cet
41:(2020)
ISSN:1668-9178

Medición del diámetro de la pupila ocular con libertad de movimiento y flexibilidad técnica

Juan Ignacio Contino y Andrés Martín

Facultad de Ciencias Exactas y Tecnología, Universidad Nacional de Tucumán, Tucumán, Argentina.

Resumen

Los estudios de la visión humana requieren con frecuencia medir el tamaño de la pupila de un sujeto. Esta tarea es compleja por un número de razones, y es un tema de especial interés en los diseños experimentales. La medición se realiza tradicionalmente con dispositivos que detectan las pupilas y siguen su movimiento, llamados *eyetrackers*. Los *eyetrackers* requieren que las medidas se tomen de tal manera que el sujeto mantenga una posición corporal determinada o de movilidad restringida, limitando las tareas posibles en un experimento. Para solucionar estos inconvenientes se puede optar por un *eyetracker* que consta de una cámara fijada a un marco de anteojos (y permite el movimiento libre del sujeto), con facilidades de conexión. Este informe técnico documenta el trabajo hecho en la puesta a punto de un sistema que toma medidas del diámetro de la pupila de un sujeto mientras éste realiza una tarea visual. Se diseña también un protocolo para realizar estas mediciones. Para controlar el dispositivo se diseña un sistema instalable en dispositivos pequeños portátiles o *PCs* (placas capturadoras). El sistema final es capaz de adquirir mediciones de pupila que permiten libertad de movimiento al sujeto y portabilidad de todo el sistema a nivel *software* y *hardware*.

Palabras clave: *eyetracker*, pupila.

Measurement of the ocular pupil diameter with freedom of movement and technical flexibility

Abstract

Studies of human vision frequently require measuring the size of a person's pupil. This is a complex task, for a number of reasons, and is a topic of particular interest in experimental designs.The measurement is traditionally done with devices that detect the pupils and follow their movement, called eyetrackers. Eyetrackers require that measurements be taken in such a way that the person maintain a certain body position or restricted mobility, limiting the possible tasks in an experiment. An eyetracker consisting of a camera fixed to an eyeglass frame (and that allows free movement of the person), with connection facilities, can be used to overcome these drawbacks. The present technical report documents the work done in setting up a system that takes measurements of a person's pupil diameter while performing a visual task. A protocol is also designed to perform these measurements. To control the device, a system that can be installed on small portable devices or PCs is designed. The final system is capable of acquiring pupil measurements that allow freedom of movement to the person and the portability of the entire system at the software and hardware level.

Keywords: *eyetracker, pupil.*

Introducción

La técnica de la medición de pupila es hoy ampliamente utilizada, se encuentra desarrollada, y se realiza con dispositivos que detectan las pupilas y siguen su movimiento llamados *eyetrackers*, denominación que se utilizará en adelante. Sin embargo, esta tarea tiene algunas dificultades asociadas, como el traslado del dispositivo, la complejidad de la conexión de este equipo con *PCs* (placas capturadoras) y la movilidad del sujeto en el experimento. Los *eyetrackers* se pueden subdividir entre los que necesitan que los sujetos mantengan una posición determinada y fija de su cabeza durante el desarrollo del experimento y los que dan ciertos grados de libertad en su movimiento. Siempre que el experimento lo permita, es deseable que el sujeto pueda moverse. Una tarea con un diseño que requiera mantener la posición de la cabeza de un sujeto fija provoca agotamiento cuando estos experimentos son prolongados. Esto lleva a pérdidas de atención, entre otros efectos, interfiriendo de manera indeseada en el rendimiento de un sujeto en la tarea.

Las mediciones de pupila que se hacen tradicionalmente en el Departamento de Luminotecnia, Luz y Visión de la Facultad de Ciencias Exactas y Tecnología (FACET), Universidad Nacional de Tucumán (UNT) se realizan con *eyetrackers* de ambos tipos, con dispositivos *Cambridge Research System Eyetracker Toolbox* y *Arrington Research ViewPoint EyeTracker*. El último permite realizar mediciones con libertad de movimiento, pero su conexión con una computadora requiere de la compra e instalación de placas adquisidoras de datos específicas y su *software* solo se encuentra disponible para un sistema operativo determinado, lo que dificulta su uso, y al final, su portabilidad. Para solucionar estos problemas se adquiere un nuevo *eyetracker* y se adapta un sistema de *software* y *hardware* para poner a punto un equipo que se utilizará en las futuras configuraciones experimentales psicofísicas. El sistema final necesita de un dispositivo de *hardware* donde implementar la solución que soporte el *software* del *eyetracker* comprado para manejarlo y ejecutar los experimentos.

