
Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

# Optimización de un modelo de búsqueda visual: Adaptaciones y mejoras para cIBS

Tesis de Licenciatura en Ciencias de la Computación

Gonzalo Ruarte

Director: Juan E Kamienkowski
Codirector: Gastón Bujia
Buenos Aires, 2021

# OPTIMIZACIÓN DE UN MODELO DE BÚSQUEDA VISUAL: ADAPTACIONES Y MEJORAS PARA CIBS

Las neurociencias han avanzado al punto en el que se conoce en profundidad el procesamiento visual en el cerebro de una imagen que se encuentra en el centro del campo visual y con los ojos fijados. Asimismo, el sistema visual biológico está íntimamente vinculado al sistema oculomotor, ya que, para salvar el limitado ángulo visual en el que el sistema visual posee alta resolución, los ojos se están moviendo constantemente, muestreando distintas posiciones con la mirada (fijaciones) de la imagen cada 300 ms aproximadamente. Por su lado, la Computación ha provisto modelos altamente efectivos para diversas tareas relacionadas a la percepción visual como por ejemplo, la predicción de la distribución de fijaciones en una imagen cuando se la observa libremente. Sin embargo, la predicción de movimientos oculares (el orden de las fijaciones o scanpath) durante la tarea de búsqueda visual sigue siendo un problema complejo, sin respuestas definitivas. Actualmente, existen diversos algoritmos que intentan predecir el scanpath de un sujeto al momento de buscar un objeto en una escena natural con distintos enfoques. En este proyecto nos centramos en uno de estos modelos, propuesto por el Laboratorio de Inteligencia Artificial Aplicada, para predecir los scanpaths durante la tarea de búsqueda visual: cIBS [SVRHM] o correlational Ideal Bayesian Searcher.

**Palabras claves:** Modelo bayesiano, Tarea visual, Movimiento ocular, Escenas naturales, Inteligencia Artificial.

# A VISUAL SEARCH MODEL OPTIMIZATION: EXTENSIONS AND IMPROVEMENTS FOR CIBS

Neurosciences have advanced to the point where the brain's visual processing of an image that is in the center of the visual field with the eyes fixed on it is known in depth. Moreover, to overcome the limited visual angle in which the visual system has high resolution, the biological visual system is closely linked to the oculomotor system since the eyes are constantly moving while sampling different positions (fixations) of the image every 300 ms. In turn, Computer Sciences have provided highly effective models for various tasks related to visual perception, such as predicting the distribution of observers' fixations over an image when it is freely observed (free viewing task). On the other hand, the fixation order (scanpath) prediction on a visual search task remains a complex problem, with no definitive answers. Currently, there are several algorithms that attempt to predict the scanpath of a subject when looking for an object in a natural scene, each with different approaches. In this project we focus on one of these models, proposed by the Laboratorio de Inteligencia Artificial Aplicada, to predict scanpaths during a visual search task: cIBS [SVRHM] or correlational Ideal Bayesian Searcher.

**Keywords:** Bayesian Model, Visual Task, Scanpath, Natural scenes, Artificial Intelligence.

# AGRADECIMIENTOS

A todos aquellos que hicieron posible que yo pueda estudiar lo que disfruto. A la UBA, a la FCEN y al DC por proveer un espacio apto para desarrollar nuestro conocimiento e inculcarnos importantes valores. A Fermín Travi con quien culminamos nuestra última etapa de la carrera a la par, y gracias a quien conseguimos esta oportunidad de investigación. A los directores por brindarnos dicha oportunidad y por contagiarnos el gusto por la investigación. A Agustín Penas, quién me acompañó buena parte de la carrera y con invaluables aportes durante nuestra cursada.

## PEER REVIEW

Los resultados preliminares de esta tesis fueron publicados en SVRHM [https://openreview.net/group?id=NeurIPS.cc/2021/Workshop/SVRHM#accept--poster-], presentados en la SAN [https://san2021.saneurociencias.org.ar/] y están disponibles en https://github.com/FerminT/VisualSearchBenchmark/tree/SVRHM

# Índice general

# 1. INTRODUCCIÓN

## 1.1. Motivación

La identificación de estructuras cerebrales y mecanismos asociados a tareas en las cuáles los humanos somos naturalmente expertos ha sido una gran inspiración y motivador del desarrollo de la inteligencia artificial, y se espera que en el largo este proyecto también lo sea, guiando el desarrollo de nuevos algoritmos en inteligencia artificial [5, 8, 9, 10, 11]. Así, el trabajo en la frontera entre las ciencias cognitivas y la inteligencia artificial permite proponer avances en ambas disciplinas [11]. Además, el desarrollo de buenos buscadores pueden tener aplicaciones en áreas como la robótica, desde permitirle a una cámara que transmite eventos que dirija su lente de forma autónoma, a avanzar en la creación de prótesis oculares con comportamiento similar al humano. Finalmente, la combinación de estos modelos con el estudio de variaciones individuales puede tener aplicaciones importantes en patología, ya que deficits muy comunes en memoria de trabajo o control inhibitorio tienen un impacto fuerte en tareas de búsqueda visual aunque sólo se tienen hipótesis generales de cuál es su impacto y por lo tanto son más difíciles de rehabilitar [12].

Particularmente para este trabajo se plantea como hipótesis que es posible segmentar la resolución de la tarea en una secuencia de pasos delimitados por los movimientos oculares, asumiendo que la actualización de la información tanto a nivel perceptual como cognitivo ocurre fijación-a-fijación [35, 36] entre otros). Dentro de cada paso se encuentran a su vez procesos de captura de información bottom-up, que potencialmente modelados con redes neuronales, y procesos de integración de información y actualización de las expectativas (top-down), que pueden ser modelados como procesos de inferencia bayesiana [34, 12], los cuales guían los movimientos oculares a través de una imagen.

## 1.2. El ojo y los movimientos oculares

*Fig. 1.1:* Ejemplos de imágenes foveadas.

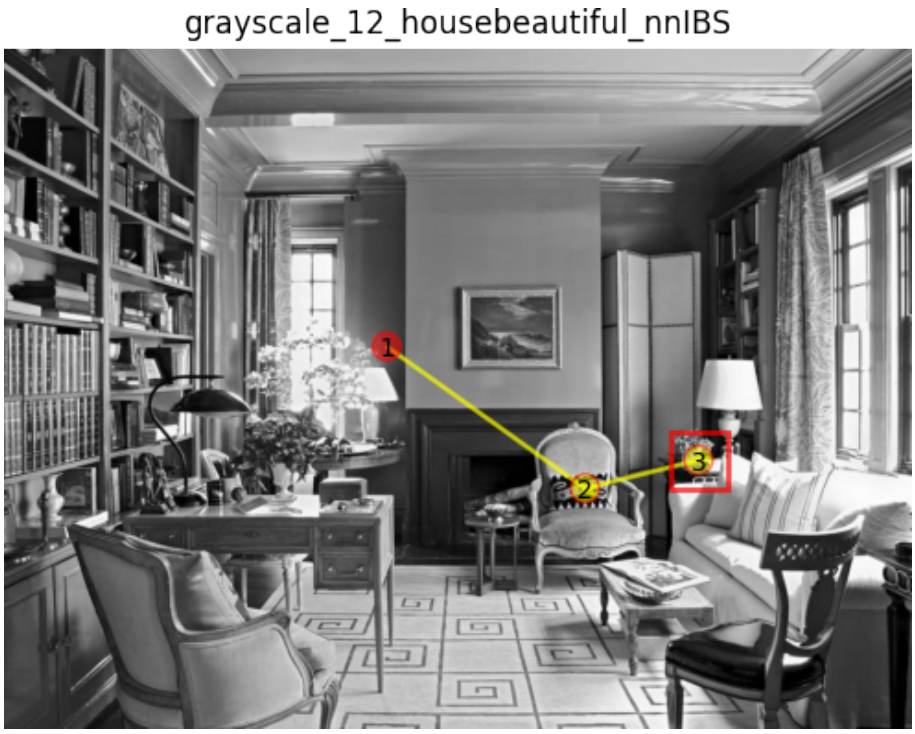


*Fig. 1.2:* Cada punto amarillo con un número es una fijación. El tamaño del círculo es análogo al tamaño de la fóvea. La linea entre dos fijaciones consecutivas representa una sacada. Todas las fijaciones ordenadas forman un *scanpath*.

Por construcción, el ojo tiene una zona de alta definición en el centro del campo visual, fuera de la cual la información que llega al cerebro está "blureada" (Ver figura 1.1). Está región, de aproximadamente 1 grado de ángulo visual, se la conoce como fóvea.

Para poder procesar una escena visual más amplia, es necesario hacer un muestreo, dirigiendo la fóvea hacia distintas regiones de la misma y deteniéndose en el transcurso para poder incorporar información. A estas acciones se las llama movimientos oculares.

Si bien hay varios tipos de movimientos oculares, son de interés las *sacadas* y *fijaciones*. Las *fijaciones* son puntos en donde se mantiene firme o estable la mirada. Duran 250 ms aprox. (rango [50ms, 1000ms]) y son momentos donde se incorpora información. Las *sacadas* son movimientos muy rápidos que cambian el punto de fijación. Pueden variar su longitud según la tarea: al leer las sacadas serán cortas, mientras que al explorar una habitación las sacadas serán largas. Si bien pueden ser movimientos voluntarios, también existen sacadas involuntarias cuando tenemos los ojos abiertos. Un ejemplo de ellas son las que ocurren durante las fijaciones (también llamadas micro-sacadas debido a su corta longitud). En las sacadas no hay incorporación de información.

A una secuencia ordenada de puntos de fijación (x,y) se la llama *scanpath* (Ver figura 1.2). En general cuando se habla de *scanpath* no se tiene en cuenta la duración de las fijaciones. En este proyecto tampoco es tenida en cuenta, aunque es interesante porque es indicador del costo de procesamiento [24, 25].

A su vez, los movimientos oculares son guiados por la atención visual (i.e.: la capacidad cognitiva de retener información relevante y descartar la información irrelevante) que asimismo depende de la finalidad del sujeto en cuestión. Por ejemplo, si se quiere

encontrar un auto, las fijaciones van a estar concentradas en las calles. A esta finalidad u objetivo con el que los sujetos observan una imagen se la llama Tarea. La tarea más sencilla en la que se reportan secuencias de fijaciones se llama free-viewing y consiste en que los sujetos vean una imagen con total libertad durante un tiempo determinado. Un paso adicional de complejidad implica que haya un objetivo como puede ser **búsqueda visual** que consiste en presentarle una imagen al sujeto y éste debe guiar su vista con el objetivo de encontrar un objeto particular que puede o no estar presente, en cuyos casos se los llaman *target-present trial* y *target-absent trial* respectivamente. Existen otras tareas como la **memorización**, pero para este trabajo interesan las dos mencionadas anteriormente, en particular la búsqueda visual.

Dependiendo de la tarea los *scanpaths* van a seguir distintos tipos de patrones, por ejemplo en free-viewing se ha visto que los humanos suelen dirigir su atención a las caras presentes en las imágenes [17]. Por otra parte, al buscar un objeto particular, la información de contexto genera patrones distintivos (como el del auto mencionado anteriormente).

## 1.3. Mapas de saliencia para predecir posiciones (no secuencia)

En psicología y neurociencias, la saliencia es una propiedad que poseen determinados estímulos que los hacen destacar por encima de otros dentro de un mismo conjunto [23]. En visión, esto significa concentrarse en los elementos de una escena visual que se destacan de sus regiones vecinas.

La atención visual que poseen los humanos se guía mediante dos procesos distintos [37]:

- Bottom-Up: factores externos determinan la saliencia de un estímulo en base a sus propiedades inherentes en relación a la escena en segundo plano, como por ejemplo el color, brillo o textura de un objeto.

- Top-down: internamente se le da importancia a un objeto o característica basándose en información previa, las tareas actuales y planificación, como por ejemplo caras o texto.

La detección de saliencia está involucrada en muchos procesos sensoriales. La saliencia visual generalmente se operacionaliza midiendo las ubicaciones de las fijaciones. En consecuencia, la detección de saliencia actualmente se refiere a predecir fijaciones (saliency prediction) o detectar objetos destacados (salient object detection) en la forma de un mapa de saliencia (Ver figura 1.3). Particularmente interesa la predicción de saliencia ya que se encarga de predecir la posibilidad de que un humano se fije en determinada región de una imagen, y por otro lado, ésta se basa en la atención visual humana.

Estos modelos tenían en cuenta, inicialmente, la información bottom-up [22]. Hoy en día los modelos de saliencia están basados en redes profundas que utilizan modelos pre-entrenados en reconocimiento de objetos [20, 21, 26] y son capaces de incorporar información top-down, a tal punto en que por ejemplo el sesgo que tenemos los humanos de enfocarnos en las caras humanas al ver una imagen, es capturado por estos modelos hoy en día [16].

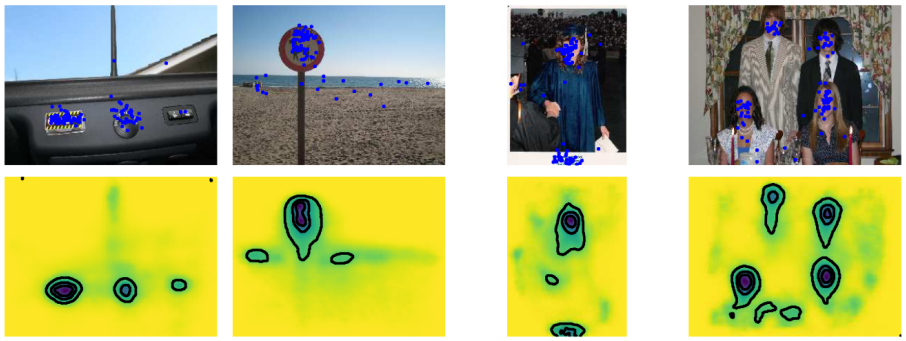

*Fig. 1.3:* Ejemplo de distribución de fijaciones humanas (primera fila) y sus respectivos mapas de saliencia predichos por DeepGazeII (segunda fila). Imagen sacada de [26]

## 1.4. Modelos de búsqueda visual

En la tarea de búsqueda visual, los sujetos deben buscar un estímulo en particular embebido en una imagen de fondo, en la cual el estímulo puede estar o no presente. Las características de la imagen, del estímulo o incluso la posible ausencia del mismo definen diferentes posibles enfoques a la hora de modelar este comportamiento. Por ejemplo, los modelos deberán tener en cuenta efectos como la inhibición de retorno (inhibition-of-return o IOR) el cual reduce la probabilidad de revisitar lugares ya observados en la imagen durante la búsqueda [27, 28]. En esta tesis interesa particularmente la búsqueda visual en imágenes naturales y queremos predecir los *scanpaths* en esta tarea cuando siempre el objetivo de la búsqueda esta presente.

Existen modelos que tratan de predecir *scanpaths* en otras tareas como en *free-viewing* o exploración libre, es decir, intentan generar *scanpaths* con patrones comportamentales similares a los de los humanos en dicha tarea [19]. Esta es una tarea más simple que la búsqueda visual, tanto en sí mismo como en términos de adquisición de datos, y por lo tanto existen conjuntos de datos más masivos. Los modelos de exploración libre buscan responder la pregunta de a dónde miramos cuando nos presentan una imagen. Interesan estos modelos entonces porque, por un lado, hay bastante estudio dedicado a esta tarea y por otro, al igual que en búsqueda visual, a partir de una imagen se computa un *scanpath* (o varios) utilizando *feature maps* de por medio.

Por su parte, los modelos de búsqueda visual son análogos a estos modelos de predicción de *scanpaths* en free-viewing pero incluyen la complejidad que acarrea dirigir la atención hacia un objeto (o tipo de objeto) particular. Típicamente, más allá del marco teórico en el que se basan, toman como input una imagen y un objetivo/visual task a partir de los cuales computan uno (o varios) scanpath. Las fijaciones del mismo son calculadas paso a paso a partir de uno o más *feature maps*. No obstante pueden haber diferencias sutiles que acaban siendo importantes. Hay modelos que no simulan la fóvea [4] y en caso de simularla no hay una única forma de hacerlo: se puede por ejemplo aplicar un filtro gaussiano [2] o utilizar métodos más complejos [9]. El objetivo se lo puede pasar al modelo o bien en forma de categoría [6], o bien con un ejemplo de dicha categoría [4], o bien con un recorte de la imagen [2], lo cual habla de qué tan bien generaliza un modelo a una categoría de objetos. La forma de simular el IoR puede estar implícita en el modelo [2] o puede ser explícita [4, 6]. El criterio para determinar si se encontró el objeto o no puede variar, aunque hoy en día lo común es utilizar un oráculo que indica que se alcanzó el número máximo de

fijaciones o que la última fijación está posada en el objetivo (o cerca de él). La cantidad de información que retiene el modelo respecto de fijaciones previas puede estar limitada o no. La siguiente fijación de un scanpath puede determinarse utilizando una estrategia WTA [4, 6] sobre los *feature maps* o una más compleja [2]. El modelo puede requerir un re-entrenamiento al ser usado en nuevos *datasets* [6] o no [2, 4]. Es decir que existen muchas formas en las que se pueden mejorar los modelos de hoy en día, no necesariamente relacionadas a su eficiencia.

Actualmente uno de los principales objetivos dentro del campo de estudio es que los modelos sean capaces de procesar información más compleja (top-down), particularmente respecto del contexto de la escena. Esto es especialmente importante en búsqueda visual ya que si la tarea es buscar un auto, vamos a dirigir nuestra atención a las calles y a las regiones inferiores de una imagen. La tarea de búsqueda visual es muy importante en este sentido, porque a pesar de requerir un componente top-down importante, es una tarea que realizamos de forma cotidiana, eficiente, automática y sin esfuerzo, y es uno de los bloques que permiten construir tareas más complejas. Una analogía es implementar una receta de cocina, donde además de buscar un ingrediente, debo utilizarlo para completar otra secuencia de pasos.

Respecto de los marcos teóricos, hoy por hoy predominan tres que no son excluyentes entre sí: modelos bayesianos, redes neuronales y aprendizaje por refuerzo. Particularmente en tres de los cuatro datasets de estudio se testeó un modelo de búsqueda visual, con el detalle de que los tres modelos se basan en paradigmas distintos: cIBS [2] que está basado en un modelo bayesiano y en menor medida utiliza redes neuronales, IVSN [4] cuyo núcleo es una parte de la red VGG-16 [33], y el modelo IRL [6] que utiliza aprendizaje por refuerzo. La presente tesis se enfoca en cIBS y en su desempeño en distintas condiciones (datasets), pero en [**?**] se pueden ver comparaciones del cIBS frente a otros modelos.

## 1.5. correlational-Ideal Bayesian Searcher (cIBS)

### 1.5.1. Bayesian Searcher

Para predecir los *scanpaths* de búsqueda visual humana, [12] propuso el Ideal Bayesian Searcher (IBS). El IBS considera cada posible próxima fijación secuencialmente y escoge la que maximiza la probabilidad de identificar la ubicación del objetivo correctamente después de la fijación. La ubicación de la fijación óptima en el paso $T+1$, $k_{opt}(T+1)$, se calcula mediante (ec. 1.1):

$$k_{opt}(T+1) = \arg\max_{k(T+1)} \{p(C|k(T+1))\} \tag{1.1}$$

Esta regla fue reescrita y condicionada a la ubicación $i$ como (ec. 1.2):

$$k_{opt}(T+1) = \arg\max_{k(T+1)} \{\sum_{i=1}^{n} p_i(T) p(C|i, k(T+1))\} \tag{1.2}$$

donde $p_i(T)$ es la probabilidad posterior de que el objetivo esté en la i-ésima ubicación dentro de la cuadrícula después de $T$ fijaciones mientras que $p(C|i, k(T+1))$ es la probabilidad de estar en lo correcto dado que la verdadera ubicación del objetivo es i, y

que la ubicación de la siguiente fijación es $k(T+1)$. $p_i(T)$ involucra el *prior*, el mapa de visibilidad ($d'_{ik(T)}$) y una noción de la ubicación destino ($W_{ik(t)}$):

$$p_i(T) = \frac{prior(i) \cdot \prod_{t=1}^{T} exp(d'^2_{ik(T)} W_{ik(t)})}{\sum_{j=1}^{n} prior(j) \cdot \prod_{t=1}^{T} exp(d'^2_{jk(T)} W_{jk(t)})} \tag{1.3}$$

El *template response*, $W_{ik(t)}$, cuantifica la similitud entre una posición i dada y el *template* desde la posición fijada $k(t)$ ($t$ es cualquier fijación previa). es definida como $W_{ik(t)} \sim N(\mu_{ik(t)}, \sigma^2_{ik(t)})$ donde:

$$\mu_{ik(t)} = \mathbb{1}_{(i=ubicacion\,del\,objetivo)} - 0{,}5\ ,\ \sigma_{ik(t)} = \frac{1}{d'_{ik(T)}} \tag{1.4}$$

Abusando la notación, en la ec. 1.3 $W_{ik(t)}$ se refiere a un valor extraído de esta distribución. IBS solo se había probado en imágenes artificiales, donde los sujetos necesitaban encontrar un parche de Gabor entre 1/f de ruido en una de las 25 ubicaciones posibles. Por ello, en [2] le realizaron las modificaciones para poder aplicar el modelo a movimientos oculares en escenas naturales.

### 1.5.2. Modificaciones a IBS para manejar escenas naturales

El IBS fue diseño para modelar las fijaciones en una tarea de búsqueda visual en imágenes artificiales, parches de gabor embebidos en un fondo de ruido 1/f [12]. Las posiciones de estos parches estaban restrigida a 25 posibles ubicaciones en un círculo. Al pasar a escenas naturales se transgreden algunas de las condiciones del experimento original, como por ejemplo que el objetivo puede tener distintos grados de similaridad con el fondo (por ejemplo en una imagen con muchas caras) o que las posiciones ya no están restringidas ni son regulares.

En un trabajo previo del grupo se exploró la posibilidad de llevar este enfoque a escenas naturales [2]. Se restringió la posible fijación a ubicaciones que se analizarán hasta los puntos centrales de una cuadrícula de $\delta$ x $\delta$ píxeles cada uno porque sería computacionalmente intratable calcular la probabilidad de fijar cada píxel de una imagen de 1024 $\times$ 768, e ineficaz ya que la información útil se extiende sobre regiones más grandes que un píxel. En consecuencia colapsaron los movimientos oculares a estos puntos: las fijaciones consecutivas dentro de una celda fueron combinadas en una fijación para ser consistentes con el comportamiento del modelo.

El modelo IBS original usa de *prior* a una distribución uniforme. Se introdujo un modelo de saliencia a modo de *prior* porque estaban tratando de calcular fijaciones del modelo en una escena natural. El *prior* (i) es el promedio de la saliencia en la i-ésima celda de la cuadrícula.

