiano, por restricciones prácticas, se ha optado por hacerlo vía servicios web. A continuación se enumeran las tecnologías consideradas para la implementación de los *web services* para la comunicación con la HCE.

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

F. Tecnologías para la detección de *facial landmarks*

Dlib es una biblioteca *OpenSource* escrita en C++. Su diseño esta influenciado por la ingeniería de *software* basada en componentes. Esto significa que es, en general, una colección de componentes de *software* independientes. En particular,

contiene componentes para *networking*, *threads*, interfaces gráficas de usuario, estructuras de datos, álgebra lineal, *machine learning*, procesamiento de imágenes, minería de datos, optimización numérica y redes bayesianas entre otras funciones. Dlib incluye una implementación del algoritmo de *face alignment* [21]. Este algoritmo comienza con la indexación de las intensidades de los píxeles en relación con la estimación actual de la forma de la cara. Las características extraídas en el vector de representación de la imagen de un rostro pueden variar en gran medida debido a la deformación de la misma y a factores como cambios en las condiciones de iluminación. Esto hace difícil lograr una estimación precisa del rostro usando estas características. El dilema que se presenta es que se necesitan características confiables para predecir de manera exacta la forma del rostro, y por otra parte es necesario una estimación precisa de la forma para extraer características confiables. Trabajos previos como: [22], [23], [24] y [25] usan un enfoque iterativo (cascada) para tratar con este problema. En lugar de estimar la forma del rostro usando las características extraídas en el sistema de coordenadas global de la imagen, la imagen es transformada a un sistema de coordenadas normalizado basado en una estimación actual de la cara, y después las características son extraídas para predecir un vector actualizado para los parámetros de la figura. Este proceso suele repetirse varias veces hasta converger.
Lo siguiente es cómo combatir la dificultad del problema de inferencia. Al momento de la prueba, un algoritmo de alineamiento estima la forma que mejor coincide con los datos de la imagen y la forma del modelo. Esta técnica propone utilizar una cascada de vectores de regresión. De esta manera, si S(t) es un conjunto de landmarks de la imagen I, siendo S(t)= (l1,l2,..,ln), donde li es una coordenada (x,y), se tiene un conjunto de vectores de regresión r1,r2,...rn que permiten predecir el conjunto de landmarks de la forma: S(t+1) = S(t) + r(t)(I,S(t)).
Debido a que el regresor r(t) predice el nuevo S(t+1) de forma relativa al conjunto de landmarks actual, esto introduce una cierta invariancia geométrica. Esta invariancia permite lograr una precisión mayor a la lograda utilizando sólo características globales de la imagen. Cada r(t) es entrenado utilizando un método de descenso por gradiente que minimiza el error cuadrático medio.

Referencias

1. Y Ginarte-Arias. Rehabilitación cognitiva. aspectos teóricos y metodológicos. *Revista de Neurología*, 34(9):870–876, 2002.
2. Elízabeth Fernández, María Luisa Bringas, Sonia Salazar, Daymí Rodríguez, María Eugenia García, and Maydané Torres. Clinical impact of rehacom software for cognitive rehabilitation of patients with acquired brain injury. *MEDICC review*, 14(4):32–35, 2012.
3. William Fals-Stewart and Wendy KK Lam. Computer-assisted cognitive rehabilitation for the treatment of patients with substance use disorders: A randomized clinical trial. *Experimental and clinical psychopharmacology*, 18(1):87, 2010.
4. Simon Larose, Sylvain Gagnon, Christiane Ferland, and Michel Pépin. Psychology of computers: Xiv. cognitive rehabilitation through computer games. *Perceptual and Motor Skills*, 69(3):851–858, 1989.

5. Amit Lampit, Claus Ebster, and Michael Valenzuela. Multi-domain computerized cognitive training program improves performance of bookkeeping tasks: a matched-sampling active-controlled trial. *Psychological perspectives on expertise*, page 194, 2015.
6. Martin L Rohling, Mark E Faust, Brenda Beverly, and George Demakis. Effectiveness of cognitive rehabilitation following acquired brain injury: a meta-analytic re-examination of cicerone et al.'s (2000, 2005) systematic reviews. *Neuropsychology*, 23(1):20, 2009.
7. Amit Lampit, Harry Hallock, and Michael Valenzuela. Computerized cognitive training in cognitively healthy older adults: a systematic review and meta-analysis of effect modifiers. *PLoS Med*, 11(11):e1001756, 2014.
8. Alexandra M Kueider, Jeanine M Parisi, Alden L Gross, and George W Rebok. Computerized cognitive training with older adults: a systematic review. *PloS one*, 7(7):e40588, 2012.
9. Volker Krüger and Gerald Sommer. Gabor wavelet networks for efficient head pose estimation. *Image and vision computing*, 20(9):665–672, 2002.
10. Yoshio Matsumoto, Naoki Sasao, Tsuyoshi Suenaga, and Tsukasa Ogasawara. 3d model-based 6-dof head tracking by a single camera for human-robot interaction. pages 3194–3199, 2009.
11. Takashi Fukuda, Kosuke Morimoto, and Hayato Yamana. Model-based eye-tracking method for low-resolution eye-images. 2011.
12. Chen Cao, Qiming Hou, and Kun Zhou. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Transactions on graphics (TOG)*, 33(4):43, 2014.
13. Chen Cao, Derek Bradley, Kun Zhou, and Thabo Beeler. Real-time high-fidelity facial performance capture. *ACM Transactions on Graphics (TOG)*, 34(4):46, 2015.
14. Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. .º'Reilly Media, Inc.", 2008.
15. Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.
16. Martin Köstinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. pages 2144–2151, 2011.
17. Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166, 2009.
18. Roberto Valenti, Nicu Sebe, and Theo Gevers. Combining head pose and eye location information for gaze estimation. *Image Processing, IEEE Transactions on*, 21(2):802–815, 2012.
19. David De Noreña, Marcos Ríos-Lago, Igor Bombín-González, Ignacio Sánchez-Cubillo, Alberto García-Molina, and Javier Tirapu-Ustárroz. Efectividad de la rehabilitación neuropsicológica en el daño cerebral adquirido (i): atención, velocidad de procesamiento, memoria y lenguaje. *Rev Neurol*, 51(11):687–98, 2010.
20. Franco Ponticelli and Lee McColl-Sylveste. Professional haxe and neko. 2008.
21. Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. pages 1867–1874, 2014.
22. Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

23. Gareth J Edwards, Timothy F Cootes, and Christopher J Taylor. Advances in active appearance models. 1:137–142, 1999.
24. David Cristinacce and Timothy F Cootes. Boosted regression active shape models. pages 1–10, 2007.
25. Piotr Dollár, Peter Welinder, and Pietro Perona. Cascaded pose regression. pages 1078–1085, 2010.