Se plantea realizar una solución de *software* portable entre *hardware* de distintos formatos que facilita la movilidad de los experimentos a entornos de campo abierto. Para este informe, la implementación se realiza en *hardware* de formato PC. Sin embargo, la configuración final es instalable en otros dispositivos de manera nativa.

El *hardware* utilizado, la puesta a punto del *software* del *eyetracker* y la configuración del *software* adicional del sistema para diseñar estímulos psicofísicos son el aporte que se detalla y se documenta en el trabajo. También se diseña un protocolo adecuado para la utilización de la solución final.

Eyetracker

El fabricante del e*yetracker* es Pupil-Labs y su modelo es Pupil Core (**Kassner et al.** (2014)). Este dispone de conexión USB para su control y se distribuye con un *software* de *eyetracking* que es necesario configurar, calibrar y comunicar con otros programas que corren los experimentos para coordinar las mediciones en los momentos adecuados.

Fisonomía

Se conforma de un marco de anteojo al que se agregan módulos con funcionalidades específicas (ver figura 1).

El dispositivo del que se dispone incluye una cámara de alta frecuencia de captura de fotogramas para el ojo derecho y una cámara frontal. Las especificaciones de las cámaras (**Kassner et al.** (2014)), se observan en la tabla 1.

Fig. 1 *Pupil Eyetracker Core* - marco y cámaras. Imagen pública tomada de https://pupil-labs.com/products/core/

Tabla 1: Especificaciones de las cámaras

Cámara	Resoluciones	Campo de Visión
Cámara Frontal	30Hz @ 1080p 60Hz @ 720p 120Hz @ 480p	100° 60° 60°
Cámara Ojo Derecho	200Hz @ 192x192p	60°

Software del eyetracker

La *suite* de *software* de Pupil-Labs hace de nexo entre el *eyectracker* y el equipo donde se conecta. Se implementa en Python y en C, su código fuente está disponible y es multiplataforma (soportado en Windows y en GNU/Linux), (**Kassner et al.** (2014)). La *suite* consta de 3 programas:

1. *Pupil Capture*: captura video de las cámaras; implementa modelos: Ojo, Gaze; calibra modelos y genera datos (extensible mediante *plugins*).
2. *Pupil Player*: visualiza grabaciones y exporta datos.
3. *Pupil Service*: Interfaz IPC.

Medición

En el diseño de la solución se considera un caso de medición genérica en tareas controladas. De este modo, el sistema final se puede usar en una amplia gama de experimentos con requisitos muy distintos. Se concibe una medición con las siguientes características:

• Automática: debe realizarse de manera automática sobre uno de los ojos del sujeto.
• Controlable: debe poder darse inicio y final al proceso de medición desde un programa diseñado en los momentos requeridos.
• Frecuencia: deben ejecutarse con frecuencia de entre 20 y 30 mediciones por segundo.
• Estampa Temporal (*timestamp*)[s]: debe proveer la referencia temporal del momento al que se corresponde la medición.
• Dato [mm]: debe proveer claramente diámetro de la pupila, que es el parámetro buscado.
• Confiable: debe proveer un parámetro que determine la calidad de la medida.
•Presentación: debe proveer los datos en un archivo de campos separados por coma (CSV).

Un protocolo apropiado para ejecutar la medición puede ser el propuesto en el Anexo I.

Diseño del sistema de medición

El diseño del sistema final tiene componentes de *hardware* y de *software* y responde a las premisas:

• Debe ser implementable en el *hardware* disponible del laboratorio (Arquitectura x86_64).
• Debe ser portable a *hardware* más pequeño de arquitectura x86_64 (AtomicPi, mini PCs) y de arquitectura ARM (Odroid, Raspberry Pi).

Hardware

Para armar el sistema se dispone de una PC cuyo CPU tiene tecnología que data del año 2008 y una GPU del año 2009. Se consigue un uso equilibrado de este *hardware* con 4GB de memoria RAM Dual Channel DDR2. *Hardware* disponible:

- *Motherboard*: *ASRock* N68C-GS FX
- CPU: *AMD Phenom* 8650 Triple-Core
- GPU: *NVIDIA GeForce* 210
- Memoria: 2 x 2GB DDR2 1066 Hz *Dual Channel*

Software

Es necesario disponer de los programas pertenecientes a la *suite* de Pupil-Labs en el sistema final. Hay que remarcar que este *software* está pensado para ser usado en computadoras actuales y además está escrito en un lenguaje interpretado. Ambas cualidades suponen un consumo notable de recursos computacionales para un sistema moderno y es una de las cosas a tener en cuenta para el diseño del sistema final.