Es importante destacar que detectar la presencia del objetivo en una determinada posición en escenas naturales no es tan sencillo como en los estímulos artificiales, donde todas las ubicaciones incorrectas son igualmente disímiles. En las escenas naturales, a menudo hay distractores (es decir, posiciones en la imagen que son visualmente similares al objetivo, especialmente si se ven con poca visibilidad). Por lo tanto, propusieron un

*template response* redefinido como $\widetilde{W}_{ik(t)} \sim N(\widetilde{\mu}_{ik(t)}, \widetilde{\sigma}^2_{ik(t)})$, donde $\widetilde{\mu}_{ik(t)} \in [-1,\ 1]$ se definió como (ec. 1.1):

$$\widetilde{\mu}_{ik(t)} = \mu_{ik(t)} \cdot (d'_{ik(T)} + \frac{1}{2}) + \phi_i \cdot (\frac{3}{2} - d'_{ik(T)}) \tag{1.5}$$

donde $\phi_i$ cuantificó cuán similar es cada ubicación i al objetivo. En [2], propusieron utilizar la correlación cruzada entre la plantilla objetivo y la ubicación que llamamos el modelo correlation IBS o cIBS.

Asimismo, incluyeron un modelo de mapa de visibilidad. Los parámetros del modelo gaussiano 2d se eligieron a priori y se estimaron a partir de los valores informados en [12, 13], y fueron los mismos para cada participante. En la gaussiana bivariada $N(\mu, \Sigma)$ se centró en cada punto de fijación (μ es la coordenada 2D en píxeles), y su covarianza era $\Sigma = \begin{pmatrix} 2600 & 0 \\ 0 & 4000 \end{pmatrix} pxs^2$. Además, modificaron $\sigma^2_{ik(t)}$ para mantener la varianza en función de la visibilidad, pero incorporaron dos parámetros (ec. 1.6):

$$\widetilde{\sigma}_{ik(t)} = \frac{1}{a \cdot d'_{ik(T)} + b} \tag{1.6}$$

Los parámetros a y b modulan conjuntamente la inversa de la visibilidad e impiden que $1/d'$ diverja. Estos parámetros no estaban incluidos en el modelo original probablemente porque $d'$ se estimó empíricamente (a partir de miles de ensayos e independientemente para cada sujeto) y la $d'$ nunca fue exactamente igual a cero. [13] simplificó la tarea al ajustar un mapa de visibilidad construido a partir de un modelo de primer principio que proponía una función analítica con varios parámetros, que aún deben ajustarse para cada participante. En [2], lo simplificaron aún más usando una gaussiana bidimensional con los mismos parámetros por cada participante, lo que evitó una posible fuga de información sobre los patrones visuales al modelo. Los parámetros se tomaron a priori (estimados a partir de los parámetros en [12, 13]). Eligieron los parámetros del modelo utilizando un procedimiento clásico de búsqueda en cuadrícula en un experimento anterior con un conjunto de datos más pequeño y los mismos mejores parámetros ($\delta = 32$, a = 3 y b = 4) se utilizaron para todos los modelos.

A partir de las modificaciones hechas, cIBS parte de un mapa de saliencia de la imagen de entrada que indica las regiones que llaman inicialmente la atención (*prior*). Luego, el modelo sucesivamente evalúa la similitud de todas las regiones de la imagen (W) pesadas por el mapa de visibilidad, que da cuenta de la inhomogeneidad del campo visual (mucha resolución en el centro y una caída rápida a medida que se aleja). En base a esta información y al *prior*, el modelo decide la mejor posición de la siguiente fijación para encontrar el objetivo en dos fijaciones. Todavía hay mucho espacio para mejorar esta primera versión: En primer lugar, en los trabajos anteriores del grupo de Geisler [12, 13] han destacado la importancia de estimar los mapas de visibilidad. Los parámetros de la Gaussiana utilizada en el mapa de visibilidad actual podrían explorarse en más detalle, así como también se podrían estudiar nuevos modelos como en [9]. En segundo lugar, el modelo fue desarrollado para imágenes en escala de grises [2], mientras que muchos conjuntos de datos son en color [7, 5]. En tercer lugar, mientras que el modelo original [2] asignaba a cada posición un -$\frac{1}{2}$ o $\frac{1}{2}$ según si estaba o no el objetivo, el presente modelo incorpora una noción de similaridad calculando la correlación en una ventana, de la imagen con el objetivo, obteniendo valores continuos entre [-$\frac{1}{2}$, $\frac{1}{2}$] [2]. Por otra parte, el modelo de estudio fue desarrollado para tareas

de búsqueda visual en escenas naturales, pero esto no quita que su uso pueda extenderse a otros entornos. Finalmente, el presente modelo considera toda la historia previa, aunque esta va perdiendo peso naturalmente con las sacadas. Sin embargo, distintos estudios comportamentales [15], y también modelos [8, 9] han mostrado que existe un límite en la historia que se considera para planear cada sacada de aproximadamente 4 fijaciones.

# 2. OBJETIVOS

## 2.1. Objetivo general

En este proyecto, se propone avanzar sobre el modelo desarrollado por el grupo [2], incorporando nuevos aspectos, tanto para mejorar su desempeño en el *dataset* generado en el LIAA como para adaptarlo a otros datos. Este modelo aún tiene amplios márgenes de mejora, por ejemplo adaptándolo a imágenes en color, o mejorando el método de comparación de las distintas regiones de la imagen con el objetivo, y permitiendo que sea capaz de generalizar a otras vistas del objeto u objetos de la misma categoría.

Se busca comprender, a nivel algorítmico, cómo el cerebro humano codifica la información del entorno visual y la integra para planificar los siguientes pasos dentro de una tarea compleja que involucra una secuencia de ellos hasta su resolución. Para ello se busca definir e implementar modelos computacionales capaces de seguir patrones comportamentales análogos a aquellos que caracterizan a los seres humanos. agregar posibles campos de aplicación.

Una parte importante de esta tesis es la portabilidad del código del modelo de código cerrado, implementado originalmente en MATLAB, a un lenguaje de código abierto como lo es Python. La importancia de la utilización de código abierto en el modelo es que mejora su accesibilidad y esto facilitará tanto su revisión así como su utilización [29]. Además, esto es una herramienta fundamental para el desarrollo de futuras extensiones o modificaciones al modelo y aportará un modelo end-to-end de predicción de *scanpaths* programado completamente en código abierto.

## 2.2. Objetivos particulares

En el presente trabajo, se propone continuar desarrollando el modelo propuesto en [2], mejorando su performance en distintos *datasets* disponibles (incluyendo el propio) en conjunto con Fermín Travi. Para ello, se plantean los siguientes objetivos:

- Reunir los distintos **conjuntos de datos** disponibles para la tarea de búsqueda visual, para ubicarlos dentro de un criterio y estructura de datos en común (tarea que se realizará en conjunto con Fermín Travi).
- Evaluar la implementación de las siguientes **extensiones del modelo** [2]:
  1. Distintas formas de calcular el *template response*.
  2. Cambio en la discretización de imágenes.
- Evaluar la implementación de las siguientes **optimizaciones al modelo** [2]:
  1. Paralelización de cómputos.
  2. Incorporación de cálculo de *prior*.
- **Portabilidad del modelo original escrito en Matlab a Python y optimización**. Esto permitirá ejecutarlo más fácilmente en los servidores del LIAA / clusters del ICC, e integrarlo al benchmark desarrollado por Fermín Travi.

# 3. MATERIALES Y MÉTODOS

## 3.1. Datasets

En cualquier campo de investigación y en particular en el de búsqueda visual existe la problemática de las diferencias a la hora de plantear una experimentación. Si bien el problema a resolver puede ser el mismo, se puede disponer de distinto equipamiento, distintos datos, distintos criterios, distintas decisiones.

Una de las tareas comunes al proyecto Visual Search Benchmark (VSB de ahora en más) [1] fue la de recopilar *datasets* públicos para poder realizar los experimentos correspondientes. En particular se trabajó con cuatro *datasets*:

1. Interiors [2].
2. Unrestricted [4].
3. COCOSearch-18 [7].
4. Microwave-Clock Search (MCS) [5].

Además de las imágenes, el formato que utilizan para describir los *scanpaths* no es necesariamente el mismo. De todos ellos en principio interesan únicamente los que provengan de *target-present trials*. Al no ser necesario entrenar cIBS, es indistinto si los *scanpaths* fueron catalogados como train, valid o test en el *dataset* del que provienen.

De ahora en más resulta conveniente definir dos conceptos:

- Scanpath trivial: aquel cuya fijación inicial cae dentro del *target bounding box* (los límites del objetivo dentro de la imagen), es decir, tiene largo 1.
- Trial trivial: aquel en el que por definición dentro del *dataset* (suponiendo que no se utiliza la misma imagen para más de una tarea), la fijación inicial cae dentro del *target bounding box*.

Para llevar todos los *datasets* a un mismo criterio, se aplicó el denominado criterio $\delta$. Esto es: se corta el scanpath humano en cuanto el píxel de una fijación + tamaño fóvea / 2 cae sobre el *target bounding box* (es decir, se arma un cuadrado de dimensiones $\delta$ x $\delta$ centrado en la fijación). El tamaño de la fijación se estima sobre cada *dataset* a partir de los datos experimentales. En cIBS e IVSN, sus autores ya lo habían estimado (32x32 y 45x45, respectivamente). En COCOSearch18 fue estimado a partir del tamaño de fóvea usado en el modelo (52x52) y en MCS se estima a partir del tamaño de la imagen (20x20), ya que los sujetos vieron un display con una resolución distinta. En todos los casos, se cubre un área similar.

- Todos los *trials* que son triviales bajo este criterio fueron removidos.
- En todos los *datasets*, al procesar los *scanpaths* humanos, se descartan los triviales. 133 *scanpaths* triviales fueron quitados del Interiors *datasets*, 552 de COCOSearch18, 25 de Unrestricted, 121 de *dataset MCS* (tener en cuenta que en estos últimos dos se removieron imágenes triviales previamente).

- En el *dataset Interiors*, la fijación inicial puede estar mal por un error del eye tracker, por lo que puede ser que la primera fijación caiga sobre el objetivo aún filtrando imágenes.
- Hay *scanpaths* humanos que tienen coordenadas por fuera de los límites de la imagen. Para subsanar esto, se los acota al mínimo y máximo posibles. De esta manera, tienen coordenadas válidas para los modelos (importante para la métrica fijación a fijación).

En las siguientes secciones se describe con mayor profundidad la aplicación del criterio a cada *dataset* junto con las particularidades encontradas en cada uno.

Tamaño de objetivos en los *datasets* luego de haberlos validado (porcentaje en relación al tamaño de imagen):

- Unrestricted:
  Tamaño promedio: 111.70x105.12 (11 %x8 %)
  STD altura: 68.58; STD ancho: 70.98
- Interiors:
  Tamaño promedio: 72.0x72.0 (9 %x7 %)
  STD altura: 0.0; STD ancho: 0.0
- COCOSearch18:
  Tamaño promedio: 254.08x280.43 (24 %x17 %)
  STD altura: 126.20; STD ancho: 139.18
- MCS:
  Tamaño promedio: 73.22x82.13 (14 %x14 %)
  STD altura: 58.06; STD ancho: 70.49

### 3.1.1. Validación de Unrestricted

En este *dataset* se tiene en cuenta únicamente la versión *Oracle* de los *scanpaths* de los sujetos (i.e.: los *scanpaths* son cortados en el momento en que una fijación cae en el objetivo, sin necesidad de que el sujeto haya presionado el botón [4]). Además, la misma imagen se presentaba dos veces para los humanos. Se tiene en cuenta el primer trial de cada imagen, ya que no se está evaluando la memoria o consistencia entre sujetos.

Por otro lado, se realizó la siguiente verificación: para aquellos *scanpaths* que tenían el flag de *target found* en True, la última fijación del scanpath debe ubicarse dentro del *target bounding box*. Hubo 2854 *scanpaths* donde se encontró al objetivo, y 411 que estaban marcados como *target found* pero que no tenían la última fijación dentro de los límites del objetivo. Esta gran diferencia se debe a que en su paper consideran que la última fijación está dentro de los límites del objetivo si la misma cae dentro de una ventana de 200x200 a partir del objetivo (es decir, la fóvea estaría a 100 píxeles de distancia del *target bounding box*). Estos *scanpaths* fueron modificados cambiando su flag de *target found* a False.

Se notó que puede haber fijaciones con una distancia mínima de hasta 45 píxeles entre sus respectivos centros (a pesar de usar ventanas de 200x200). Este último punto no es nada trivial, ya que en cIBS no hace esa distinción (i.e.: no puede haber fijaciones cuyos centros estén a una distancia menor del tamaño de fijación), entonces a la hora de comparar *scanpaths* de cIBS con los de este *dataset*, es un punto a tener en cuenta.

| | Interiors | Unrestricted | MCS | COCO Search18 |
|---|---|---|---|---|
| #Subjects | 57 | 15 | 27 | 10 |
| #Images | 134 | 234 | 1790 | 2489 |
| #Scanpaths* | ~3,8K | ~2,8K | ~3,7K | ~22K |
| Average scanpath length* | 5,2±2,36 | 10,7±10,9 | 3,76±2,0 | 2,79±1,22 |
| People | ✗ | ✓ | ✗ | ✗ |
| Exact target | ✓ | ✗ | ✗ | ✗ |
| Distractors | ✓ | ✗ | ✗ | ✗ |
| Color | ✗ | ✗ | ✓ | ✓ |
| Fovea size | 32x32 | 45x45 | 20x20 | 52x52 |
| Mean target size | 72x72 | 115x105 ±68x71 | 73x82 ±58x70 | 254x280 ±126x139 |
| Image res. | 768x1024 | 1024x1280 | 508x564 | 1050x1680 |

*Fig. 3.1:* Comparación entre *datasets*.

Vale la pena remarcar que este *dataset* contiene varias imágenes que, o bien están difuminadas, o bien el objetivo se encuentra fuera de foco, o bien poseen muchos objetos aglutinados.

### 3.1.2. Validación de Interiors

Se hizo la misma verificación en este *dataset* y se encontraron 3810 *scanpaths* donde se encontró al objetivo, y 482 que estaban marcados como objetivo found pero que no tenían alguna fijación dentro de los límites del objetivo. Estos *scanpaths* fueron modificados cambiando su flag de *target found* a False. Hay una gran diferencia en el rendimiento de los humanos en este *dataset* frente a aquellos de Unrestricted: hubo un 51 % de efectividad en Interiors, pero un 79 % de efectividad en Unrestricted. Esto se debe principalmente a que al construir este *dataset* se limitó la cantidad de sacadas permitidas a un número muy bajo (2, 4 o 8 para la mayoría de los *trials*).

Además, este *dataset* contenía algunos *trials* que no cumplían el límite de 2, 4, 8 o 12 sacadas como máximo, si no que tenían números bastante mayores como 16 o 64. En estos casos, se truncó el scanpath a las primeras 12 sacadas y se cambió su número máximo de sacadas a 12. Si el scanpath original tenía una cantidad de sacadas menor o igual a 12, se lo mantuvo con el campo *target found* en True si correspondía. Si la cantidad de sacadas era mayor a 12, el campo se puso en False. Hubo 108 *scanpaths* que fueron truncados. En este *dataset* no hay fijaciones cuyos centros tengan una distancia menor al tamaño de fijación usado (32x32).

### 3.1.3. Validación de COCOSearch-18

Solo los datos de validación y entrenamiento están disponibles (que comprenden al 80 % del *dataset*) mientras que los datos de testing no fueron habilitados por los autores. No parecen indicar en el paper el tamaño que usaron para aglutinar a las fijaciones humanas, por lo que se utilizó un script que calcula la distancia entre fijaciones consecutivas para tener un estimado (similar al usado en el dataset Unrestricted). Al igual que en los *datasets* anteriores, se excluyeron los *scanpaths* que dicen haber encontrado al objetivo pero su última fijación cae por fuera de la misma.

Con este criterio unificado para recortar *scanpaths* hay 22.809/24.880 donde se encontró al objetivo, y ningún scanpath donde estaba marcado erróneamente que se encontró al objetivo. De estos 22.809 *scanpaths*, 13.484 contenían una fijación previa a la última que caía sobre el objetivo (usando un oráculo esos *scanpaths* deberían ser más cortos).

Se encontraron imágenes repetidas en los *trials* humanos. Esto se debe a que usaron la misma imagen para tareas distintas (por ejemplo, en un *trial* para encontrar el inodoro, en otro trial para encontrar el lavamanos). Se duplicaron esas imágenes y fueron renombradas, para indicar que pertenecen a *trials* distintos. Una cuestión a notar es que la memoria humana pudo haber influido en estos *trials*. Hay 233 imágenes que se mostraron dos veces, y 15 imágenes que se mostraron tres veces.

El mismo script que procesa los *scanpaths* humanos se encarga de estimar distintos parámetros que no figuran en el paper [7]. Por ejemplo: las coordenadas de las fijaciones iniciales, el largo máximo de los *scanpaths*, qué tamaño de ventana usaron para aglutinar las fijaciones humanas. Se estimó que las fijaciones humanas fueron aglutinadas en parches de 52x52, que es el tamaño de fijación standard utilizado en el modelo de ellos (cubriendo aprox. 3 % y 5 % de la imagen con cada fijación). Un 6 % de los *scanpaths* humanos tienen fijaciones consecutivas con un tamaño menor a ese parche.

A su vez, dado que solo está disponible el 80 % del *dataset* (de los *scanpaths* humanos), se encontraron 612 (de 3101) imágenes inutilizadas. Se estima que esas imágenes están en los *trials* de *testing* faltantes. Fueron eliminadas.

### 3.1.4. Validación de MCS

Surgieron varios problemas con este *dataset*:

Las imágenes se mostraron en distintos tamaños en el monitor. Para training, mantuvieron el tamaño de la imagen, pero el monitor tenía una resolución bastante mayor (1280x800). Para testing, reescalaron y pusieron padding en las imágenes para el monitor de tamaño 1680x1050. Los *scanpaths* humanos están en estas resoluciones. Hubo que reescalarlos para que estén dentro de la imagen. No hay datos sobre los *target bounding boxes*. Se tuvo que descargar 40GB del *dataset* original (COCO) y cargarlo a través de Tensorflow, que contiene las anotaciones (como los bbox de los objetos), para extraer toda esta información. Hay 30 imágenes que no tienen información sobre su *target bounding box* (no están anotadas en COCO) por lo que fueron eliminadas del *dataset* (notar que algunas estaban en ambas categorías, por lo que en total son 47 imágenes eliminadas). Hay imágenes que tienen tanto reloj como microondas, y tienen el mismo nombre, por lo que hay que renombrarlas según su categoría (análogo a lo ocurrido en COCOSearch18).

Las imágenes tienen resoluciones variadas. Se reescalaron todas a su resolución promedio que es 508 x 564. Esto implicó reescalar los *scanpaths* y los *target bounding boxes*.

Dado que los sujetos debían apretar un botón para ver si el objetivo estaba o no, no hay certeza de que los *scanpaths* terminen en el objetivo incluso para *trials* exitosos. Por ende, se truncaron los *scanpaths* en el momento en que una fijación cayó sobre el objetivo (análogo a la versión Oracle de Unrestricted).

No mencionan el tamaño con el que se juntaron las fijaciones; no se sabe cuál es el tamaño de la fovea en los *scanpaths* humanos. Procesando los datos, 20x20 parece ser una buena estimación, que representa cerca del 4 % de la imagen (cIBS ocupa el 3/4 % de la imagen por cada fijación, Zhang es similar, COCOSearch18 anda por el 5 %). Con este tamaño, en 3775/5660 *trials* los humanos encontraron el objetivo.

Se ignoraron todos los *target-absent trials*, solo se tuvieron en cuenta los *target-present trials*. A diferencia de los demás *datasets*, cada sujeto no vio todas las imágenes, si no un subconjunto de ellas.

La fijación inicial es en el centro de las imágenes. No parece haber techo para la cantidad de fijaciones de los *scanpaths* humanos, ya que el más largo tiene 81 fijaciones.

### 3.1.5. Formato de imágenes

De la forma en que fue implementado, cIBSMat trabaja con imágenes de resolución 768x1024 en escala de grises. Adaptarlo a otras resoluciones o a color implica realizar cambios en la implementación. Ya estaba pautado hacer un port de MATLAB a Python, por lo que se optó en principio por reescalar las imágenes de *datasets* que usen otras resoluciones a 768x1024 en escala de grises. Esto abrió la puerta a dos nuevas preguntas: ¿Qué algoritmo se utilizará para reescalar las imágenes? ¿Qué algoritmo se utilizará para pasar las imágenes en color a escala de grises?

cIBS tiene un funcionamiento determinístico, es decir, al ejecutarlo sobre una misma imagen dos o más veces, se obtiene el mismo resultado. Sin embargo, al probar distintos algoritmos de transformación de imágenes (i.e.: mat2gray de MATLAB 2018) se observó que los *scanpaths* obtenidos eran distintos, con lo cual no es una elección trivial. Investigando más, se descubrió que son problemas recurrentes cuando se trabaja con imágenes.

Finalmente se optó por utilizar el módulo transform del paquete skimage [30]. Similarmente, para el pasaje a escala de grises se utilizó el módulo color del paquete skimage. Un detalle a tener en cuenta es que al reescalar las imágenes hay que hacer lo propio con los objetivos de las tareas, sus respectivas coordenadas dentro de la imagen (*target bounding box*) y las coordenadas de los *scanpaths* humanos dentro de cada imagen. De esta forma se evitan comparaciones erróneas por utilizar distintas escalas entre los elementos a comparar.

```
{
    "images_dir"               : "data/images/",
    "targets_dir"              : "data/templates/",
    "saliency_dir"               : "data/saliency/",
    "trials_properties_file" : "data/trials_properties.json",
    "save_path"               : "output/",
    "image_height"               : 768,
    "image_width"                : 1024
}
```

Listing 1: Ejemplo de formato JSON utilizado para describir los *datasets*.

cIBSPy requiere que el *dataset* a ser procesado tenga un archivo en formato JSON llamado *dataset_info.json* (Ver listing 1) que contenga toda la información pertinente a él. A su vez, se requiere otro archivo en formato JSON llamado *trials_properties.json* (Ver listing 2) que describa cada imagen dentro del *dataset*. Se requiere que las imágenes dentro de un *dataset* tengan la misma resolución.

```
[
    {
        "image": "grayscale_100_oliva.jpg",
        "target": "grayscale_100_oliva_template_2.jpg",
        "dataset": "Interiors Dataset",
        "target_matched_row": 381,
        "target_matched_column": 713,
        "target_height": 72,
        "target_width": 72,
        "image_height": 768,
        "image_width": 1024,
        "initial_fixation_row": 339,
        "initial_fixation_column": 400
    },

    ...

]
```

Listing 2: Ejemplo de formato JSON utilizado para describir las imágenes dentro de un *dataset*.

**Forma:** Diferencia vectorial entre pares alineados de sacadas, $(u_i - v_j)$- una medición de la similtud en la forma del scanpath. El resultado es normalizado por 2X la diagonal de la pantalla (el valor teórico máximo).
**Longitud:** Diferencia en longitud entre los límites de los vectores de la sacada- una medición de la similitud en la longitud de la sacada. El resultado es normalizado por la diagonal de la pantalla.
**Dirección:** Distancia angular entre vectores de sacada- una medición de la similitud de la forma cuando las amplitudes de las sacadas son distintas. El resultado es normalizado por $\pi$.
**Posición:** Diferencia en posición entre fijaciones alineadas- una medición común de la similitud del scanpath en términos de distancia Euclidea. El resultado es normalizado por la diagonal de la pantalla.
**Duración:** Diferencia en duración de fijación entre fijaciones alineadas- una medición de similitud en tiempo de procesamiento. El resultado es normalizado por la duración máxima de las dos que están siendo comparadas.