Realizar el trabajo con el *hardware* disponible es una

potencial limitación para la obtención de las mediciones con la frecuencia esperada. Esto también influye en otros parámetros, por ejemplo: resolución del video grabado, postprocesado de la imagen, uso de *plugins* en el *software* de control del *eyetracker*, etc.).

Según cómo se concibe la solución, el *software* de la instalación debe resolver los siguientes problemas en el siguiente orden de prioridades:

1. **La distancia generacional del *hardware***

La interfaz gráfica de usuario (GUI) del *software* del *eyetracker* está escrita en Python. Las funciones de los modelos del ojo que implementa la suite están escritas en C y hacen uso intensivo de librerías matemáticas (BLAS, OpenCV, OpenGL, Ceres Solver, etc), (**Kassner et al.** (2014)). Este problema es el más importante de todos, ya que determina si la solución puede implementarse o no. Hay sistemas operativos que se pueden utilizar en *hardware* antiguo, pero no soportan las librerías que implementan las funciones matemáticas de la *suite* de *Pupil* (incompatibles a nivel arquitectura), o no lo hacen con la performance necesaria para la tarea (problema generacional).

Para solucionar este inconveniente se elige un sistema operativo con *kernel Linux*. Este *kernel* soporta el *hardware* en toda la gama generacional del caso de uso, (***The Linux Kernel Development Community*** (2019)). Un sistema operativo basado en ese núcleo soporta también el *software* del *eyetracker* de Pupil-Labs. Sobre este *kernel* se puede construir un sistema operativo a medida con mucha flexibilidad, lo que permite completar el resto de la configuración. Se compila el *kernel* desde el código fuente para la arquitectura del procesador y el *hardware* presente para evitar gastos de recursos computacionales en funcionalidades que no se utilizan (el procedimiento está fuera del alcance de este informe). Se usa el código fuente correspondiente al día 15/09/2017. Se utiliza para el resto del sistema la distribución de paquetes de *Archlinux* (*Rolling Release*).

2. **La elección del lenguaje de programación para escribir los experimentos**

El lenguaje que se usa tradicionalmente para implementar los experimentos psicofísicos en el Departamento de Luminotecnia, Luz y Visión (FACET/UNT) es MATLAB. MATLAB no se puede usar en esta solución, ya que:

- Sus requisitos de *hardware* superan en demasía a las características de equipo del que se dispone.
- El valor económico de su licencia es elevado.
- No está disponible en la arquitectura ARM.

En su lugar se utiliza *Octave. Octave* es un lenguaje cuyo intérprete es casi completamente compatible con el intérprete de MATLAB. Este *software* tiene requerimientos mínimos de *hardware*. Además, cuenta con las siguientes ventajas respecto del *software* tradicional:

- Puede ser optimizado para la arquitectura compilando desde el código fuente.
- No necesita un GUI que consuma recursos.
- Sus dependencias son nativas del sistema operativo.
- Se desarrolla nativamente para el sistema operativo.

Se instala *Octave* 4.2.1, **(Eaton** (2018)).

3. **Los requisitos de las características de la medición**

Para obtener el grado de control y confiabilidad de la medición requerido en el sistema, es necesario programar también estas mediciones en lugar de dejar que el *software* del *eyetracker* haga este trabajo de manera independiente. La manera correcta de hacer esto es comandar las mediciones desde dentro del código del experimento escrito, sincronizándolas con los eventos apropiados. Para esto, la *suite* de Pupil-Labs provee un medio de comunicación TPC/IP mediante el cual el *software* del *eyetracker* queda esperando órdenes externas que se pueden estructurar en el código del programa.

Para solucionar esta comunicación se instala un complemento del lenguaje Octave que implementa la librería ZeroMQ, (**Pieter Hintjens** (2019)). Esta librería es portable y permite la comunicación entre procesos de un mismo equipo o de equipos de una red mediante el conjunto de protocolos TCP/IP. De esta manera, se puede controlar el algoritmo que maneja las cámaras del *eyetracker*, la calibración de los algoritmos Ojo y Gaze, etc., (**Kassner et al.** (2014)).

Esto le da al sistema la capacidad de realizar mediciones automáticas, controlables, confiables, de administrar su presentación, y de registrar el momento exacto en que se toman los datos.