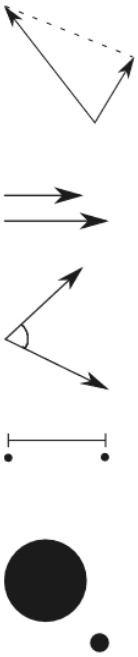

*Fig. 3.2:* Descripción de las métricas que forman parte de MultiMatch

### 3.1.6. Métricas

A la hora de evaluar los resultados, existen distintas metodologías para llevarlo a cabo. Es de particular importancia comparar el rendimiento entre modelos y humanos, utilizando métricas como el índice de Jaccard, Mean Agreement Score (MAS) [2] o la eficiencia acumulada [4]. No obstante, también es de relevancia el estudio sobre qué tanto difieren los *scanpaths* generados por los modelos frente a aquellos generados por los sujetos, en particular utilizando un conjunto de métricas denominado MultiMatch[1] [10, 11]. Asimismo, interesan métricas que permitan comparar *scanpaths* fijación a fijación [18]. Estas también son útiles a la hora de comprar distintas versiones de un mismo modelo, o distintos modelos entre sí.

En este trabajo se consideran especialmente relevantes a la eficiencia acumulada y MultiMatch. La primera consiste en calcular y acumular la cantidad de objetivos encontrados (*target found*) sobre el total de *trials* en función de la longitud de los *scanpaths*. Por su parte Multimatch es un conjunto de 5 métricas que compara de a pares de *scanpaths*, y cada una de ellas devuelve un número entre 0 y 1. Ellas son forma, longitud, posición, dirección y duración (Ver figura 3.2), pero en este trabajo solamente se tienen en cuenta las primeras cuatro. La forma en la que se utilizan es más sencilla de ver en el siguiente pseudocódigo:

Se obtiene una colección de tuplas dentro de las que se encuentran valores entre 0 y 1.3 Gráficamente se piensan a los valores de cada tupla como si correspondiesen al eje x y al eje y respectivamente, por lo que en los gráficos de MultiMatch cada punto se corresponde con una imagen y se posiciona según el valor de los promedios calculados. Intuitivamente se busca ver si el modelo se comporta "como si fuera un humano más", siendo el resultado

[1] https://multimatch.readthedocs.io/en/latest/

```
def computar_multimatch(dataset, resultados_modelo):
    '''se asume que resultados_modelo contiene una colección de
    trials donde cada trial es una tupla que almacena la imagen
    donde fue hecho el trial y el scanpath resultante.'''
    '''se asume que un dataset tiene por un lado las imágenes y por
    otro lado los trials, donde cada trial es una tupla que
    almacena identificador del sujeto, imagen del trial y
    el scanpath correspondiente.'''
    '''se asume que las imágenes en cada trial están dentro de las
    imágenes del dataset.'''

    for image in dataset:
        humanos = obtener_scanpaths_en_imagen(dataset,image)
        scanpath_modelo = \
        obtener_scanpath_en_imagen(resultados_modelo,image)
        for (primer_scanpath,segundo_scanpath) in humanos:
            if primer_scanpath != segundo_scanpath:
                cinco_métricas = MultiMatch(primer_scanpath,\
                segundo_scanpath)
                cuatro_métricas = descartar_duración(cinco_métricas)
                #la duración no fue medida por lo que es descartada
                promedio_métricas = promediar(cuatro_métricas)
                agregar(resultados_humanos, promedio_métricas)
        promedio_resultados_humanos = promediar(resultados_humanos)
        for scanpath in humanos:
            cinco_métricas = MultiMatch(scanpath_modelo,scanpath)
            cuatro_métricas = descartar_duración(cinco_métricas)
            #la duración no fue medida por lo que es descartada
            promedio_métricas = promediar(cuatro_métricas)
            agregar(resultados_humanos_y_modelo, promedio_métricas)
        promedio_resultados_humanos_y_modelo = \
        promediar(resultados_humanos_y_modelo)

        agregar(resultados_finales, ( \
        promedio_resultados_humanos, \
        promedio_resultados_humanos_y_modelo))
```

Listing 3: Computar MultiMatch entre *scanpaths* de humanos y modelo. [10, 11]

```
i = 0
fijaciones_calculadas = [fijacion_inicial]
#al principio solo poseo la fijacion inicial
while i < máximo_sacadas:
  i++
  fijacion_actual = fijaciones_calculadas[i]
  ...
  fijaciones_calculadas[i + 1] = siguiente_fijacion(search_model,...)
  ...
```

Listing 4: Computar fijaciones normalmente.

```
i = 0
fijaciones_calculadas = [fijaciones_humanas[0]]
#se asume que las fijaciones humanas
#forman parte del scanpath de un sujeto particular
máximo_sacadas = longitud(fijaciones_humanas)
while i < máximo_sacadas:
  fijacion_actual = fijaciones_humanas[i]
  ...
  fijaciones_calculadas[i + 1] = siguiente_fijacion(search_model,...)
  ...
calcular_métricas(fijaciones_calculadas,fijaciones_humanas)
```

Listing 5: Computar fijaciones al calcular Human Scanpath Prediction.

ideal que todos los puntos se encuentren sobre la diagonal.

Si bien estas métricas funcionan bien a la hora de comparar *scanpaths* entre sí, no son capaces de compararlos fijación a fijación. Para ello se implementó una forma de comparar *scanpaths* humanos con el del modelo (Ver listings 4 y 5), dada una imagen, basada en [18].

La idea es hacer que el modelo en cada paso reemplace su historia previa por aquella de un sujeto particular, y ver qué tanto difiere lo que calcula el modelo de lo que decidió el humano. En la implementación no se comparan los dos *scanpaths* per se, sino que se compara el scanpath humano con los mapas de probabilidad que generó el modelo (que es prácticamente el scanpath del modelo). Se recorren ambos *scanpaths* a la par fijación a fijación mientras se calculan el NSS y el AUC[2] utilizando como ground truth las coordenadas de la fijación humana actual. Luego, para resumir los resultados, por cada imagen y cada sujeto se promedian los resultados de las respectivas métricas obtenidos en sus fijaciones, y posteriormente para cada imagen se promedian los valores de cada sujeto. De forma tal que por cada imagen queda un valor de NSS y un valor de AUC.

Este enfoque requiere que el modelo sea provisto de un sujeto y luego se ejecute, con lo

[2] https://github.com/matthias-k/pysaliency/blob/master/pysaliency/metrics.py

```
{
    "search_model":"bayesian",
    "target_similarity":"correlation",
    "prior":"deepgaze",
    "max_saccades":15,
    "cell_size":32,
    "scale_factor":3,
    "additive_shift":4,
    "seed":1234,
    "norm_cdf_tolerance":0.001,
    "save_target_similarity_map": false
}
```

Listing 6: Ejemplo de formato JSON con los parámetros configurables de cIBSPy. Ésta es la configuración default, basada en cIBSMat.

cual requiere mucho poder de cómputo a diferencia de, por ejemplo, MultiMatch. Además, si se aplica sobre todo un *dataset*, el modelo ignora imágenes que el sujeto no haya visto.

## 3.2. Implementación en Python

### 3.2.1. Mejoras en el diseño

Se empezó el proceso con una visión top-down: se portea primero el main y luego función por función a medida que son llamadas durante el código.

A su vez se aplican varias refactorizaciones relacionadas con un enfoque orientado a objetos: crear clases para los objetos importantes dentro de cIBSPy (*Visibility Map*, *Prior*, *Target Similarity Map*, *Search Model*, *Grid*), renombrar variables y métodos (por ejemplo $p$ pasó a llamarse *posterior*, en tanto $a$ y $b$ pasaron a llamarse *scale factor* y *additive shift* respectivamente), parametrizar variables previamente hardcodeadas, delegar funcionalidades a los objetos correspondientes, eliminar código innecesario y utilizar polimorfismo. Los parámetros configurables de cIBSPy se deben guardar en un archivo con formato JSON en la carpeta *configs* (Ver listing 6).

Los componentes notables en el nuevo diseño son:

- Prior: Aporta la información inicial a cIBSPy. Se corresponde con el primer vistazo que realizan los humanos a la imagen, previo a la búsqueda.
- Visibility Map: También llamado detectabilidad en otros modelos [9], simula la definición de la imagen respecto de la distancia a la fóvea para cada fijación posible mediante una gaussiana 2D, teniendo en cuenta que la fóvea se sitúa en el centro de determinada fijación.
- Target Similarity Map: Llamado *Template Response* en el código original y basado en el W de Geisler [12], representa la similitud entre el objetivo y cada posición posible dentro de la imagen, condicionado por el *visibility map*.

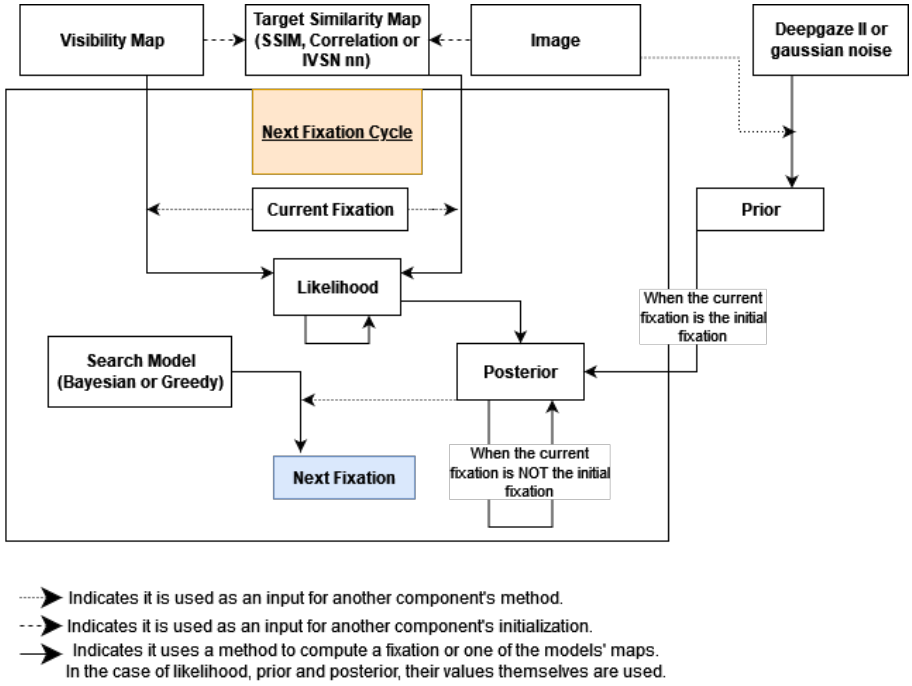


*Fig. 3.3:* Descripción de la interacción entre los distintos componentes a la hora de computar fijaciones a partir de una imagen.

- Search Model: Llamado Dynamic Model originalmente, es el motor que se encarga de determinar la próxima fijación basándose en el *posterior*.

Una característica que tienen en común todos estos componentes es que computan o utilizan mapas cuyos valores dependen directa o indirectamente de la imagen que se esté procesando actualmente. La imagen inicialmente es discretizada en base a un tamaño de fijación configurable, por lo que dichos mapas también deben ser discretizados utilizando el mismo procedimiento. Es por ello que se delegó la discretización a un objeto llamado Grid, no presente en la implementación MATLAB.

Otra característica que comparten esos componentes es que no hay una única forma de calcular sus mapas, por lo que se incorporó un protocolo a cada uno que se debe respetar, pero se facilita la incorporación y utilización de nuevos algoritmos para computarlos gracias al polimorfismo y la subclasificación.

Para la manipulación de vectores, matrices y operaciones matemáticas sobre ellos se utiliza Numpy[3].

A continuación se detalla cómo es la estructura del modelo implementado en Python y sus respectivos componentes (Ver Figura 3.3). Esencialmente, el modelo se inicializa tomando un archivo de configuración (Ver listing 6) y los parámetros de entrada que sirven para definir el modelo de búsqueda a utilizar, el mapa de visibilidad, el tipo de mapa de similiaridad y el algoritmo para calcular el *prior* (*Search Model*, *Visibility Map*, *Target Similarity Map*, y *Prior* respectivamente en Figura 3.3); estos dos últimos dependientes de la imagen de entrada y el *Target Similarity Map* dependiente también del *Visibility Map*.

[3] https://numpy.org/

Una vez hecho esto, el modelo comienza a computar el scanpath fijación a fijación mediante un ciclo que requiere en el primer paso una fijación inicial y un *prior* (*Next FIxation Cycle* en Figura 3.3). La fijación inicial la debe proveer el *dataset* correspondiente a la imagen. El *Visibility Map* y el *Target Similarity Map* son arreglos 4D que por sí solos no aportan mucho, pero al ser provistos de una fijación (la actualmente última), estos retornan los dos mapas correspondientes a ella y ambos son utilizados junto con el último *likelihood* obtenido (o un arreglo 2D lleno de 0s en el primer paso) para así calcular el *likelihood* de la siguiente fijación. El *Prior*, por su parte, es un arreglo 2D que una vez calculado se utiliza únicamente durante el primer paso del ciclo para computar, junto con el *likelihood* ya calculado, el *Posterior* de la siguiente fijación (otro arreglo 2D). Éste último hace las veces de *prior* en el paso siguiente del ciclo. Finalmente, el Search Model genera la siguiente fijación en base al *posterior*. El ciclo termina una vez que se encuentra el objetivo o que se alcanza el número máximo de fijaciones/sacadas.

## 3.3. Nuevos componentes y optimizaciones

### 3.3.1. Optimizaciones

A continuación se listan funcionalidades no presentes en cIBSMat, implementadas en cIBSPy.

1. Verificar si la fijación inicial cae dentro del objetivo.

2. Los *scanpaths* son descriptos en formato JSON (Ver listing 7).

3. La opción de poder correr el cIBSPy sobre un conjunto de imágenes o sobre una imagen particular.

4. La posibilidad de frenar la ejecución, guardando un checkpoint para poder resumirla en otro momento, particularmente importante cuando el *dataset* contiene una gran cantidad de imágenes.

5. La opción de guardar el *target similarity map* en una imagen (útil cuando se tarda bastante en calcularlos) dentro de la carpeta de data (similar a saliency) y la opción de cargarlo en caso de que haya sido guardado previamente. Debe establecerse el flag en el archivo de configuración (Ver listing 6).

6. Se incorporó el cálculo de *prior* a cIBSPy, adaptando una Jupyter Notebook que provee BETHGE LAB[4], que computa mapas de saliencia utilizando el center bias generado por TensorFlow[5]. Esto es una mejora ya que cIBSMat no calcula los *priors*. En particular, para las experimentaciones en su *dataset*, habían generado los mapas de saliencia para cada imagen con Deepgaze II [16] adhoc que posteriormente eran cargados por cIBSMat para ser usados a modo de *prior*.

[4] https://deepgaze.bethgelab.org/
[5] https://www.tensorflow.org/

```
{
    "grayscale_100_oliva.jpg": {
        "subject": "cIBS Model",
        "dataset": "Interiors Dataset",
        "max_fixations": 16,
        "receptive_height": 32,
        "receptive_width": 32,
        "image_height": 24,
        "image_width": 32,
        "target_found": true,
        "target_bbox": [
            11,
            22,
            14,
            24
        ],
        "X": [
            12,
            22
        ],
        "Y": [
            10,
            13
        ]
    }

    ...

}
```

Listing 7: Ejemplo de formato JSON utilizado en los *scanpaths* generados por cIBSPy.

7. Cambios en discretización de imágenes: El contenido de cada celda se puede tomar promediando los valores de los píxeles en cada celda o tomando el valor máximo. Por defecto se usa la primera opción. Si $\delta$ no divide a la resolución de la imagen:

   *a*) Enfoque viejo: se tiene en cuenta el centro de la imagen. Los bordes (que surgen del resto de dividir el tamaño de la imagen por $\delta$) se descartan.

   *b*) Enfoque nuevo: se mantienen los bordes en la cuadrícula y estos contienen el valor promedio de los píxeles de la imagen original que no llegan a completar una celda.

### 3.3.2. SSIM

Se decidió utilizar el algoritmo de structural similarity[6] (SSIM) [32] como un posible algoritmo a la hora de calcular el *Target Similarity Map*. La idea surgió a partir de pequeños experimentos hechos con imágenes a color utilizando correlación, cuyos resultados no fueron favorables. Como SSIM suele utilizarse para cuantificar ruido imágenes a color, entre otros usos, se implementó en el modelo mediante 3 enfoques distintos.

El primer enfoque consiste en centrar la imagen en el objetivo y a partir de allí dividir la imagen en una grilla cuyas celdas son del tamaño del objetivo. De ahí en más se calcula SSIM entre el objetivo y cada celda. Los bordes se manejan de forma especial ya que no necesariamente el objetivo va a dividir al tamaño de la imagen. Los píxeles que sobren en los bordes son comparados con la sección del objetivo que corresponda. El valor de SSIM se replica en todos los píxeles de la celda utilizada.

El segundo enfoque consiste en computar SSIM entre el objetivo y cada región posible de la imagen (cuyas dimensiones coincidan con las del objetivo). Posteriormente el resultado de SSIM se replica sobre todos los píxeles de la sección sobre la cual se hizo la comparación, por lo que queda más de un valor para cada pixel. Finalmente se promedia el valor obtenido en cada píxel, de forma tal que a cada uno le corresponde un único valor. Esta versión tarda bastante más por lo que se añadió paralelismo (multiprocessing, ya que es intensivo en el uso del CPU) para el cálculo. Por otro lado, se obtiene un mapa más definido que en el enfoque anterior. En lugar de promediar puede explorarse otro tipo de cálculo. A diferencia del primer enfoque, no hay problemas en los bordes.

El tercer enfoque difiere del segundo en la replicación del valor de SSIM: en este se almacena el resultado únicamente en el pixel del centro de la región correspondiente. Al hacer solamente este cambio, quedarían barras negras en los bordes de las imágenes, por lo que se realiza algo similar al primer enfoque: para obtener los valores de SSIM de los píxeles en los bordes se compara una región de la imagen más pequeña que el objetivo, con la sección del objetivo que corresponda.

Finalmente se optó por este último, y a esta versión del modelo se la llama sIBS (SSIM Ideal Bayesian Searcher).

### 3.3.3. Red de IVSN

En IVSN [4] se utiliza parte de la red VGG-16 [33] para capturar las características de alto nivel, es decir, estructuras complejas tanto de la imagen como del objetivo de búsqueda (Ver figura 3.4). Con esa información se genera un mapa de atención que es comparable al *Target Similarity Map* de cIBSPy. Teniendo esto en cuenta, se tomó dicho componente de IVSN para generar el mapa. Se espera que este algoritmo pueda capturar mejor la invarianza de los objetivo tal y como lo hace en IVSN. A ésta versión del modelo se la llama nnIBS (neural network Ideal Bayesian Searcher) (ver figura 3.5).

[6] https://scikit-image.org/docs/dev/auto_examples/transform/plot_ssim.html

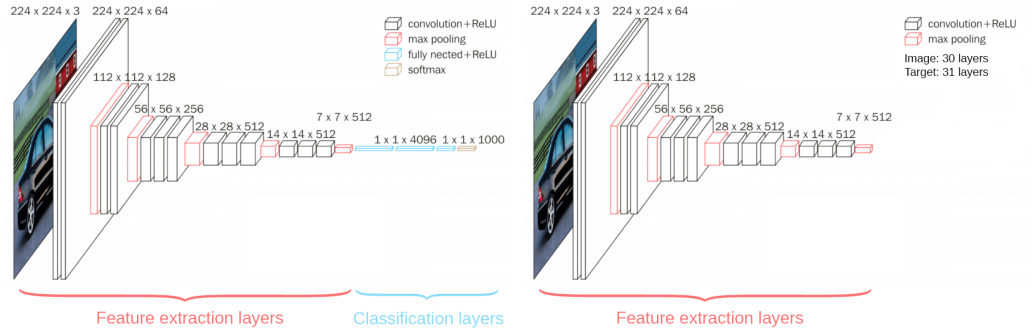


*Fig. 3.4:* A) VGG-16 B) VGG-16 sin las capas de clasificación, utilizado en IVSN [4] para computar mapa de probabilidad y en nnIBS para calcular el *Target Similarity Map*.

*Fig. 3.5:* A) Imagen grayscale_1_housebeautiful.png de Interiors con el objetivo marcado en rojo. B),C),D) *Target Similarity Maps* usando cIBS, sIBS y nnIBS respectivamente en la imagen grayscale_1_housebeautiful.png de Interiors.

En este proceso también fue necesario el pasaje de código escrito en LUA a Python, para no tener que ejecutar código adhoc. El script a grandes rasgos carga el modelo vgg-16 (desde un archivo adhoc), pasa la imagen (dividida en bloques cuyas dimensiones fueron definidas por los autores de IVSN) junto al objetivo por dicha red y luego realiza la convolución entre ambos outputs. El resultado es un conjunto de bloques de un mapa de atención que luego se unifican para construir el mapa completo. Utiliza torch para operar sobre imágenes y mapas en formato de tensores. Esta funcionalidad es en esencia la que se porteó a python con dos diferencias a destacar: se utilizaron pytorch en reemplazo

de torch y el modelo vgg-16 implementado en torchvision (no es necesario descargar un archivo adhoc). Ambas opciones fueron elegidas luego de una investigación y revisión de los paquetes disponibles en Python.

# 4. EXPERIMENTACIÓN Y RESULTADOS

Dentro de la experimentación el foco está puesto principalmente en las particularidades que tienen tanto el *prior* como los distintos *target similarity maps*.

## 4.1. Análisis de performance

Más allá de las cuestiones de diseño, hay dos objetivos intrínsecos dentro del porting:

1. Los resultados de cIBSPy deben ser lo más similares posibles a los arrojados por cIBSMat.

2. El tiempo de ejecución debe estar en un orden de magnitud igual o menor al previo.

Lo primordial era cumplir con el primer objetivo, por lo que fue el primer punto a verificar. Para hacerlo se compararon cIBSMat y cIBSPy contra los sujetos en el dataset Interiors. Se utilizaron dos métricas: Multimatch [10, 11] y rendimiento acumulado (Ver figura 4.1).

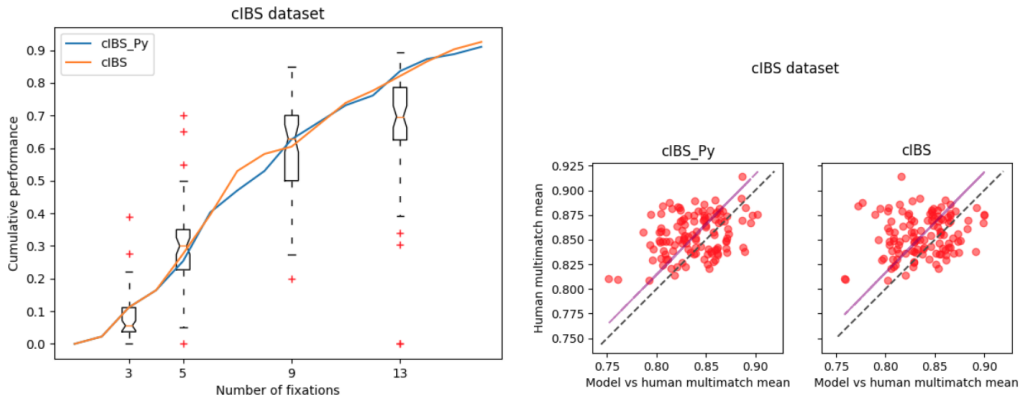


*Fig. 4.1:* A) Porcentaje de objetivos encontrados acumulado sobre la cantidad de fijaciones permitida para cIBSMat (cIBS en la figura), cIBSPy (cIBS_Py en la figura) y humanos (en boxplots estos últimos) en el *dataset Interiors*. B) MultiMatch para cIBSMat y cIBSPy donde cada punto corresponde a una imagen y sus coordenadas (x,y) están dadas por: el valor de MultiMatch promedio entre todos los pares posibles de sujetos que vieron esa imagen (coordenada y); el valor de MultiMatch promedio entre todos los pares posibles de (modelo, sujeto) con sujetos que vieron esa imagen (coordenada x). El valor de MultiMatch de cada par de sujetos o par (sujeto, modelo) se obtiene promediando entre los valores las cuatro métricas tenidas en cuenta (forma, longitud, dirección, posición).

En primera instancia, se observó que los resultados en cIBSPy no eran determinísticos producto de la semilla y el algoritmo utilizados para generar valores aleatorios, ya que diferían de lo hecho en cIBSMat. Luego de replicar ambos utilizando NumPy, se observó que los resultados entre ambas versiones eran prácticamente idénticos, por lo que se prosiguió a utilizar la versión en Python.

El segundo objetivo fue más complejo de alcanzar. Si bien ambas versiones corrían en tiempos similares, se podía aprovechar la paralelización para disminuir el tiempo del cómputo.

Se intentaron dos enfoques razonables:

1. Paralelizar el grueso del cómputo: *next_fixation* del Bayesian Search Model, que es el Search Model por defecto en cIBSMat [2].

2. Paralelizar el *dataset* (i.e.: dividir el procesamiento del *dataset* actual equitativamente entre cierto número de *workers*). Las pruebas se hicieron sobre el dataset Interiors.

Paralelizar el *dataset* es un enfoque análogo a MapReduce[1], por lo que es coherente. En cambio, pensar en paralelizar *next_fixation* no era tan directo. La razón por la cual pareció coherente paralelizar *next_fixation* es que consiste en cuatro bucles anidados ya que para cada posible próxima fijación se determina la probabilidad de que el objetivo esté en cada posición posible. En base a eso se genera un mapa probabilístico y se retorna la posición de la fijación que maximiza la probabilidad de encontrar el objetivo en el siguiente paso. La idea es que el proceso principal genere tantos workers como sean estipulados en el archivo de configuración, tomando como máximo el número de procesadores lógicos de la computadora usada, y que cada uno se ocupe de una parte equitativa del bucle cuádruple.

Como primera medida se probó threading[2] en *next_fixation* pero fue descartado rápidamente en ambos enfoques ya que se utiliza CPython como intérprete y este posee un Global Interpreter Lock que, entre otras cosas, no permite el cómputo paralelo entre threads [31]. La siguiente opción fue multiprocessing[3], que si bien tiene más overhead por la creación de nuevos procesos en lugar de threads, permite realizar cómputos en paralelo:

1. Primer enfoque: cIBSPy tardó 5827 segundos en terminar (1:37hs).

2. Segundo enfoque: cIBSPy tardó 7512 segundos en terminar (poco más de dos horas). También se intentó este enfoque usando Pandas[4] pero el rendimiento fue similar (7592 segundos).

Se optó por seguir con el primer enfoque en base a los resultados preliminares. Sin embargo, el rendimiento estaba lejos del de cIBSMat, que tardaba 1300 segundos aproximadamente (usando parfor). Intentando entender las marcadas diferencias en tiempos de ejecución se intentaron las siguientes opciones sin éxito:

- Usar threading, ya que aparentemente NumPy libera el GIL para la mayoría de sus operaciones. Sin embargo, dio resultados considerablemente peores.

- Usar comprensión de listas en lugar de bucles anidados.

- Investigando, se mencionaba que MATLAB puede estar optimizando el bucle mediante su compilador JIT[5] (Just in Time). Aún así, se probó PyPy (otro intérprete de Python que implementa un JIT) y los resultados fueron peores (probablemente porque PyPy utiliza una versión de cPython anterior).

[1] https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
[2] https://docs.python.org/3/library/threading.html
[3] https://docs.python.org/3/library/multiprocessing.html
[4] https://pandas.pydata.org/docs/
[5] https://www.mathworks.com/products/matlab/matlab-execution-engine.html

Analizando la función en sí, se midieron los tiempos de cada operación para intentar dilucidar dónde podía estar la diferencia. La interpolación, la integración y las cuentas intermedias se realizan con velocidad semejante en NumPy. La única diferencia notable es en la verificación de límites para integrar, donde se usa una matriz con booleanos para filtrar elementos. En MATLAB, esto lleva 0.00002 seg en promedio, mientras que a NumPy le toma 0.0003, un orden de magnitud más. El tiempo total de la función le toma 0.0007 seg. a Python, frente a 0.0004 de MATLAB.

La única mejora que se pudo aplicar fue cambiar dos try catch por dos if, bajando el tiempo total de 5827 segundos a 4495 segundos. Aún así, estaba lejos de MATLAB.

El problema no reside en la paralelización, ya que los incrementos en velocidad son los esperados: una sacada consume 130 segundos en Python sin utilizar múltiples núcleos. Al paralelizar, ese tiempo se reduce a 17 segundos (sobre una máquina con 12 núcleos, 6 físicos y 6 lógicos). En cambio MATLAB, sin paralelización, consume 40 segundos en una sacada. La eficiencia pasa por otro lado.

Se pudo reducir el tiempo de Python cambiando el código del método que calcula la integral, al hacer una verificación distinta para los valores de los límites de la misma. Este cambio se pudo realizar ya que el mapa de visibilidad siempre es positivo (en realidad, no-negativo), dado que es un muestreo de la normal acumulada, entonces se puede simplificar la cuenta. También se cambió la función de interpolación al aplicar una lineal, en lugar de nearest-neighbour y luego extrapolar, lo cual funcionaba más lento (agregaba 6 segundos al tiempo total de una sacada).

Siguiendo esta línea, se pudo aplicar otra mejora al no enmascarar los arreglos para filtrar elementos: basta con simplemente eliminar uno de ellos. Esto trajo mejoras sustanciales al rendimiento, llevando el tiempo de ejecución a uno muy similar a MATLAB (40 segundos por sacada vs. 55 segundos por sacada).

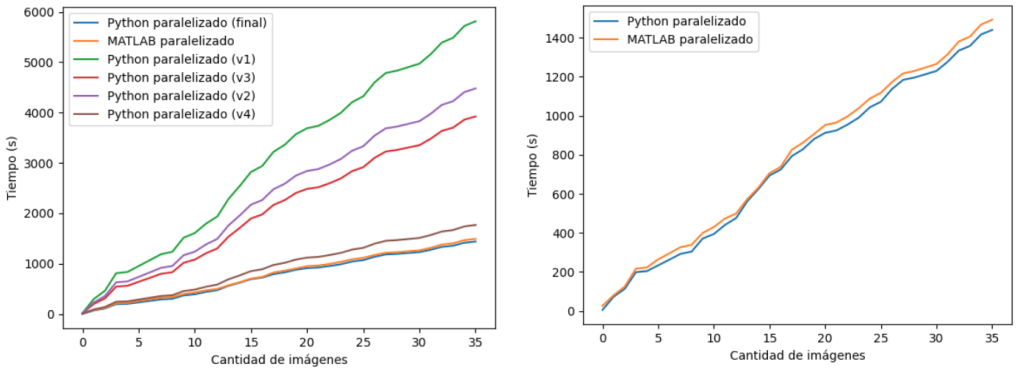


*Fig. 4.2:* A) Tiempo en milisegundos en función de cantidad de imágenes entre cIBSMat paralelizado (MATLAB paralelizado en la figura) y cIBSPy paralelizado (Python paralelizado en la figura) B) Tiempo en milisegundos en función de cantidad entre cIBSMat paralelizado y las distintas versiones de cIBSPy donde: (v1) corresponde a la versión inicial; de (v1) a (v2) se eliminó un try catch; de (v2) a (v3) se eliminó la verificación de toda la matriz y se cambió la función de interpolación; de (v3) a (v4) se descartó el uso de máscaras de NumPy; de (v4) a (final) se cambió el numpy.delete por un filtro con la matriz entera (vectorización).

Finalmente se reemplazó la utilización de numpy.delete para filtrar el elemento necesario por un filtro aplicado a la matriz entera (vectorización). En la versión final de cIBSPy

una sacada (sin paralelizar) toma 42 segundos en promedio, mientras que en cIBSMat toma 40 segundos. Al paralelizar ambas versiones la eficiencia de cIBSPy es aún más notoria, llegando incluso a superar la de cIBSMat (Ver figura 4.2). De aquí en más se asume que cIBS corresponde a cIBSPy.

## 4.2. Búsqueda de hiperparámetros: Variación de $\delta$

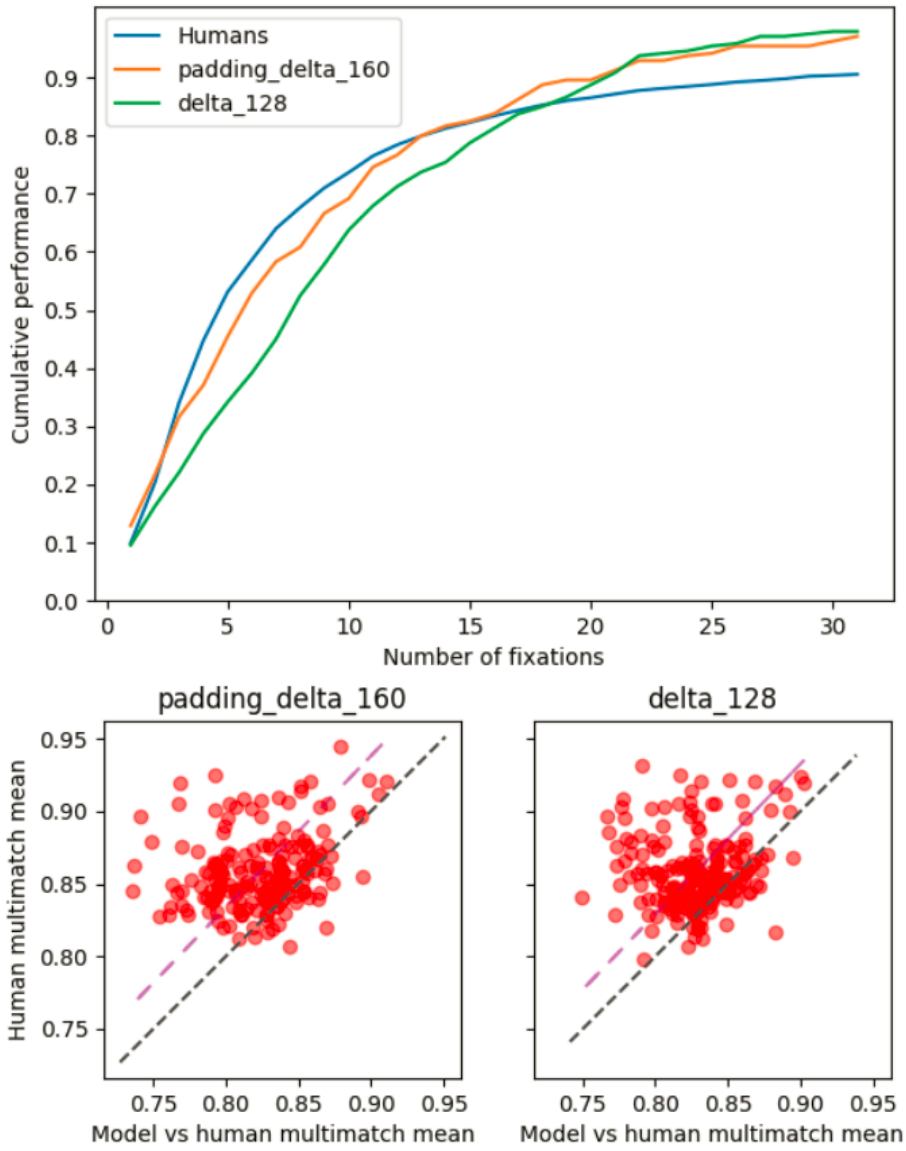


*Fig. 4.3:* A) Porcentaje de objetivos encontrados acumulado sobre la cantidad de fijaciones permitida para cIBS con delta = 160 (padding_delta_160 en la figura), cIBS con delta = 128 (delta_128 en la figura) y humanos (Humans en la figura) en el *dataset Interiors*. B) MultiMatch en *dataset Interiors* para cIBS con delta = 160 y con delta = 128 donde cada punto corresponde a una imagen y sus coordenadas (x,y) están dadas por: el valor de MultiMatch promedio entre todos los pares posibles de sujetos que vieron esa imagen (coordenada y); el valor de MultiMatch promedio entre todos los pares posibles de (modelo, sujeto) con sujetos que vieron esa imagen (coordenada x). El valor de MultiMatch de cada par de sujetos o par (sujeto, modelo) se obtiene promediando entre los valores las cuatro métricas tenidas en cuenta (forma, longitud, dirección, posición).

Una de las modificaciones al modelo fue cómo se manejan los bordes a la hora de reducir la imagen a una grilla (Ver sección 3.4.1). El valor estándar de $\delta$ de la publicación

es 32, que divide a las dimensiones de imagen permitidas por el modelo (1024x768) por lo cual en ese caso no es necesario tener ninguna consideración especial para los bordes [2]. Por lo tanto, se decidió probar con un $\delta$ que no divida ni al largo ni al ancho de las imágenes ($\delta = 160$) pero aprovechando, también se decidió probar con un $\delta$ que sí divida al largo y al ancho mas que sea mayor que 32 ($\delta = 128$). Para validar el diseño y la implementación del viejo esquema, se experimentó con el conjunto de datos Unrestricted usando cIBS con $\delta = 160$ y se observó lo siguiente:

1. producía inconvenientes en casos en los que el target se encontraba en alguno de esos bordes (cIBS salteaba la imagen directamente, 55 imágenes en total).

2. se notó que en algunos casos cIBS no salteaba la imagen, pero no encontraba al target y se quedaba buscando en la misma celda de la grilla una y otra vez (no aplicaba IoR, 16 imágenes aprox.). Este último comportamiento no sucedió en ningún caso con $\delta = 32$, ya que si bien el IoR no es aplicado explícitamente, el modelo lo tiene incorporado de forma implícita.

En cambio, al validar el nuevo enfoque se vio lo siguiente:

1. Con $\delta = 160$ cIBS pasó a no descartar ninguna imagen y a encontrar al objetivo en 234 de 240 imágenes.

2. También se probó agregar padding a la imagen (para que delta sea divisor) antes de procesarla, y de esta forma se encontró al objetivo en 232 de 240 (siempre con delta 160).

3. Con $\delta = 128$, cIBS encontró al objetivo en 235 de 240 imágenes.

También se analizaron los resultados con rendimiento acumulado y MultiMatch (Ver figura 4.4). Se espera que con ambos valores de $\delta$ el modelo requiera de pocas fijaciones para encontrar al objetivo, ya que el tamaño de la grilla es muy pequeño por lo que se requieren pocas fijaciones para recorrerla completamente. Por otro lado, al esperar una gran eficiencia en todas las imágenes y ya que a los humanos les fue generalmente bien sin necesidad de un gran número de fijaciones, los gráficos de MultiMatch deberían concentrar sus valores cerca de la diagonal (dando indicio de un comportamiento semejante al de los humanos). No obstante, por un lado se notó que el comportamiento del modelo está un tanto alejado del de los humanos (valores muy dispersos y alejados de la diagonal), y por otro lado se notó que si bien luego de muchas fijaciones la eficiencia es alta, hay que tener en cuenta que las fijaciones son grandes por lo que las grillas en las que se encuentran son bastante pequeñas en ambos casos, y en consecuencia la eficiencia habla más que nada de agotar casi todas las celdas de la grilla que de realizar una búsqueda eficiente aprovechando el gran tamaño de fijación.

Para analizar más en detalle el comportamiento del modelo, se optó por graficar *scanpaths* en los que no se encontró el objetivo ya que tienen el mayor número de fijaciones posible y por lo tanto son más proclives a presentar patrones distintivos. En principio se observó que luego de más de 15 fijaciones aproximadamente, se perdía el IoR. Además las búsquedas no parecían muy eficientes, parecía barrer prácticamente toda la imagen sin mucho sentido. Esto ocurrió tanto con $\delta = 160$ como con $\delta = 128$.


*Fig. 4.4:* Peor caso de *dataset* Unrestricted usando cIBS, donde las fijaciones 26, 27, 28, 29, 30 y 31 caen todas sobre el mismo lugar y la búsqueda no tiene coherencia.

## 4.3. Búsqueda de hiperparámetros: a y b

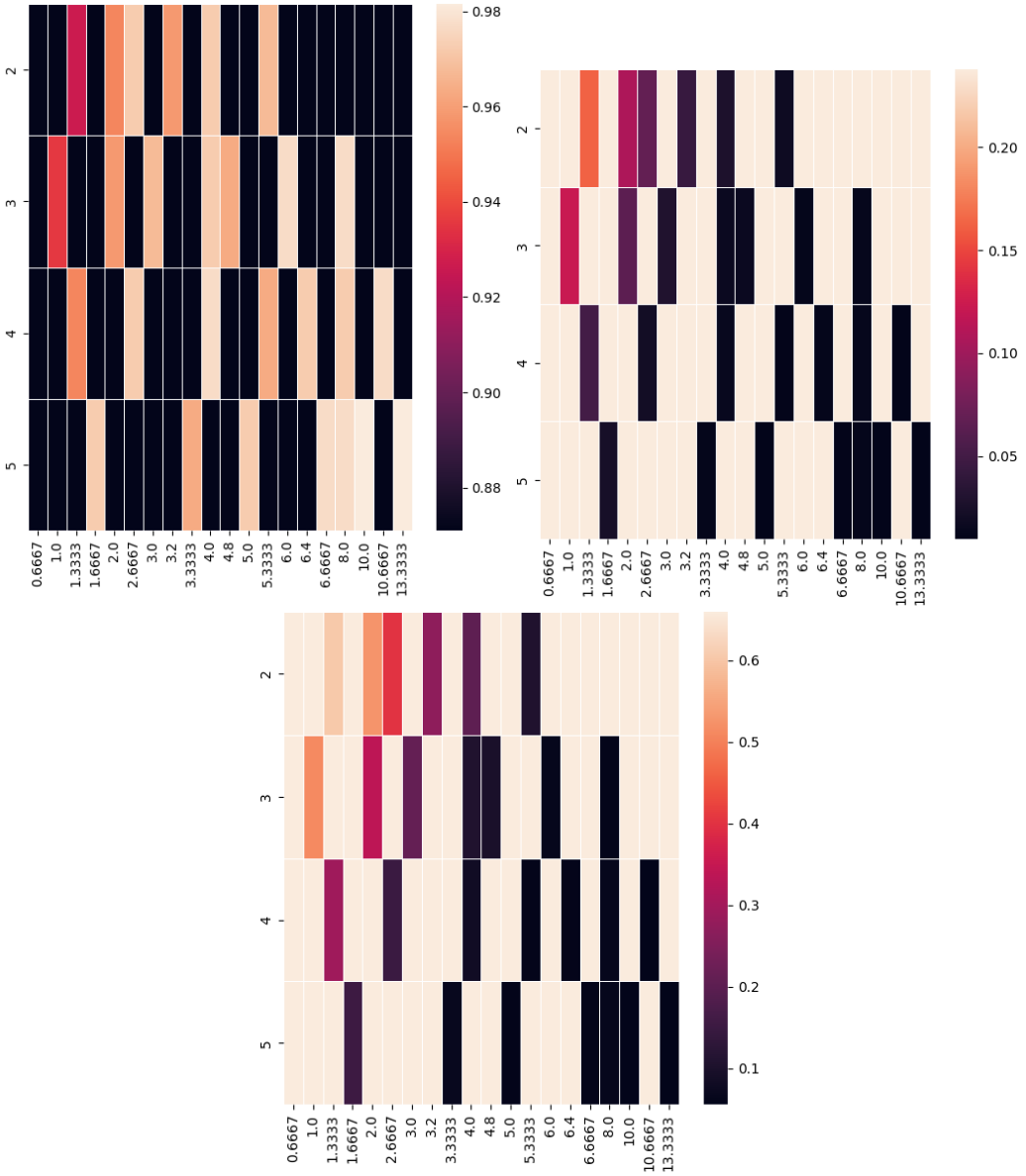


*Fig. 4.5:* A) % de objetivos encontrados. B) % de fijaciones consecutivas repetidas (en proporción a la cantidad total de fijaciones). C) % de *trials* que contienen fijaciones consecutivas repetidas. En los tres casos el eje y corresponde al hiperparámetro $a$ y el eje x al hiperparámetro $b$. A su vez los valores se calcularon utilizando cIBS en el *dataset Unrestricted*.

Intentando subsanar el problema de la inhibición de retorno con delta 128 en el *dataset Unrestricted*, se propuso buscar los hiperparámetros definidos en ecuación 1.6, $a$ y $b$, que mejoren los resultados. Se utilizó el modelo cIBS sobre el *dataset Unrestricted*. La lista de hiperparámetros buscados es la siguiente (donde el primer elemento de la tupla siempre corresponde al hiperparámetro $a$ y el segundo siempre corresponde al hiperparámetro $b$): [(2, 0.6667), (2, 1.3333), (2, 2), (2, 2.6667), (2, 3.2), (2, 4), (2, 5.3333), (3, 1), (3, 2), (3, 3), (3, 4), (3, 6), (3, 8), (3, 4.8), (4, 4), (4, 8), (4, 6.4), (4, 1.3333), (4, 2.6667), (4, 5.3333),

(4, 10.6667), (5, 5), (5, 8), (5, 10), (5, 1.6667), (5, 3.3333), (5, 6.6667), (5, 13.3333)]

Para los resultados se calcularon tres métricas 4.5: % de objetivos encontrados, % de fijaciones consecutivas repetidas (en proporción a la cantidad total de fijaciones) y % de *trials* que contienen fijaciones consecutivas repetidas.