4. **El requisito de portabilidad del sistema**

La portabilidad entre arquitecturas x86_64 y ARM queda solucionada con la elección realizada del kernel y la distribución de *software*, (***The Linux Kernel Development Community*** (2019)). El sistema operativo, la *suite* de Pupil-Labs y el lenguaje de programación soportan de manera nativa las arquitecturas de procesadores requeridas. Para portar el sistema solamente hace falta instalar la misma configuración de software detallada y eventualmente compilar algunas librerías necesarias en la arquitectura destino.

La portabilidad permite instalar el sistema en dispositivos pequeños alimentados con batería para poder hacer mediciones en entornos abiertos y sin instalación eléctrica.

Esta configuración permite programar completamente el *eyetracker* en el equipo donde se conecta y de manera remota, esto es: mediante una red cableada o inalámbrica.

Software adicional

Para realizar las mediciones es solamente necesario un sistema operativo de soporte con la configuración detallada anteriormente, el lenguaje para programación, y el *software* del *eyetracker*. Sin embargo, la solución se complementa con instalaciones accesorias que permiten la operación futura de personal no técnico del laboratorio, y la elaboración de estímulos visuales complejos para unificar la actividad experimental psicofísica en un solo equipo.

Se necesitan facilidades para montar memorias extraíbles, administrar archivos de manera gráfica, etc. Se instala un entorno gráfico muy básico para no consumir recursos que deben ser aprovechados en la aplicación para la que fue diseñado el sistema. Se opta por LXDE para este fin, con una configuración especial y el recorte de las funcionalidades que no se utilizan.

Para diseñar los estímulos visuales se instala además la *suite* de librerías Psychtoolbox 3.0.14 para *Octave*, (**Brainard** (1997)).

Presentación de los datos

El *software* de Pupil-Labs devuelve los siguientes recursos:

• Videos: Contienen la filmación asociada a la cámara frontal y a la del ojo.
• Archivos CSV: Cada línea de estos archivos está formada por una tupla compuesta por todos los valores devueltos por los dos algoritmos ejecutados en el *software* de Pupil-Labs. Entre estos valores están el diámetro de la pupila en milímetros (que es el dato que se busca), la confiabilidad del valor provisto por el algoritmo (en puntuación 0 a 1), datos correspondientes a las coordenadas esféricas de la mirada, y datos asociados a la temporalidad de las mediciones realizadas. Ver Anexo II.

Temporalidad de los datos

Como los algoritmos involucrados son de naturaleza iterativa, (**Kassner et al.** (2014)), no se puede definir una frecuencia regular de entrega de resultados. Por esta razón el algoritmo incluye en cada registro de cada archivo .CSV un primer campo del tipo índice que registra el momento en segundos en el que se tomó la medición y en el segundo campo se registra el número de captura de imagen (asociado a ambos videos) más cercano a esa medición. De esta manera, se puede comprobar que no todas las capturas de imágenes tienen la misma cantidad de mediciones asociadas, y que algunas carecen de medición. Ver Anexo II.

El campo índice temporal de medición es relativo a un momento seleccionado de manera arbitraria por el desarrollador del experimento, y se puede establecer en el programa del experimento a través de ZeroMQ. El índice temporal establecido por omisión corresponde al momento de arranque de Pupil Capture o Pupil Service (lo que suceda primero).

Resultados

A partir de los datos entregados por la suite de Pupil-Labs se puede obtener la medida del diámetro pupilar. Estos datos también se encuentran en el archivo .CSV devuelto por el *software* del *eyetracker*. El error del diámetro de la pupila es función de un parámetro que determina la confiabilidad de la medida tomada. Este dato varía entre 0 y 1, el valor de la medición se forma de la siguiente manera:

*Diámetro de pupila = diameter_3d +/- (1 - model_confidence) * diameter_3d*

Donde *diameter_3d* y *model_confidence* son campos en el archivo de datos devuelto por el *software* del *eyetracker*, (**Kassner et al**. (2014)). Ver Anexo II.

Conclusiones

Se planteó realizar la puesta a punto de un sistema que realiza medidas de diámetro pupilar con posibilidad de movimiento de sujeto, resultados confiables y portabilidad de *hardware*. Se diseñó un sistema de *hardware* y *software* portable a partir de equipamiento disponible teniendo en cuenta cada aspecto que podía ser una limitación para la implementación de la solución. Se logró un sistema apto para realizar mediciones de pupilas para permitir su empleo en investigaciones sobre el funcionamiento de la visión humana. Este sistema cumple con las características especificadas en los requisitos.