Pareciera ser que cuanto más grandes los números, mejores los resultados. En comparación a la configuración por default (a = 3 y b = 4), la configuración óptima (a = 5 y b = 10) tiene la mitad de los *trials* con fijaciones consecutivas repetidas (5 % frente al 10 % del default), y una baja en la cantidad total de fijaciones consecutivas repetidas (0.99 % frente al 1.6 %). Por otro lado, la pérdida del IoR en la imagen anterior sigue ocurriendo, con la salvedad de que las fijaciones 22 y 23 ya no se repiten:

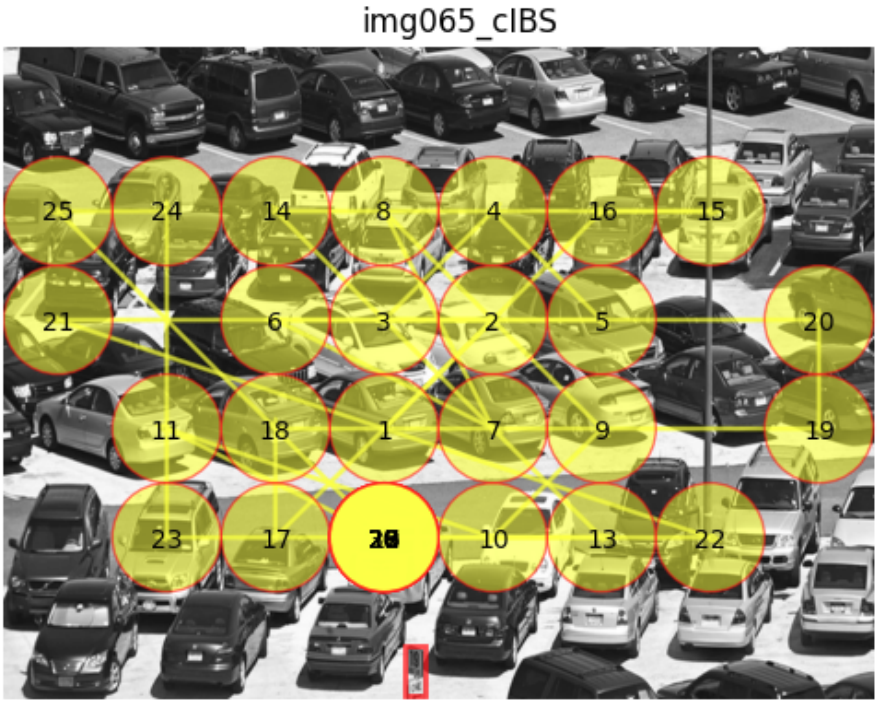


*Fig. 4.6:* Peor caso del *dataset Unrestricted* usando cIBS donde $\delta = 128$ con a y b óptimos (5 y 10 respectivamente), aún con pérdida de IoR y búsqueda incoherente.

El modelo resulta ser muy sensible al tamaño de la grilla. Al tomar valores más grandes de $\delta$, se produjeron comportamientos no deseables en el modelo. Por los resultados obtenidos, a mayor tamaño de la grilla, se pierde granularidad en la información obtenida en cada posición de la misma, lo que podría estar provocando esta pérdida de la inhibición de retorno. Sin embargo, esta hipótesis no fue puesta a prueba aún y se plantea como trabajo a futuro. Dados los resultados obtenidos en el primer trabajo [2] se decidió mantener el mismo valor de hiperparámetros (a = 3, b = 4, $\delta = 32$).

## 4.4. Comparación entre Target Similarity Maps

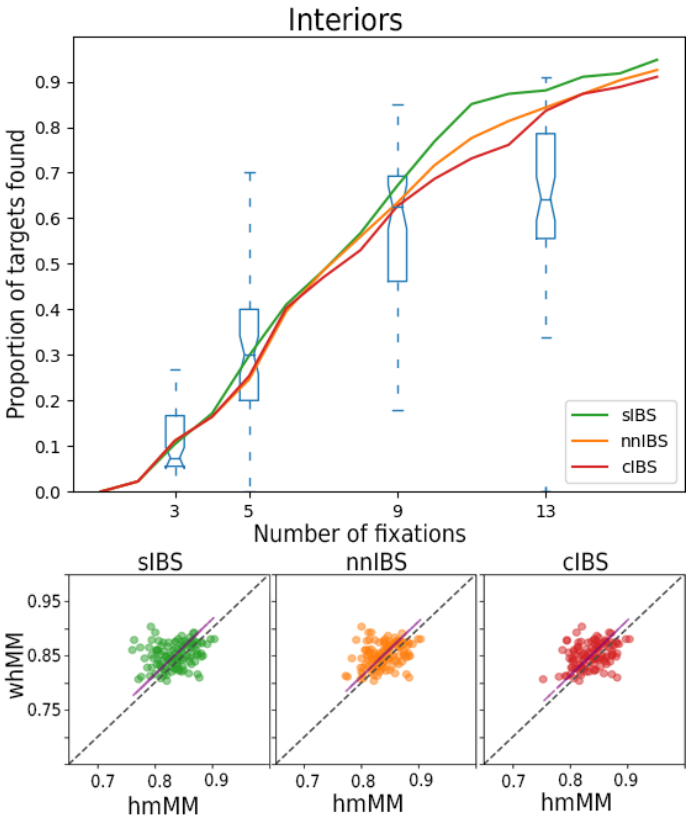


*Fig. 4.7:* A) Porcentaje de objetivos encontrados acumulado sobre la cantidad de fijaciones permitida para cIBS (rojo), sIBS (verde), nnIBS (naranja) y humanos (boxplots en la figura) en el *dataset Interiors*. B) MultiMatch en *dataset Interiors* para cIBS, sIBS y nnIBS donde cada punto corresponde a una imagen y sus coordenadas (x,y) están dadas por: el valor de MultiMatch promedio entre todos los pares posibles de sujetos que vieron esa imagen (coordenada y); el valor de MultiMatch promedio entre todos los pares posibles de (modelo, sujeto) con sujetos que vieron esa imagen (coordenada x). El valor de MultiMatch de cada par de sujetos o par (sujeto, modelo) se obtiene promediando entre los valores las cuatro métricas tenidas en cuenta (forma, longitud, dirección, posición).

Una de las mejoras implementadas fue la de incluir diferentes *target similarity maps* (ver secciones 3.3.1, 3.4.2 y 3.4.3). Para validar sus implementaciones y medir sus respectivas eficiencias, se realizaron experimentos utilizando las tres versiones disponibles (cIBS, sIBS, nnIBS) sobre los cuatro *datasets* a disposición (Interiors, Unrestricted, COCOSearch-18, MCS).

El objetivo del experimento es encontrar un *Target Similarity Map* que sea eficiente tanto en imágenes en escala de grises como en color, y que sea capaz de generalizar al objetivo a otras instancias de la misma categoría. Teniendo en cuenta que los experimentos con humanos fueron hechos en escala de grises tanto en Interiors como en Unrestricted, se respetó esa condición para ambos *datasets*. Por su parte en COCOSearch-18 y MCS los experimentos se condujeron en imágenes a color, por lo que en estos casos el modelo también hace lo propio.

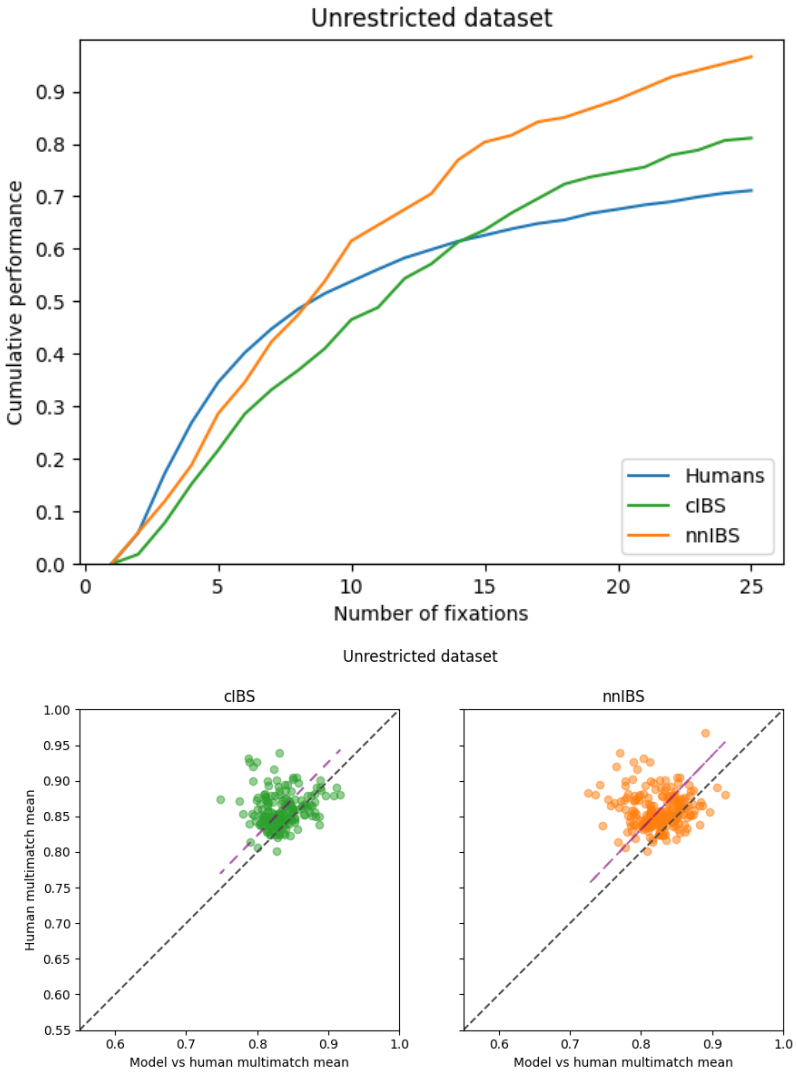


*Fig. 4.8:* A) Porcentaje de objetivos encontrados acumulado sobre la cantidad de fijaciones permitida para cIBS (verde), nnIBS (naranja) y humanos (azul) en el *dataset Unrestricted*. B) MultiMatch en *dataset Unrestricted* para cIBS y nnIBS donde cada punto corresponde a una imagen y sus coordenadas (x,y) están dadas por: el valor de MultiMatch promedio entre todos los pares posibles de sujetos que vieron esa imagen (coordenada y); el valor de MultiMatch promedio entre todos los pares posibles de (modelo, sujeto) con sujetos que vieron esa imagen (coordenada x). El valor de MultiMatch de cada par de sujetos o par (sujeto, modelo) se obtiene promediando entre los valores las cuatro métricas tenidas en cuenta (forma, longitud, dirección, posición).

Por otro lado, en MCS y en COCOSearch-18 solo se les indicó el nombre de la categoría a las cuales pertenecían los objetivos (por ej. un reloj), pero como el modelo de momento es incapaz de trabajar de esa forma, en MCS y en COCOSearch-18 se usaron recortes de los objetivos dentro de las imágenes. En cambio en Interiors los objetivos fueron presentados a los humanos utilizando recortes de las imágenes, mientras que en Unrestricted se les presentaron diferentes instancias del mismo tipo de objeto, lo cual fue respetado por el

modelo también.

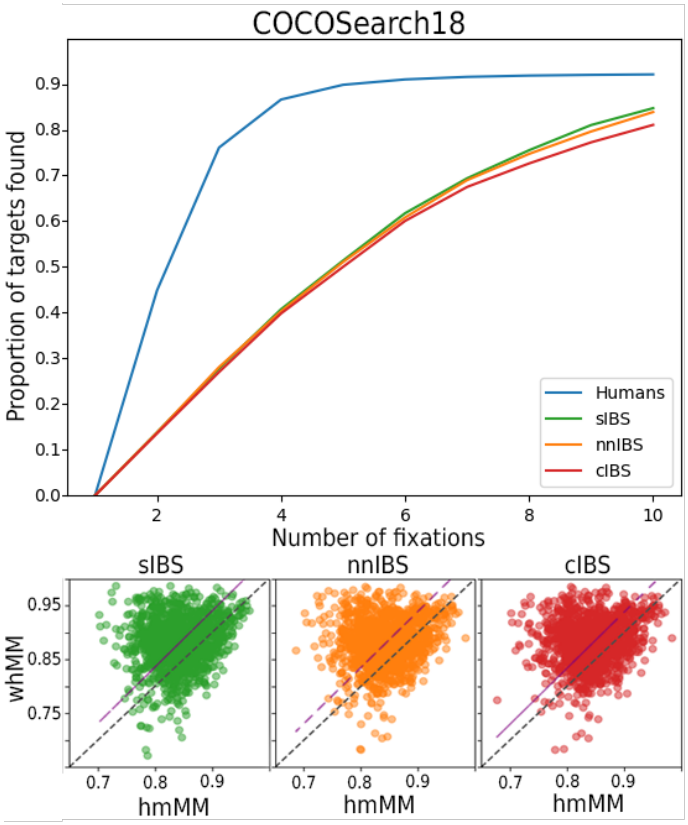


*Fig. 4.9:* A) Porcentaje de objetivos encontrados acumulado sobre la cantidad de fijaciones permitida para cIBS (rojo), sIBS (verde), nnIBS (naranja) y humanos (azul) en el *dataset COCOSearch-18*. B) MultiMatch en *dataset COCOSearch-18* para cIBS, sIBS y nnIBS donde cada punto corresponde a una imagen y sus coordenadas (x,y) están dadas por: el valor de MultiMatch promedio entre todos los pares posibles de sujetos que vieron esa imagen (coordenada y); el valor de MultiMatch promedio entre todos los pares posibles de (modelo, sujeto) con sujetos que vieron esa imagen (coordenada x). El valor de MultiMatch de cada par de sujetos o par (sujeto, modelo) se obtiene promediando entre los valores las cuatro métricas tenidas en cuenta (forma, longitud, dirección, posición).

Para esta experimentación se estableció el máximo de sacadas que puede realizar el modelo en cada *dataset* según la longitud promedio de sus respectivos *scanpaths* humanos. En todos los casos se evaluó tanto la eficiencia acumulada como MultiMatch. Se busca por un lado que los objetivos se encuentren en la menor cantidad posible de fijaciones, y por otro lado que los valores de MultiMatch se concentren cerca o sobre la diagonal (indicando que los *scanpaths* generados por los modelos son similares a aquellos de los humanos). Finalmente a modo de resumen se reporta para cada versión de IBS y sobre cada *dataset*, el área bajo la curva (AUC) de la eficiencia acumulada y el valor de MultiMatch (de la misma forma que fue descrito en 3.2) promediado sobre todo el *dataset* junto con la correlación correspondiente (ver tabla S1). Asimismo, se reporta para nnIBS y sobre cada *dataset*, el NSS y el AUC (como fueron descritos en 3.2) promediados sobre todo el *dataset* (ver tabla S2).

Comenzando por Interiors (ver figura 4.7), se puede observar que las 3 versiones fun-

cionan realmente bien. En particular sIBS funciona un poco mejor, pero tarda aproximadamente 8 minutos (el tiempo aumenta si el objetivo es grande) en computar el *Target Similarity Map* de una imagen mientras que cIBS y nnIBS tardan 4 segundos en hacer lo propio y según los resultados de MultiMatch hay menos concentración de valores en la diagonal, por lo que su comportamiento se asemeja un poco menos al de los humanos que el de las otras dos versiones.

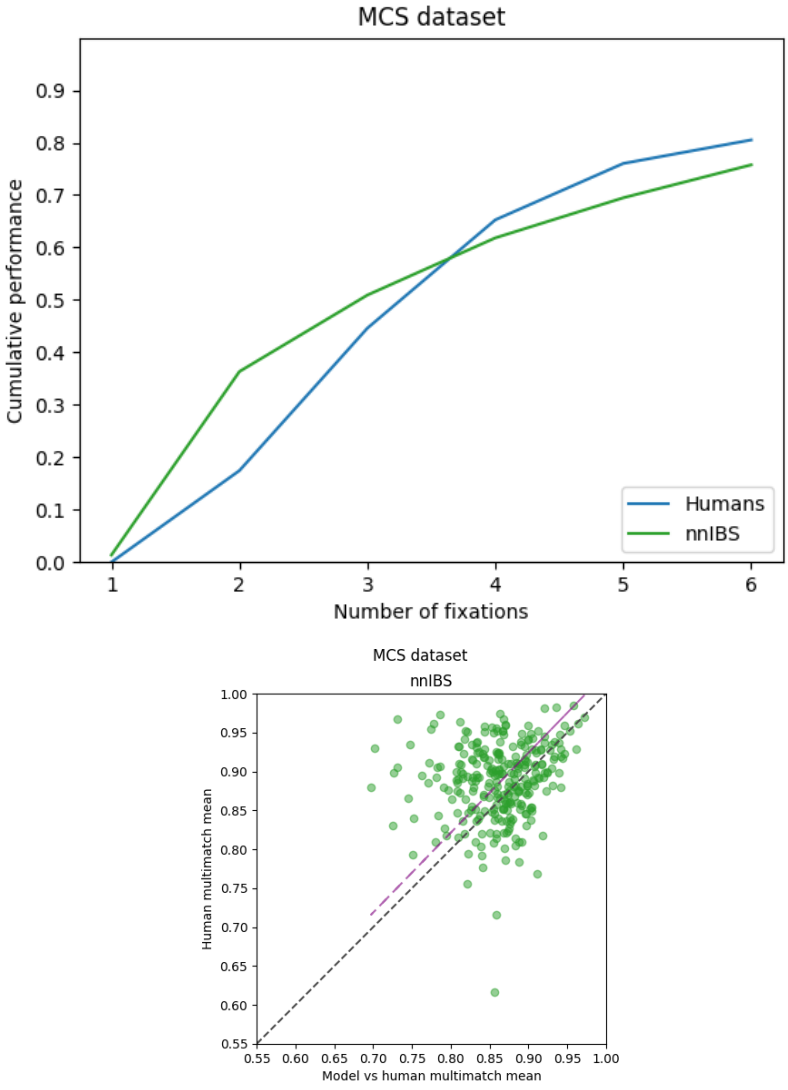


*Fig. 4.10:* A) Porcentaje de objetivos encontrados acumulado sobre la cantidad de fijaciones permitida para nnIBS (verde) y humanos (azul) en el *dataset MCS*. B) MultiMatch en *dataset MCS* para nnIBS donde cada punto corresponde a una imagen y sus coordenadas (x,y) están dadas por: el valor de MultiMatch promedio entre todos los pares posibles de sujetos que vieron esa imagen (coordenada y); el valor de MultiMatch promedio entre todos los pares posibles de (modelo, sujeto) con sujetos que vieron esa imagen (coordenada x). El valor de MultiMatch de cada par de sujetos o par (sujeto, modelo) se obtiene promediando entre los valores las cuatro métricas tenidas en cuenta (forma, longitud, dirección, posición).

Al analizar los resultados obtenidos en COCOSearch-18 (ver figura 4.9) ocurren fenómenos interesantes. En principio las tres versiones se adaptan prácticamente de igual manera a imágenes a color. No obstante, en el gráfico de MultiMatch los valores están bastante dispersos, lo cual indica que los *scanpaths* generados por el modelo (en las tres versiones) son en general bastante distintos a los de los sujetos. Además parece ser que requieren, en proporción, bastantes más fijaciones que los humanos para encontrar el objetivo. Por ende, las imágenes a color perjudican al modelo y en mayor medida si se permiten pocas fijaciones.

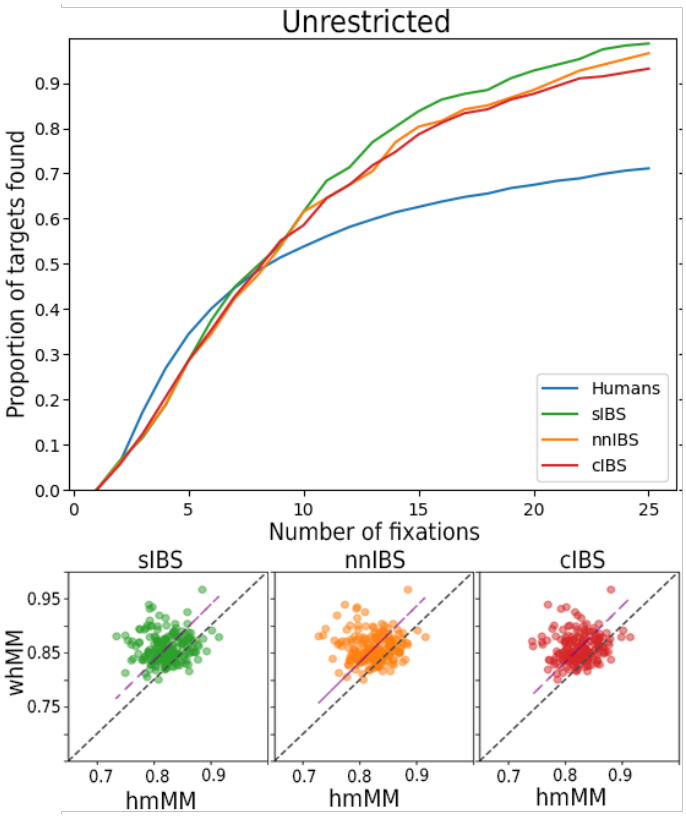


*Fig. 4.11:* A) Porcentaje de objetivos encontrados acumulado sobre la cantidad de fijaciones permitida para cIBS usando recorte de la imagen como objetivo (rojo), sIBS usando recorte de la imagen como objetivo (verde), nnIBS usando objetivo categórico del *dataset* (naranja) y humanos (azul) en el *dataset* Unrestricted. B) MultiMatch en *dataset* Unrestricted para cIBS usando recorte de la imagen como objetivo, sIBS usando recorte de la imagen como objetivo y nnIBS usando objetivo categórico del *dataset* donde cada punto corresponde a una imagen y sus coordenadas (x,y) están dadas por: el valor de MultiMatch promedio entre todos los pares posibles de sujetos que vieron esa imagen (coordenada y); el valor de MultiMatch promedio entre todos los pares posibles de (modelo, sujeto) con sujetos que vieron esa imagen (coordenada x). El valor de MultiMatch de cada par de sujetos o par (sujeto, modelo) se obtiene promediando entre los valores las cuatro métricas tenidas en cuenta (forma, longitud, dirección, posición). Se puede apreciar como nnIBS se mantiene a la par de las otras dos versiones tanto en concentración de valores en la diagonal en B) como en eficiencia en A), a pesar de no conocer el objetivo específico.

Por su parte, en MCS (ver figura 4.10) por limitaciones de tiempo y de poder de cómputo, no se pudo correr cIBS y sIBS sobre él (principalmente sIBS que es bastante

*Tab. S1:* Área bajo la curva (AUC) de la eficiencia acumulada (AUCPerf) y Multimatch (AvgMM) junto con su correlación (Corr) para las tres versiones de IBS en los distintos *datasets*.

| | AUCperf | Corr | AvgMM |
|---|---|---|---|
| **Interiors** | | | |
| nnIBS | 0.53 | **0.28** | **0.84** |
| sIBS | **0.56** | 0.12 | 0.83 |
| cIBS | 0.52 | 0.24 | **0.84** |
| Humans | 0.42 | - | 0.85 |
| **Unrestricted** | | | |
| nnIBS | 0.63 | 0.017 | **0.82** |
| sIBS | **0.65** | 0.09 | **0.82** |
| cIBS | 0.62 | **0.10** | **0.82** |
| Humans | 0.56 | - | 0.86 |
| **COCOSearch18** | | | |
| nnIBS | **0.51** | 0.14 | **0.84** |
| sIBS | **0.51** | **0.16** | **0.84** |
| cIBS | 0.50 | **0.16** | **0.84** |
| Humans | 0.78 | - | 0.89 |
| **MCS** | | | |
| nnIBS | **0.51** | **0.14** | **0.86** |
| Humans | 0.49 | - | 0.89 |

demandante), no hay resultados de estas versiones sobre este *dataset*. No obstante, al tener bastantes inconvenientes (descritos en sección 3.1.4) y siendo que ya se hizo un experimento con un *dataset* con imágenes a color, se vuelve complicado extraer resultados útiles de él.

En Unrestricted (ver figura 4.8) tanto cIBS como sIBS utilizando otras instancias del mismo tipo de objeto funcionan bastante mal, al punto de encontrar el objetivo en poco menos del 70 % de las imágenes (en el caso de sIBS se perdieron los datos necesarios para generar los gráficos, pero quedó el porcentaje de las imágenes encontradas), mientras que nnIBS alcanza una mejor eficiencia en toda cantidad de fijaciones permitida. No obstante, vale destacar que en el gráfico de MultiMatch en nnIBS los valores no están tan concentrados en la diagonal como ocurre con cIBS, por lo que los *scanpaths* de esta versión del modelo se asemejan a los de los humanos menos de lo que lo hacen aquellos generados por cIBS ya que a los humanos también les costó el experimento en este *dataset*.

En base a estos resultados, el que resultó ser una mejora sustancial frente a las otras opciones es nnIBS ya que esta versión, a diferencia de las otras dos, es capaz de generalizar el objetivo a otras instancias del mismo tipo de objeto. Para ilustrarlo, se compararon la

eficiencia y los valores de MultiMatch en nnIBS utilizando los objetivos categóricos del *dataset* frente a los valores obtenidos en sIBS y cIBS que por su parte usaron recortes de las imágenes a modo de objetivos (ver figura 4.11). Se puede apreciar como nnIBS se mantiene a la par de las otras dos versiones tanto en MultiMatch como en eficiencia acumulada, a pesar de no conocer al objetivo específico.

*Tab. S2:* AUC y NSS promedios (avgAUC y avgNSS respectivamente) calculados según sección 3.2 para nnIBS en los distintos *datasets*.

| Dataset | avgAUC | avgNSS |
|---|---|---|
| Interiors | 0.69 | 0.77 |
| Unrestricted | 0.72 | 0.94 |
| COCOSearch18 | 0.76 | 1.01 |
| MCS | 0.78 | 1.81 |

## 4.5. Efecto de *prior* en fotos con caras

Hoy en día hay estudios que indican que los humanos estamos programados para buscar caras [38]. En algunos casos vemos caras donde no las hay (pareidolia de caras) [39]. En principio esto no tiene relación directa con este trabajo, pero luego de mirar *scanpaths* particulares durante experimentos con este modelo se empezó a notar un patrón en imágenes del *dataset* Unrestricted (el único de los *datasets* que posee caras humanas) (ver figura 4.12): las primeras fijaciones del modelo iban hacia caras humanas aún cuando la tarea consiste en buscar objetos concretos sin relación a las mismas. No obstante, los sujetos que participaron en los experimentos con ese *dataset* no siguieron este patrón. Desde el inicio dirigen su atención hacia regiones de la imagen donde podría encontrarse el objetivo, ignorando casi completamente a las caras. Podría decirse que estamos programados para buscar caras excepto cuando hay una tarea específica de por medio. Si bien esto último ameritaría un estudio particular, la pregunta de interés es ¿Por qué los humanos no dirigen la atención a las caras mientras que el modelo de estudio sí?

Observando con mayor detalle, el patrón se da principalmente en las primeras fijaciones, que es cuando el *prior* tiene una mayor preponderancia. Como este es calculado utilizando Deepgaze II, que fue entrenado en tareas de *free-viewing*, es de esperarse que los humanos en dichos experimentos se hayan enfocado bastante en las caras. Y por lo tanto que el modelo de saliencia haya captado ese fenómeno y quede plasmado en los mapas de saliencia que computa.

Esto no es algo menor, ya que denota que hay margen para mejorar el modelo de estudio: el *prior* no debería darle tanta relevancia a las caras humanas cuando la tarea es buscar objetos no relacionados a ellas.

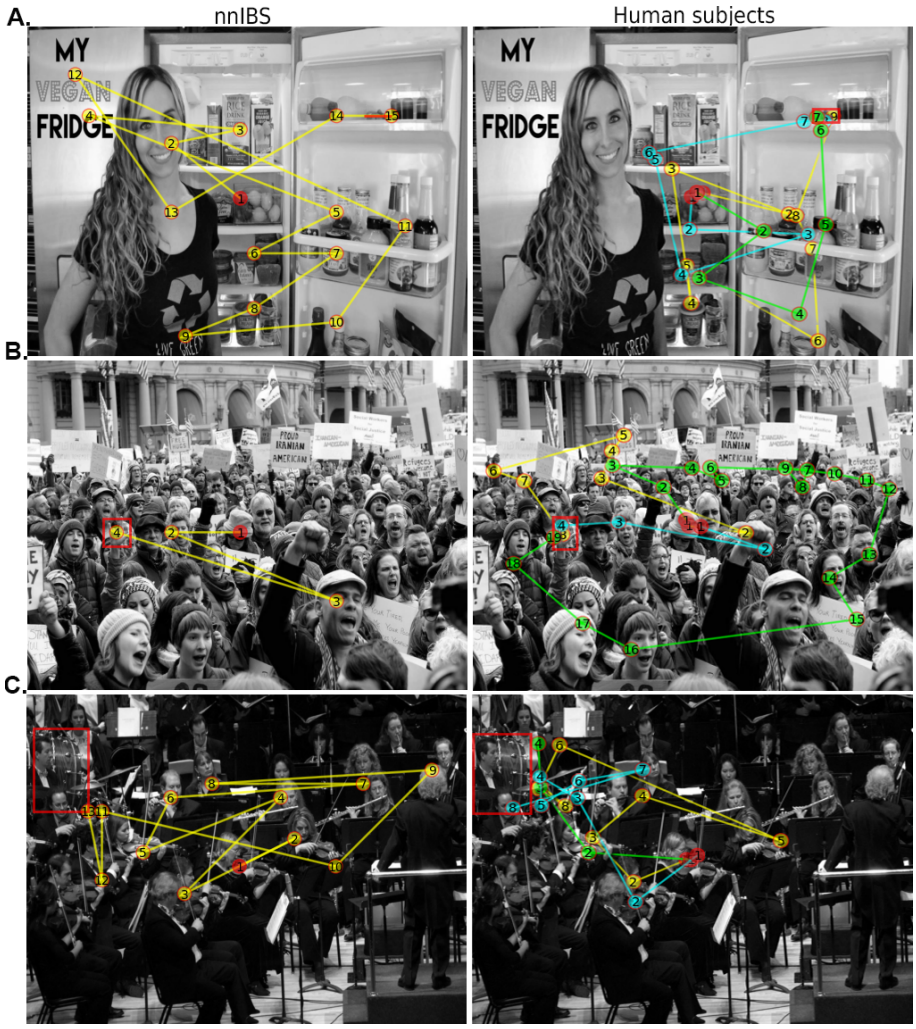


*Fig. 4.12:* Diferencias entre patrón del modelo y patrones humanos en primeras fijaciones de *scanpaths* en imágenes con caras del *dataset* Unrestricted.

## 4.6. Mejores y peores casos de nnIBS

El objetivo de este experimento es explorar posibles mejoras a futuro del modelo. Para lo cual se presentarán scanpaths de varios datasets en los que le fue muy bien al modelo y otros en los que le fue muy mal, y en base a ello, buscar características o patrones en las imágenes. Se debe tener en cuenta que las imágenes, y el objetivo en cada una de ellas, fueron redimensionadas para que pueda procesarlas el modelo, por lo que es posible que la relación de aspecto cambie respecto de la imagen original.

Por un lado se tendrá en cuenta la eficiencia (i.e.: encontrar el objetivo en pocas fijaciones o no encontrar el objetivo) y por otro lado la similitud con scanpaths humanos basándose en las métricas de MultiMatch, descartando la duración. Para este último escenario las imágenes cuyo punto dentro del gráfico de MultiMatch se encuentre cerca de la diagonal indica que, en los trials llevados a cabo en dicha imagen, el modelo generó un

scanpath similar a los generados por los humanos mientras que si el punto se encuentra alejado de la diagonal es indicio de que el modelo generó un scanpath distinto al que generaron los humanos. Por otra parte, los puntos que se encuentran debajo de la diagonal indican que los scanpaths humanos no eran muy similares entre sí en primer lugar, por lo que interesan únicamente puntos por encima de la diagonal para este experimento.

Tanto en el *dataset Unrestricted* como en *COCOSearch-18*, puede observarse que en caso de haber comida presente, el modelo dirige sus primeras fijaciones hacia esos sectores aunque no están relacionadas con el objetivo (ver figuras 5.1, 5.2); mientras que los sujetos directamente ignoran la comida. Esto hace pensar que al igual que sucedía con las caras, el prior esté sesgando la atención del modelo hacia sectores con comida.

Asimismo, sucede un fenómeno similar en imágenes donde hay letras o palabras (o logos que pueden interpretarse como letras o palabras)(ver figuras 5.3, 5.4). Hay imágenes de estas características presentes en los *datasets Unrestricted*, *MCS* y *COCOSearch-18*.

Otros trials interesantes de analizar son aquellos en los que la información de contexto es clave. Por ejemplo, al buscar un edificio los humanos tienden a dirigir su atención a regiones superiores de la imagen pero el modelo no necesariamente hace lo mismo(ver figuras 5.5). Otro ejemplo ocurre al buscar recipientes como bowls o botellas, que suelen estar sobre una mesa o mesada y generalmente cerca del centro de la imagen, sin embargo el modelo no necesariamente busca en esos lugares(ver figuras 5.6, 5.7). No obstante, hay trials en los que el modelo parece estar aprovechando información bastante compleja(ver figuras 5.8). Estos tipos de trials están presentes en los cuatro *datasets*.

En cambio, cuando hay muchos distractores presentes en las imágenes, la dificultad es alta tanto para los humanos como para el modelo, por lo que en ambos casos hay un gran número de fijaciones(ver figuras 5.9). En este tipo de imágenes es común que los valores de MultiMatch correspondientes se concentren en la diagonal.

Otros trials interesantes se dieron en el *dataset Unrestricted*, ya que hubo trials en los que el modelo parece haber perdido la propiedad de inhibición de retorno(ver figuras 5.10). Por ejemplo, en uno de esos trials hay muchos logos en la imagen, mientras que en otros el objetivo es muy pequeño, lo cual puede estar afectando el funcionamiento del modelo, teniendo en cuenta que trabaja con una grilla.

A su vez, hay muchos trials en los que el modelo hace sacadas muy largas(ver figuras 5.11, 5.12), lo cual no es algo que suelen hacer los humanos.

Cabe destacar que en aquellos trials en los que el objetivo es muy grande y/o se encuentra cerca del centro de la imagen, el modelo presenta una gran eficiencia, muchas veces completando la tarea en 1 o 2 sacadas (ver figuras 5.13, 5.14). Si bien este tipo de trial puede ser catalogado como "fácil", es importante que el modelo pueda aprovechar esa facilidad. Finalmente, hubo trials difíciles de explicar desde el punto de vista de los *scanpaths* generados por el modelo:

- En img149 del *dataset Unrestricted* no queda del todo claro por qué el modelo fue directamente al objetivo (ver figura 5.15).
- En img161 del *dataset Unrestricted* el modelo parece haberse distraído por varios objetos dentro de la imagen, pero no ocurrió lo mismo con los humanos (ver figura 5.16).
- En grayscale_4_kitchen del *dataset Interiors* el modelo busca en muchos sitios donde generalmente estaría un jarrón. Sería interesante conocer qué tanto ayuda la vista periférica a los sujetos en ese caso (ver figura 5.17).

- En grayscale_45_oliva del *dataset Interiors* el modelo sesga su búsqueda a una región pequeña de la imagen (ver figura 5.18).

- En grayscale_100_oliva y grayscale_12_housebeautiful de Interiors donde el modelo va directamente al objetivo siendo que en ningún caso este es grande ni está cerca del centro(ver figuras 5.19).

- En grayscale_10_housebeautiful de Interiors el modelo tiene dificultades que los sujetos no (ver figura 5.20).

A futuro se prestará especial atención a estos casos, y se evaluará si agregando variabilidad al modelo es posible obtener otras respuestas.

## 4.7. Poniendo al límite a nnIBS

La idea de este experimento es aplicar distintos tipos de ruido a un subconjunto de imágenes del *dataset Interiors*, y ver qué tan robusto se mantiene el modelo ante ellos. El criterio para elegir el subconjunto es tomar imágenes en las que el modelo encontró el objetivo en 4 a 6 fijaciones inclusive, a la vez que una cantidad mayor a 10 sujetos hayan encontrado el objetivo en dichas imágenes.

Se eligieron cuatro tipos de ruido: "*gaussian*", "*speckle*", "*salt&pepper*", "*poisson*". Para los tres primeros a su vez se incorporaron 5 niveles de ruido, ya sea aumentando la varianza en el caso de "*gaussian*" (en intervalos de 0.02) y "*speckle*" (en intervalos de 0.05) o aumentando la cantidad de puntos en "*salt & pepper*" (en intervalos de 0.05). Todos ellos fueron aplicados directamente a las imágenes utilizando *random_noise* del módulo *util* del paquete *skimage*[6].

En promedio, el tipo o el nivel de ruido no parecen afectar el desempeño del modelo (ver figura 4.13). Sin embargo, para cada imagen particular se puede observar un impacto diferente del ruido, requiriendo tanto mayores como menores cantidades de fijaciones para encontrar el target.

Lo siguiente fue intentar encontrar un nivel de ruido para el que haya una diferencia muy marcada en todas las imágenes, para ello se hizo una búsqueda similar a la anterior aumentado los niveles de ruido en cada intervalo de la siguiente forma:

- 0.1 para "gaussian"

- 1.0 para "speckle"

- 0.1 para "salt & pepper"

[6] https://scikit-image.org/docs/dev/api/skimage.util.html#skimage.util.random_noise

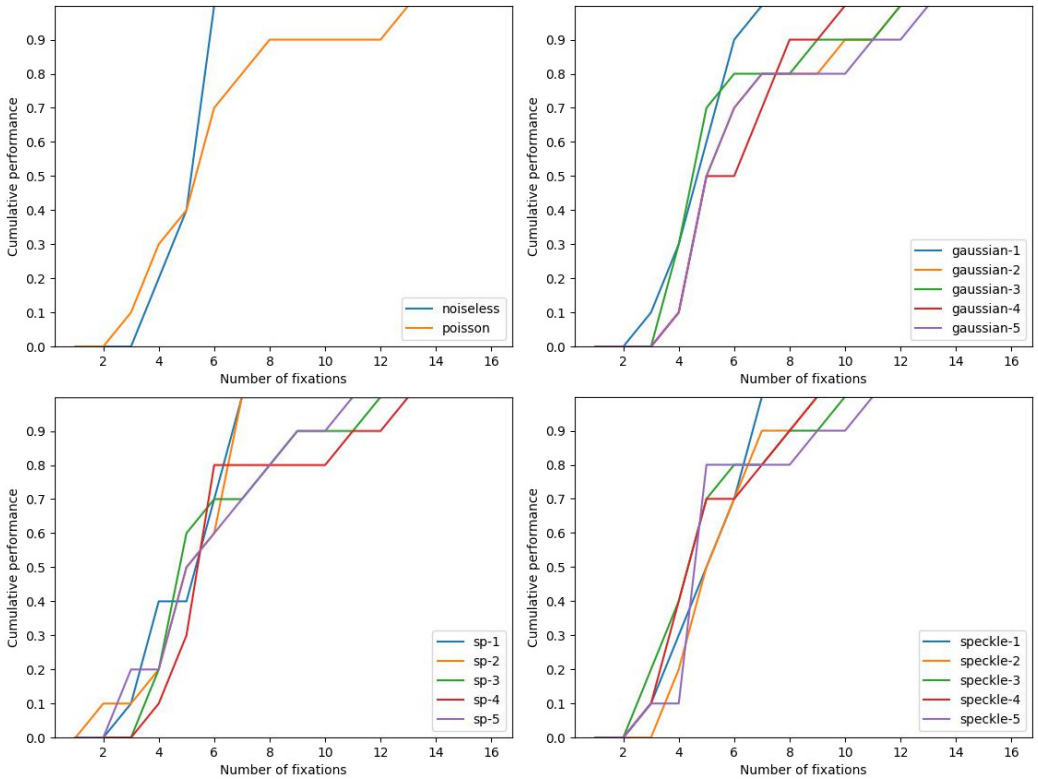


*Fig. 4.13:* Resistencia de nnIBS a distintos tipos de ruido, donde las curvas representan: *noiseless* se corresponde con las imagenes sin ruido; poisson a las imágenes con ruido *poisson*, *gaussian-1* a *gaussian-5* a las imágenes con ruido gaussiano (varianza en 0.02; 0.04; 0.06; 0.08; 0.10 respectivamente); *sp-1* a *sp-5* a las imágenes con ruido *salt & pepper* (cantidad en 0.05; 0.10; 0.15; 0.20; 0.25 respectivamente); y *speckle-1* a *speckle-5* a las imágenes con ruido *speckle* (varianza en 0.05; 0.10; 0.15; 0.20; 0.25 respectivamente).

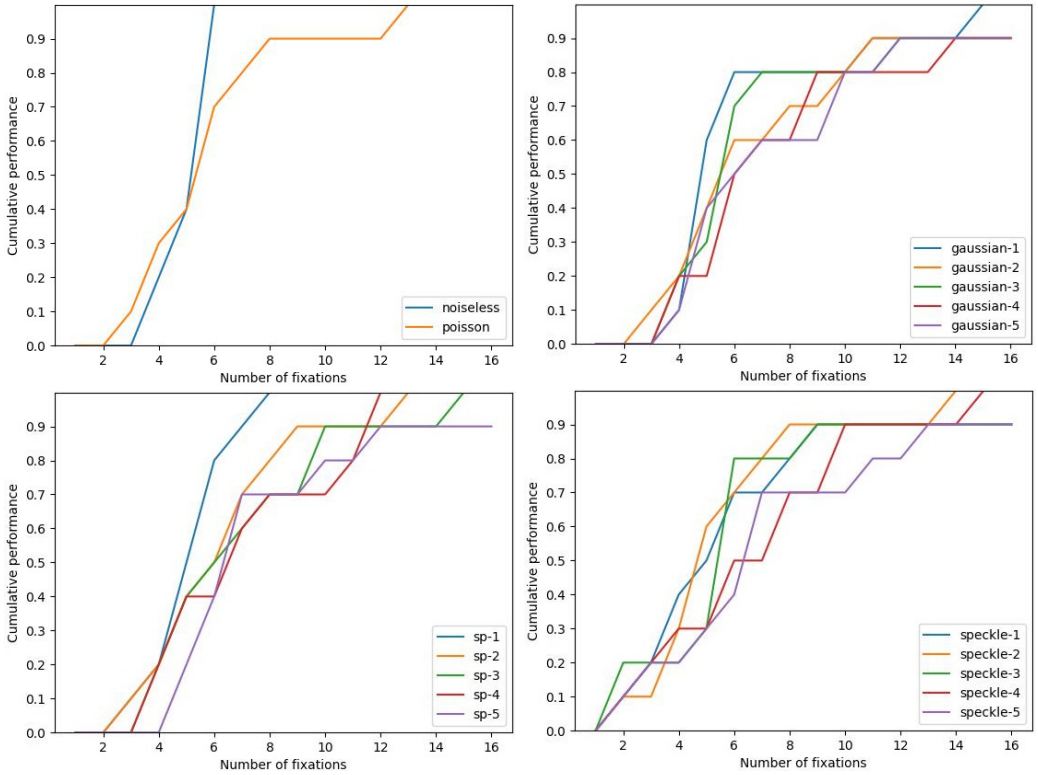


*Fig. 4.14:* Resistencia de nnIBS a distintos tipos de ruido, donde las curvas representan: *noiseless* se corresponde con las imagenes sin ruido; poisson a las imágenes con ruido *poisson*, *gaussian-1* a *gaussian-5* a las imágenes con ruido gaussiano (varianza en 0.1; 0.2; 0.3; 0.4; 0.5 respectivamente); *sp-1* a *sp-5* a las imágenes con ruido *salt & pepper* (cantidad en 0.1; 0.2; 0.3; 0.4; 0.5 respectivamente); y *speckle-1* a *speckle-5* a las imágenes con ruido *speckle* (varianza en 1; 2; 3; 4; 5 respectivamente).

En base a estos resultados (ver figura 4.14), el modelo parece ser bastante robusto ante la presencia de ruido en las imágenes. No obstante, hay que remarcar que depende bastante de la imagen, de la tarea y del tipo de ruido, no necesariamente de la cantidad de ruido.

# 5. CONCLUSIONES

En el presente trabajo se optimizó una versión del modelo cIBS en Python, superando la performance de la versión anterior en Matlab, y abriendo la posibilidad a mejorar aún más en el futuro. A partir de esta nueva implementación se trabajó sobre todo sobre el criterio se similaridad de una región de la imagen y el target. Se desarrolló una versión del modelo basado en redes profundas (nnIBS) capaz de generalizar el target a distintas instancias del mismo (perspectivas, tamaños, etc). Este modelo resultó robusto frente a distintos tipos de ruido aplicados a las imágenes de entrada, pero presenta sesgos en su búsqueda cuando hay, en principio, caras, comida y/o letras en la imagen. Se realizó una búsqueda de hiperparámetros y modificaciones para adaptarlo a búsquedas en imágenes a color. Sin embargo, para el caso de los datasets a color, en todos los casos parecía ser más atraído por los distractores.

Aunque a veces se lo subestime desde la perspectiva de ciencias de la computación, recopilar datos es todo un trabajo en sí mismo. Es una etapa que no debe tomarse a la ligera y demanda bastante tiempo, si se tienen en cuenta las diferencias que hay en cada dataset y la toma de decisiones que implica adaptar un conjunto de datos a un modelo o viceversa, y cómo ello puede influir más de lo previsto en los resultados. Dada la gran cantidad de parámetros tanto físicos (tamaño, color, ángulo, distancia, etc) como de contenido, esta etapa resulta particularmente clave en los casos en los que se trabaja con imágenes.

Brindar un enfoque orientado a objetos facilita la incorporación de mejoras o refactorizaciones a componentes del modelo, sin necesidad de hacer cambios en gran parte del código. Además, tener el código escrito en Python facilita la escalabilidad a futuro y la experimentación con el mismo. Es una gran mejora desde ese punto de vista ya que anteriormente todas las versiones del modelo estaban escritas en Matlab.