Anexo I: Protocolo para realizar mediciones

Para que el sistema entregue valores confiables y precisos debe utilizarse de manera correcta. Este sistema necesita ser calibrado y posicionado para cada sujeto. Por lo tanto, es necesario articular la tarea del experimento con la preparación del sistema. Se enumeran a continuación las tareas que deben realizarse antes de medir sujetos. Se recomienda realizar estas tareas en orden para tener al sujeto en posición la menor cantidad de tiempo posible antes del experimento.

- Encendido del PC
- Selección de cámara frontal (100º o 60º) y colocación en el marco con herramienta provista
- Preparación del marco del *eyetracker* con la lente del ojo derecho (60º)
- Colocación del marco en el sujeto (figura 2)
- Conexión del *eyetracker* vía USB
- Arranque del intérprete Octave
- Arranque del programa Pupil Capture
- Configuración de Pupil Capture
    - Seleccionar método de calibración adecuado para el experimento, (**Kassner et al.** (2014))
    - Habilitar el monitoreo del diámetro de la pupila
    - Habilitar el monitoreo de la confiabilidad del modelo del *gaze*
    - Verificación del Socket para IPC
    - Habilitar cámara del ojo derecho
    - Validar que se ejecute el proceso de la cámara del ojo derecho
- Posicionado del sujeto frente a la pantalla del estímulo
- Configuración de las cámaras
    - Ojo Derecho
        - Posicionar la cámara del ojo de manera que el ojo entero quepa en el campo de visión de la cámara (figura 3)
        - Regular el tiempo de exposición de la cámara para obtener buen contraste en la imagen
        - Pasar el monitoreo del ojo al "Modo Algoritmo"
        - Regular la sensibilidad de la detección de la pupila para cubrirla entera excluyendo las otras partes oscuras del ojo
        - Validar el modelo del ojo controlando en el modo *debug*
    - Frontal
        - Posicionarla de manera que entre en su FOV la mayor parte del FOV del ojo derecho
- Cargar la librería ZeroMQ (se puede hacer manualmente en el intérprete *Octave* o desde el programa)
- Calibración de los modelos del *gaze* y el ojo del *eyetracker* (se puede hacer manualmente en *Pupil Capture* o desde el programa)
- Ejecución del experimento

Fig. 2 Colocación del marco de anteojo en el sujeto.

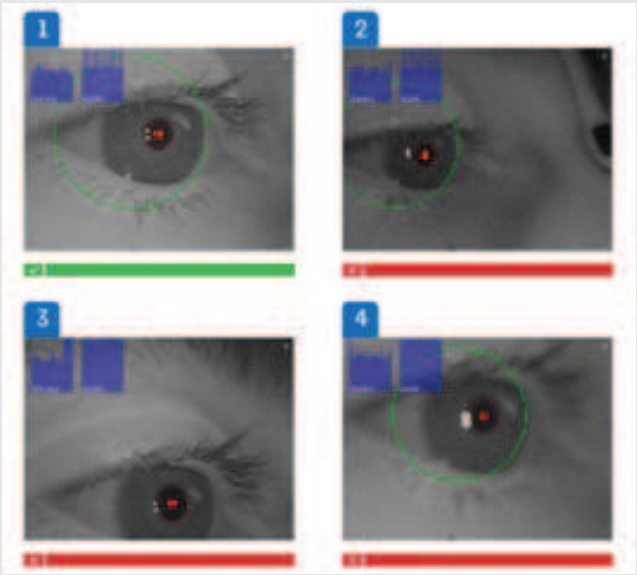

Fig. 3: 1- Correcto. 2- Incorrecto, ojo descentrado.
3- Incorrecto, ojo fuera del campo visual de la cámara.
4- Incorrecto, cámara desenfocada

Anexo II: Ejemplo - Resultado de archivo CSV

Se exponen a modo de ejemplo solo algunos campos de interés de la tupla y algunos registros del archivo pupil_positions.csv correspondientes a mediciones de la pupila.
timestamp [s]: estampa relativa de tiempo en que se toma la medida.
index: fotograma más próximo del video grabado.
confidence: validez de la medida - de 0 (mala) a 1 (excelente).
diameter_3d [mm]: diámetro de la pupila.
model_confidence: precisión del cálculo del tamaño de la pupila.