El aporte de nnIBS es bastante grande en términos de generalización de objetivos, mas no tanto en eficiencia en la búsqueda. Otra ventaja es que es mucho más rápido que las versiones anteriores como sIBS por ejemplo.

## Trabajo a futuro

Las métricas consideradas junto con las experimentaciones a nivel global con cada dataset, permiten cuantificar de distintas formas la eficiencia de las versiones del modelo. Sin embargo, es con el análisis fino de *scanpaths* particulares que se pueden descubrir casos sencillos, casos difíciles, sesgos, patrones interesantes, etc. que sirven para en un futuro poder mejorar aún más la eficiencia y el comportamiento.

De este trabajo se desprenden distintos ejes desde los cuales se puede mejorar aún más el modelo, entre los que destacan:

- MEMORIA DE TRABAJO: A modo de trabajo a futuro se propone lograr que el modelo almacene una historia finita (las últimas n fijaciones). Se puede pensar truncando la sumatoria de la ecuación 1.2 en su límite inferior. Existen otros trabajos en los que se realizaron enfoques similares [8].

- SESGOS Y CONTEXTO: Es importante encontrar nuevos y mejores priors. Por un lado es útil para eliminar los sesgos actuales, entre los que se pudieron observar: caras en sección 4.5; letras y comida en sección 4.6. Por otro lado sirve para incorporar información de contexto más compleja, como por ejemplo la relación que suele haber entre la posición de una mesa con la posición de un techo. Un enfoque interesante es el de [6], en el que utilizan una red de detección de objetos[1] para generar por cada lo que ellos llaman DCBs, que es en donde realizan la búsqueda realmente. Este enfoque está limitado por los objetos que la red es capaz de reconocer.

- BÚSQUEDA CATEGÓRICA: Inicialmente cIBS era capaz de realizar búsquedas eficientes tomando como objetivo el recorte correspondiente dentro de cada imagen. Con nnIBS se mejoró este escenario de forma tal que le basta con conocer una imagen con instancia distinta del mismo tipo de objetivo para hacer una búsqueda eficiente (visto en sección 4.4). Un siguiente paso consistiría en que el modelo pueda realizar una búsqueda categórica. Es decir, que pueda tomar como objetivo el nombre de la categoría a la que pertenece (reloj, vaso, etc.). El mismo enfoque de [6] en el que utilizan una red de detección de objetos, sirve para que su modelo pueda hacer una búsqueda categórica. No obstante, este enfoque está limitado, en este caso, por las categorías que conoce el modelo.

- MODELO DE LA FÓVEA: El modelo actualmente emula la fóvea mediante una gaussiana, pero existen trabajos en los que la modelan de forma más compleja [9]. Incorporar nuevos mapas de visibilidad sin perder la generalización a cualquier imagen que posee el actual, es otro desafío a encarar.

- TAMAÑO DE LA FIJACIÓN: Además, si bien el tamaño de fijación elegido parece razonable, es importante analizar y corregir las dificultades que se presentaron a la hora de usar tamaños más grandes, ya que habla de poca robustez por parte del modelo: Se puede hacer un análisis más exhaustivo tal vez manipulando variables en el Search Model o verificando hasta qué tamaño de $\delta$ el modelo sigue manteniendo su esencia comportamental. También vale la pena analizar si la relación entre el tamaño del objetivo y el valor de $\delta$ influyen en la capacidad de búsqueda del modelo. Asimismo, si bien no hubo cambios sustanciales, se propone profundizar la exploración de hiperparámetros.

- RECONOCIMIENTO DEL OBJETIVO / FINALIZACIÓN DE LA BÚSQUEDA: Por su parte, implementar un método para decidir si se encontró o no el objetivo que reemplace al oráculo actual sería un gran paso hacia el comportamiento humano y hacia la no supervisión del modelo (no debería saber de antemano donde se encuentra el objetivo). Una posibilidad es agregar un nivel de confianza (oscilando entre 0 y 1) basándose en el mapa probabilístico que determina la siguiente fijación: si la probabilidad de que la fijación elegida sea la correcta es mucho mayor a la de las demás fijaciones, la confianza sería alta. En cambio, si las probabilidades son similares, la confianza será baja. Si la confianza supera un umbral, el modelo asume que encontró el objetivo.

- IMÁGENES A COLOR: Si bien para los humanos las búsquedas se facilitan cuando hay imágenes a color [41], con el modelo no sucede lo mismo (ver sección 4.4). Falta

[1] https://github.com/facebookresearch/detectron2/blob/main/README.md

hacer un análisis de bajo nivel corriendo el modelo sobre la misma imagen en escala de grises y a color, y ver a partir de qué momento empieza a haber discrepancias.

- MÉTRICAS: Si bien se hizo mucho énfasis en métricas que miden la eficiencia o comparan *scanpaths* enteros (MultiMatch), también se implementó un protocolo para comparar un *scanpath* generado por el modelo frente a un ground truth que suele ser un *scanpath* humano, calculando fijación a fijación el NSS y el AUC (Ver sección 3.2) mas no se las aprovechó del todo en este trabajo. Puede ser útil agregar otras métricas a este protocolo (como information gain [40]) y utilizarlo para hacer análisis más finos.

- RUIDO EN LA POSICIÓN DE INICIO: Una optimización en la implementación permitiría matchear el punto de inicio exacto para cada sujeto e imagen. Si bien esto implica, en principio, correr el modelo tantas veces como sujetos hayan visto cada imagen, hoy todos los modelos comienzan desde el mismo lugar (el punto de inicio propuesto por el experimentador), aunque los sujetos se hayan movido levemente del mismo (cayendo a veces en otra celda). Esto puede ser particularmente útil a la hora de calcular métricas fijación a fijación, ya que los resultados serían más precisos.

- MOVIMIENTOS OCULARES EN FUNCIÓN DE LA IMAGEN (Y DEL MODELO): Analizar qué condiciones afectan las propiedades de los movimientos oculares que realiza el modelo. Por un lado, en muchos scanpaths se ven sacadas largas y sin un patrón evidente en las imágenes. Por otro lado, en scanpaths de más de 20 fijaciones es posible que se pierda la propiedad de IoR. Esto último puede estar relacionado con que el modelo retiene una historia infinita, lo cual podría estar generando ruido a partir de algún número de fijaciones. Habría que corroborarlo quitando el límite de fijaciones del modelo.

Finalmente, hoy en día el grueso del poder de cómputo del modelo se encuentra en la correlación cruzada. Mejoraría la cantidad de experimentos que se pueden hacer en un tiempo determinado si se logran optimizar y/o paralelizar los cálculos. Esto sumado a que hay un cluster en el DC, implementar el modelo para que sea capaz de aprovecharlo al máximo sería el escenario ideal.

## Bibliografía


[1] Travi F (2021) "Modelos computacionales de búsqueda visual humana en escenas naturales: Comparación de modelos y conjuntos de datos de referencia" (Tesis en Cs. de la Computación, FCEyN, UBA; director: Kamienkowski JE; co-director: Bujia G

[2] Sclar, M., Bujia, G., Vita, S., Solovey, G., & Kamienkowski, J. E. (2020). Modeling human visual search: A combined Bayesian searcher and saliency map approach for eye movement guidance in natural scenes. arXiv preprint arXiv:2009.08373.

[3] Bujia, G., Sclar, M., Vita, S., Solovey, G., & Kamienkowski, J. E. (enviado). Modeling human visual search: A combined Bayesian searcher and saliency map approach for eye movement guidance in natural scenes.

[4] Zhang, M., Feng, J., Ma, K. T., Lim, J. H., Zhao, Q., & Kreiman, G. (2018). Finding any Waldo with zero-shot invariant and efficient visual search. Nature communications, 9(1), 1-15.

[5] Zelinsky, G., Yang, Z., Huang, L., Chen, Y., Ahn, S., Wei, Z., ... & Hoai, M. (2019). Benchmarking gaze prediction for categorical visual search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).

[6] Yang, Z., Huang, L., Chen, Y., Wei, Z., Ahn, S., Zelinsky, G., ... & Hoai, M. (2020). Predicting goal-directed human attention using inverse reinforcement learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 193-202).

[7] Chen, Y., Yang, Z., Ahn, S., Samaras, D., Hoai, M., & Zelinsky, G. (2021). COCO-Search18 Fixation Dataset for Predicting Goal-directed Attention Control. Scientific Reports, 11 (1), 1-11, 2021.

[8] Zhou, Y., & Yu, Y. (2021). Human visual search follows a suboptimal Bayesian strategy revealed by a spatiotemporal computational model and experiment. Communications Biology, 4(1), 1-16.

[9] Rashidi, S., Ehinger, K., Turpin, A., & Kulik, L. (2020). Optimal visual search based on a model of target detectability in natural images. Advances in Neural Information Processing Systems, 33.

[10] Jarodzka, H., Holmqvist, K., & Nyström, M. (2010). "A vector-based, multidimensional scanpath similarity measure". In Proceedings of the 2010 symposium on eye-tracking research & applications (pp. 211-218).

[11] Dewhurst, R., Nyström, M., Jarodzka, H., Foulsham, T., Johansson, R., & Holmqvist, K. (2012). It depends on how you look at it: Scanpath comparison in multiple dimensions with MultiMatch, a vector-based approach. Behavior research methods, 44(4), 1079-1100.

[12] Najemnik, J., & Geisler, W. S. (2005). Optimal eye movement strategies in visual search. Nature, 434(7031), 387-391.

[13] Bradley, C., Abrams, J., & Geisler, W. S. (2014). Retina-V1 model of detectability across the visual field. Journal of vision, 14(12), 22-22.

[14] Zhou Wang; Bovik, A.C.; ,"Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," Signal Processing Magazine, IEEE, vol. 26, no. 1, pp. 98-117, Jan. 2009.;

[15] L. W. Renninger, P. Verghese, and J. Coughlan, "Where to look next? eye movements reduce local uncertainty," Journal of vision, vol. 7, no. 3, pp. 6–6, 2007.

[16] Kümmerer, Matthias & Wallis, Thomas & Bethge, Matthias. (2016). DeepGaze II: Reading fixations from deep features trained on object recognition.

[17] Bindemann M, Burton AM, Hooge IT, Jenkins R, de Haan EH. Faces retain attention. Psychon Bull Rev. 2005 Dec;12(6):1048-53. doi: 10.3758/bf03206442. PMID: 16615327.

[18] Matthias Kümmerer, Matthias Bethge. State-of-the-Art in Human Scanpath Prediction arXiv preprint: arXiv:2102.12239v2

[19] Sun W, Chen Z, Wu F. Visual Scanpath Prediction Using IOR-ROI Recurrent Mixture Density Network. IEEE Trans Pattern Anal Mach Intell. 2021 Jun;43(6):2101-2118. doi: 10.1109/TPAMI.2019.2956930. Epub 2021 May 11. PMID: 31796389.

[20] Sen Jia and Neil D.B. Bruce. Eml-net: An expandable multilayer network for saliency prediction. Image and Vision Computing, 95:103887, 2020

[21] Matthias Kümmerer, Lucas Theis, and Matthias Bethge. Deepgaze i: Boosting saliency prediction with feature maps trained on imagenet. arXiv preprint arXiv:1411.1045, 2014.

[22] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 20, no. 11, pp. 1254–1259, 1998

[23] Yantis, S. & Hillstrom, A. P. Stimulus-driven attentional capture: evidence from equiluminant visual objects. J. Exp. Psychol. Hum. Percept. Perform. 20, 95–107 (1994).

[24] Tatler, Benjamin & Vincent, Benjamin. (2008). Systematic tendencies in scene viewing. Journal of Eye Movement Research. 2. 1-18. 10.16910/jemr.2.2.5.

[25] Wilming N, Harst S, Schmidt N, König P. Saccadic momentum and facilitation of return saccades contribute to an optimal foraging strategy. PLoS Comput Biol. 2013;9(1):e1002871. doi:10.1371/journal.pcbi.1002871

[26] M. Kümmerer, T. S. A. Wallis, L. A. Gatys and M. Bethge, Ünderstanding Low- and High-Level Contributions to Fixation Prediction,"2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 4799-4808, doi: 10.1109/ICCV.2017.513.

[27] Components of visual orienting - M.I. Posner and Y. Cohen - H. Bouma, D.G. Bouwhuis (Eds.), Attention and performance, Vol. 10, Erlbaum, Hillsdale, NJ (1984), pp. 531-556

[28] Zhiguo Wang, Raymond M. Klein, Searching for inhibition of return in visual search: A review, Vision Research, Volume 50, Issue 2, 2010

[29] C. Goble, "Better Software, Better Research,ïn IEEE Internet Computing, vol. 18, no. 5, pp. 4-8, Sept.-Oct. 2014, doi: 10.1109/MIC.2014.88.

[30] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014) https://doi.org/10.7717/peerj.453

[31] https://docs.python.org/3/c-api/init.html

[32] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.

[33] Simonyan, Karen and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." CoRR abs/1409.1556 (2015): n. pag.

[34] Lake BM, Ullman TD, Tenenbaum JB, Gershman SJ. Building machines that learn and think like people. Behav Brain Sci. 2017 Jan;40:e253.
doi: 10.1017/S0140525X16001837. Epub 2016 Nov 24. PMID: 27881212.

[35] Dehaene, S. & Sigman, M. (2012). From a single decision to a multi-step algorithm. Current Opinion in Neurobiology, 22(6), 937-945.
http://dx.doi.org/10.1016/j.conb.2012.05.006

[36] Kamienkowski, J. E., Varatharajah, A., Sigman, M., & Ison, M. J. (2018). Parsing a mental program: Fixation-related brain signatures of unitary operations and routines in natural visual search. NeuroImage, 183, 73–86.
https://doi.org/10.1016/j.neuroimage.2018.08.010

[37] Katsuki F, Constantinidis C. Bottom-up and top-down attention: different processes and overlapping neural systems. Neuroscientist. 2014 Oct;20(5):509-21. doi: 10.1177/1073858413514136. Epub 2013 Dec 20. PMID: 24362813.

[38] Grill-Spector K, Weiner KS, Kay K, Gomez J. The Functional Neuroanatomy of Human Face Perception. Annu Rev Vis Sci. 2017 Sep 15;3:167-196. doi: 10.1146/annurev-vision-102016-061214. Epub 2017 Jul 17. PMID: 28715955; PMCID: PMC6345578.

[39] Wardle, S.G., Taubert, J., Teichmann, L. et al. Rapid and dynamic processing of face pareidolia in the human brain. Nat Commun 11, 4518 (2020). https://doi.org/10.1038/s41467-020-18325-8

[40] Kümmerer M, Wallis TS, Bethge M. Information-theoretic model comparison unifies saliency metrics. Proc Natl Acad Sci U S A. 2015 Dec 29;112(52):16054-9. doi: 10.1073/pnas.1510393112. Epub 2015 Dec 10. PMID: 26655340; PMCID: PMC4702965.

[41] Zhuang X, Papathomas TV. Cue relevance effects in conjunctive visual search: cueing for location, color, and orientation. J Vis. 2011 Jun 8;11(7):6. doi: 10.1167/11.7.6. PMID: 21652771.

# 5. APÉNDICE

## 5.1. Imágenes eliminadas de MCS

['000000215878.jpg', '000000435046.jpg', '000000154382.jpg', '000000516439.jpg', '000000331326.jpg', '000000100558.jpg', '000000352252.jpg', '000000091942.jpg', '000000302147.jpg','000000530533.jpg', '000000067647.jpg', '000000432417.jpg', '000000280850.jpg', '000000401182.jpg', '000000330990.jpg', '000000274632.jpg', '000000121612.jpg', '000000104326.jpg', '000000124004.jpg', '000000344488.jpg', '000000444531.jpg', '000000398661.jpg', '000000206784.jpg', '000000571808.jpg', '000000553129.jpg', '000000068833.jpg', '000000095569.jpg', '000000117899.jpg', '000000262054.jpg','000000533978.jpg'].

## 5.2. Imágenes eliminadas de Unrestricted

['img027.jpg', 'img041.jpg', 'img115.jpg', 'img120.jpg', 'img165.jpg', 'img183.jpg'] Particularmente en la imagen img047.jpg el target se escapa de los límites, ocupando hasta el pixel 1027. Se corrigió esto restándole 3 pixeles.

## 5.3. Algunos ejemplos

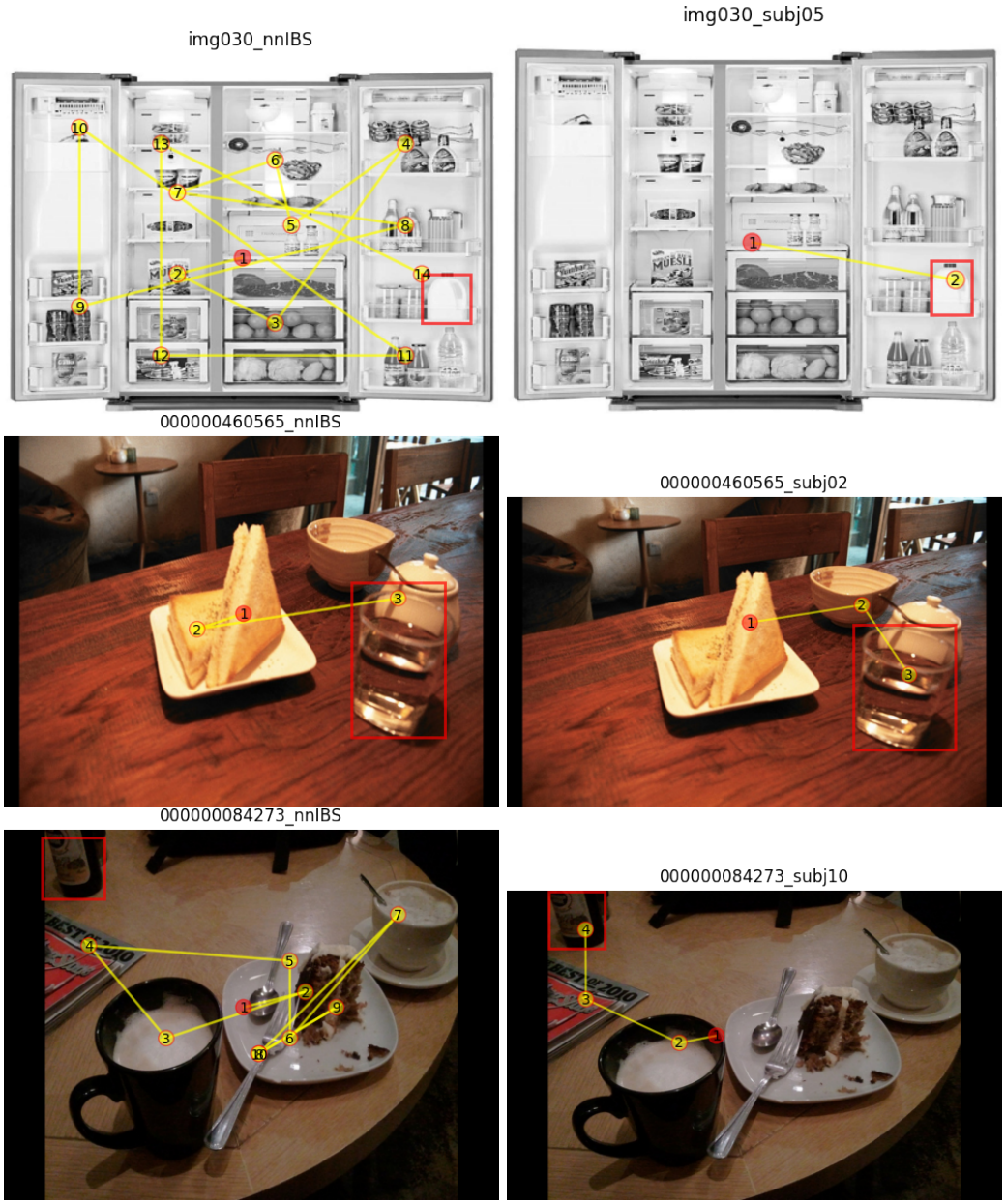


*Fig. 5.1:* Fila superior: *scanpath* generado por nnIBS en imagen img030 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 5 en la imagen img030 del *dataset Unrestricted* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo prioriza regiones de la imagen con comida presente. Fila media: *scanpath* generado por nnIBS en imagen 000000460565 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 2 en la imagen 000000460565 del *dataset COCOSearch-18* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo se distrajo con comida en la primera sacada. El objetivo es bastante grande. Fila inferior: *scanpath* generado por nnIBS en imagen 000000084273 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 16 en la imagen 000000084273 del *dataset COCOSearch-18* (derecha). El modelo no encuentra el objetivo y sesga su búsqueda al centro de la imagen, donde también hay comida presente.

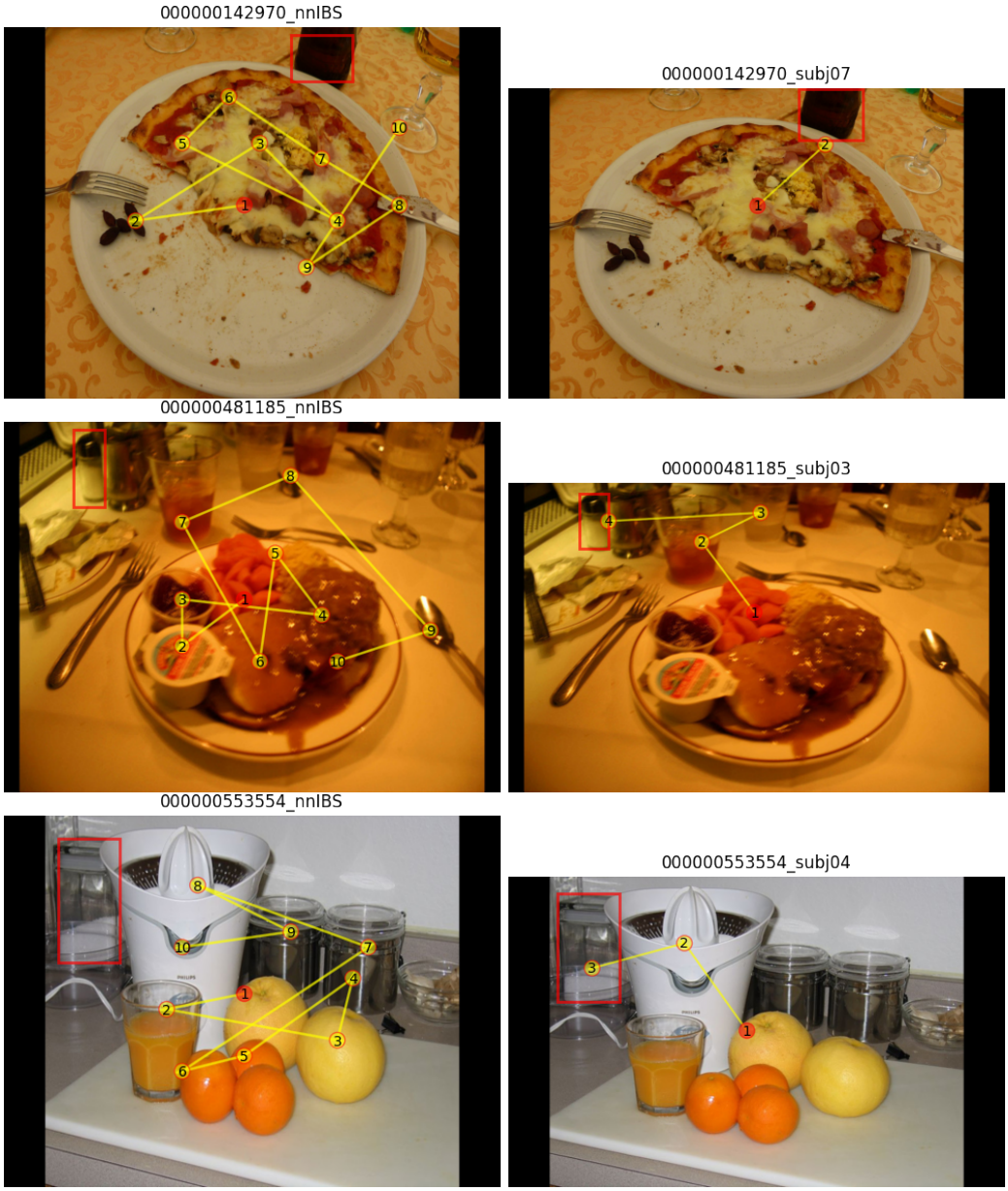


*Fig. 5.2:* Fila superior: *scanpath* generado por nnIBS en imagen 000000142970 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 7 en la imagen 000000142970 del *dataset COCOSearch-18* (derecha). El modelo no encuentra el objetivo y sesga su búsqueda a regiones de la imagen donde hay comida presente. Fila media: *scanpath* generado por nnIBS en imagen 000000481185 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 3 en la imagen 000000481185 del *dataset COCOSearch-18* (derecha). El modelo no encuentra el objetivo y sesga su búsqueda a regiones de la imagen donde hay comida presente. Fila inferior: *scanpath* generado por nnIBS en imagen 000000553554 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 4 en la imagen 000000553554 del *dataset COCOSearch-18* (derecha). El modelo no encuentra el objetivo y sesga su búsqueda a regiones de la imagen donde hay comida presente.