Se obtiene así: *Diámetro de pupila = diameter_3d +/- (1 - model_confidence) * diameter_3d*

```
$ cut -d ',' -f 1,2,4,14,15 pupil_positions.csv|tr , '\t'|column -t|head -20
timestamp   index confidence diameter_3d        model_confidence
1109.770421 0     1.0        5.8242450722712995 0.986042585811997
1109.837621 2     1.0        5.830889352034895  0.9856324173089237
1109.921621 5     1.0        5.837669568083826  0.9852772193013393
1109.980421 7     1.0        5.8686854503955015 0.9848442669655899
1110.039221 8     1.0        5.884321656151671  0.9845085873924075
1110.098021 10    1.0        5.922918158703925  0.9839617321709127
1110.165221 12    1.0        5.942944340664603  0.9834818138701795
1110.224021 14    1.0        5.865716612872556  0.9831016848341756
1110.291221 16    1.0        5.733040422495739  0.9828682148639523
1110.350021 18    1.0        5.610970405786894  0.9823701427810194
1110.417221 20    1.0        5.506256738911309  0.9818016511133232
1110.476021 21    1.0        5.394042791026594  0.9810050674763815
1110.543221 23    1.0        5.313590291105265  0.9800646787420764
1110.602021 25    1.0        5.232971559239439  0.978737642790084
1110.669221 27    1.0        5.18644243541688   0.9778565308808607
1110.736421 29    1.0        5.161220693828513  0.9773666449608087
1110.795221 31    1.0        5.158631982416463  0.9768150719671631
1110.862421 33    1.0        5.155725506374164  0.975999598457522
1110.921221 35    1.0        5.21023962127008   0.9760677733550049
```

Referencias Bibliográficas

Brainard, D. H. (1997) "The psychophysics toolbox", *Spatial vision*, Vol. 10, N. 4, pp. 433-436.

Eaton, J. W. (2018) *GNU Octave version 5.1.0 manual: a high-level interactive language for numerical computations:* https://www.gnu.org/software/octave/doc/v5.1.0/

Kassner, M. Patera, W. and Bullying, A. (2014) *Pupil: An Open Source Platform for Pervasive Eye Tracking and Mobile Gaze-based Interaction*. https://docs.pupil-labs.com/

Pieter Hintjens (2019) *ØMQ - The Guide:* http://zguide.zeromq.org/page:all

The Linux Kernel Development Community (2019) *The Linux Kernel documentation*. https://www.kernel.org/doc/html/latest/

The Archlinux Community (2019) *Archlinux Wiki:* https://wiki.archlinux.org/

Este informe técnico se realizó en el 2° Semestre del año 2019 en el Dpto. de Luminotecnia, Luz y Visión de la FACET (UNT) en el marco de estudios de Doctorado en Medio Ambiente Visual e Iluminación Eficiente, titulado "Percepción del movimiento en el rango mesópico" de Juan Ignacio Contino (PUE 114 CONICET).

Juan Ignacio Contino
Ingeniero en Computación, graduado en la Facultad de Ciencias Exactas y Tecnología (FACET) de la Universidad Nacional de Tucumán (UNT). Becario doctoral (CONICET) del Doctorado en Medio Ambiente Visual e Iluminación Eficiente, UNT (FACET). Docente de Física del Colegio Salesiano Belgrano. Profesor del curso "Deep Linux" (UNT/FACET). Se aplicó al campo de las telecomunicaciones en organizaciones nacionales, de Latinoamérica y de USA (Intraway Corp.). Participó del proyecto de transferencia tecnológica "DTEC" (ANPCyT/UNT). Su área de especialidad profesional se centra en la implementación de servicios en redes, la configuración de los ambientes operativos, la optimización de éstos y el diseño de *appliances*.

Andrés Martín
Licenciado en Filosofía de la Universidad Nacional de Tucumán (UNT). Doctor en Medio Ambiente Visual e Iluminación Eficiente, graduado en la Facultad de Ciencias Exactas y Tecnología (FACET) de la UNT. Investigador Adjunto del CONICET en el Instituto de Investigación en Luz, Ambiente y Visión (ILAV), instituto de doble dependencia UNT – CONICET. Su área de investigación comprende diferentes aspectos de la percepción visual humana, especialmente la percepción del movimiento, el color y el espacio iluminado. Se encuentra involucrado en proyectos sobre aplicaciones de los conocimientos perceptuales en diferentes ámbitos productivos. Realizó publicaciones en importantes revistas de impacto internacional como: *Frontiers in Psychology, Psychology & Neuroscience, Human Perception and Performance*.