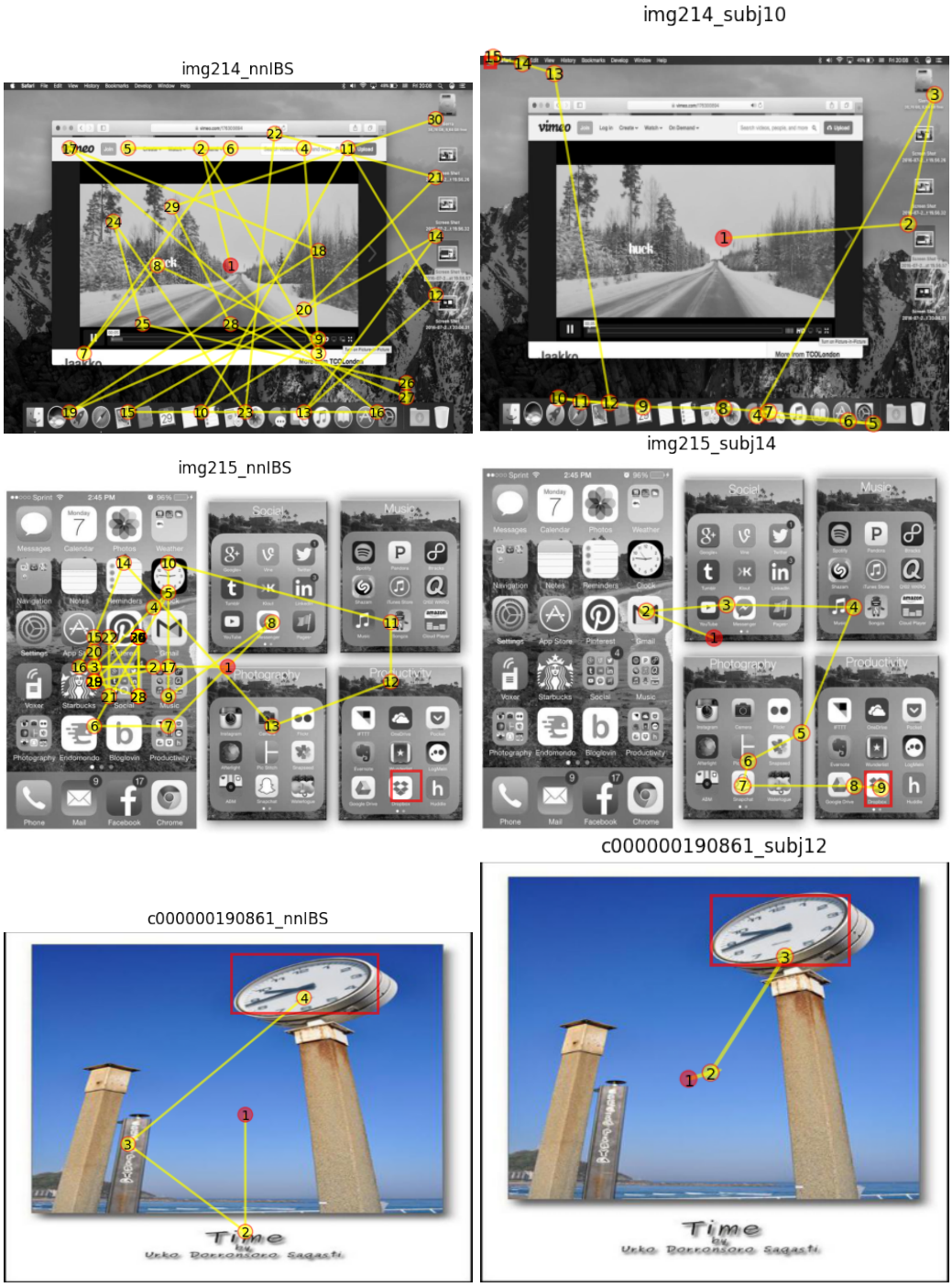


*Fig. 5.3:* Fila superior: *scanpath* generado por nnIBS en imagen img211 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 9 en la imagen img211 del *dataset Unrestricted* (derecha). Al modelo le toma muchas sacadas encontrar al objetivo, y este es demasiado pequeño. Hay logos y palabras en la imagen. Hay sacadas bastante largas por parte del modelo. Fila media: *scanpath* generado por nnIBS en imagen img238 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 14 en la imagen img215 del *dataset Unrestricted* (derecha). El modelo no encuentra al objetivo. Hay muchos logos en la imagen, de los cuales hay algunos que pueden interpretarse como letras. Fila inferior: *scanpath* generado por nnIBS en imagen c000000190861 del *dataset MCS* (izquierda) y *scanpath* generado por el sujeto 12 en la imagen c000000190861 del *dataset MCS* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo presenta un sesgo hacia letras/palabras. El objetivo es grande.

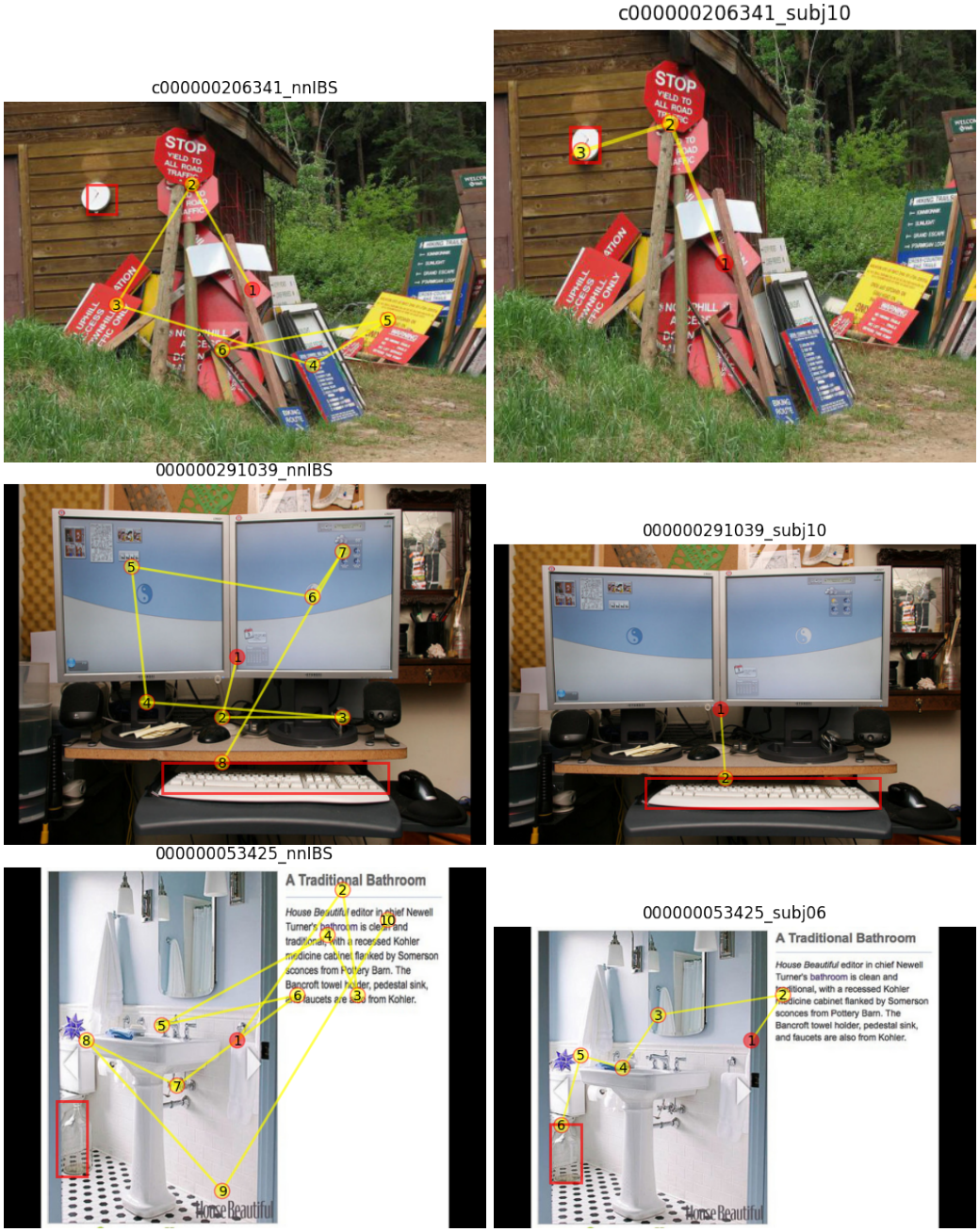


*Fig. 5.4:* Fila superior: *scanpath* generado por nnIBS en imagen c000000206341 del *dataset MCS* (izquierda) y *scanpath* generado por el sujeto 10 en la imagen c000000206341 del *dataset MCS* (derecha). El modelo no encuentra el objetivo. Hay diversos carteles con palabras en la imagen. Fila media: *scanpath* generado por nnIBS en imagen 000000291039 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 10 en la imagen 000000291039 del *dataset COCOSearch-18* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo se distrajo con diversos objetos pero los sujetos no presentaron ese incoveniente. El objetivo es grande. Hay íconos y palabras en la imagen. Fila inferior: *scanpath* generado por nnIBS en imagen 000000053425 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 6 en la imagen 000000053425 del *dataset COCOSearch-18* (derecha). El modelo no encuentra el objetivo, y sesga su búsqueda a sectores de la imagen con palabras.

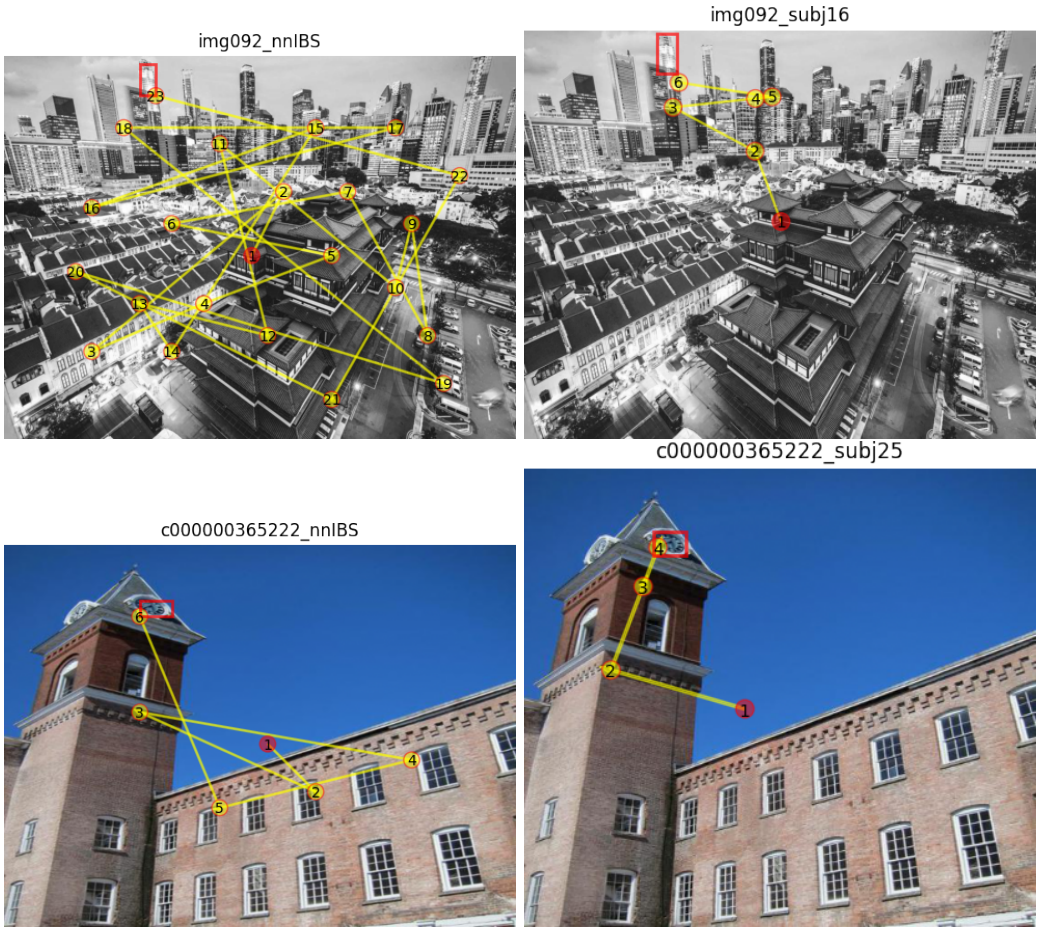


*Fig. 5.5:* Fila superior: *scanpath* generado por nnIBS en imagen img092 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 16 en la imagen img092 del *dataset Unrestricted* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. Los sujetos buscan el edificio en las regiones superiores de la imagen, pero el modelo busca por toda la imagen. Fila inferior: *scanpath* generado por nnIBS en imagen c000000365222 del *dataset MCS* (izquierda) y *scanpath* generado por el sujeto 25 en la imagen c000000365222 del *dataset MCS* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. Los sujetos buscan el reloj a lo alto de la torre, pero el modelo busca por todo el edificio.


*Fig. 5.6:* Fila superior: *scanpath* generado por nnIBS en imagen img238 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 11 en la imagen img238 del *dataset Unrestricted* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo se distrajo con diversos objetos pero los sujetos no presentaron ese incoveniente. El modelo realiza sacadas bastante largas. Fila media: *scanpath* generado por nnIBS en imagen img092 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 16 en la imagen img092 del *dataset Unrestricted* (derecha). El modelo no encuentra el objetivo, y este es demasiado pequeño. Fila inferior: *scanpath* generado por nnIBS en imagen grayscale_24_oliva del *dataset Interiors* (izquierda) y *scanpath* generado por el sujeto 5 en la imagen grayscale_24_oliva del *dataset Interiors* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo no parece captar del todo qué objeto está buscando. Hay sacadas bastante largas por parte del modelo.

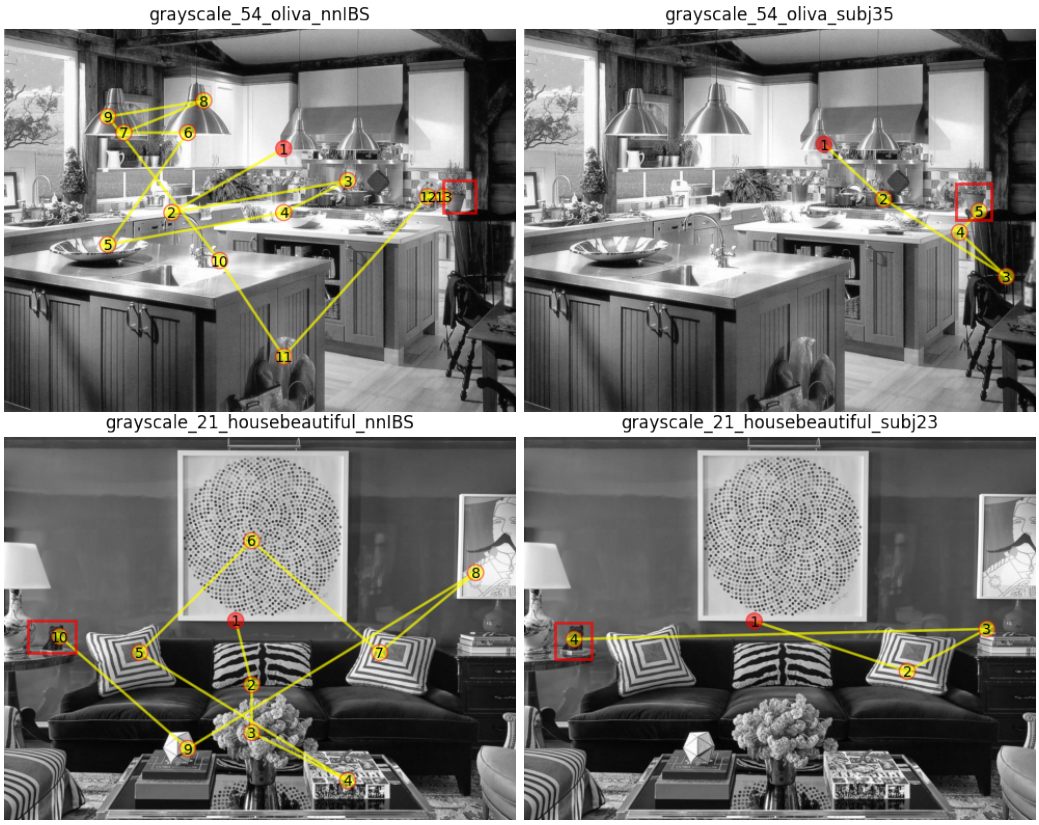


*Fig. 5.7:* Fila superior: *scanpath* generado por nnIBS en imagen grayscale_54_oliva del *dataset Interiors* (izquierda) y *scanpath* generado por el sujeto 35 en la imagen grayscale_54_oliva del *dataset Interiors* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo no parece captar del todo qué objeto está buscando. Fila inferior: *scanpath* generado por nnIBS en imagen grayscale_21_housebeautiful del *dataset Interiors* (izquierda) y *scanpath* generado por el sujeto 23 en la imagen grayscale_21_housebeautiful del *dataset Interiors* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra cerca de la diagonal, indicando que el *scanpath* del modelo se asemeja bastante al de los humanos. Hay sacadas bastante largas por parte del modelo y del sujeto 23.

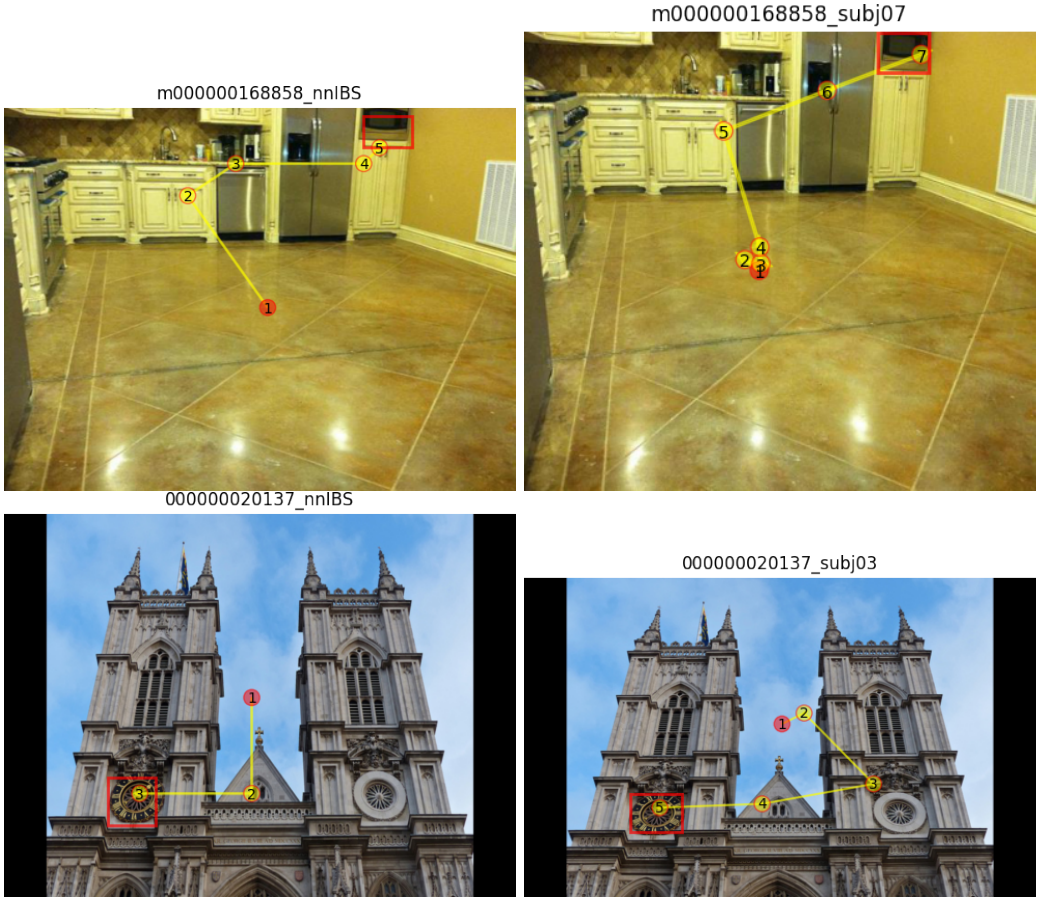


*Fig. 5.8:* Fila superior: *scanpath* generado por nnIBS en imagen m000000168858 del *dataset MCS* (izquierda) y *scanpath* generado por el sujeto 7 en la imagen m000000168858 del *dataset MCS* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra cerca de la diagonal, indicando que el *scanpath* del modelo se asemeja bastante al de los humanos. No hay muchos distractores en la imagen, lo cual simplifica la tarea tanto al modelo como a los sujetos, y sesgan (correctamente) su búsqueda a regiones superiores de la imagen. Fila inferior: *scanpath* generado por nnIBS en imagen 000000020137 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 3 en la imagen 000000020137 del *dataset COCOSearch-18* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra cerca de la diagonal, indicando que el *scanpath* del modelo se asemeja bastante al de los humanos.

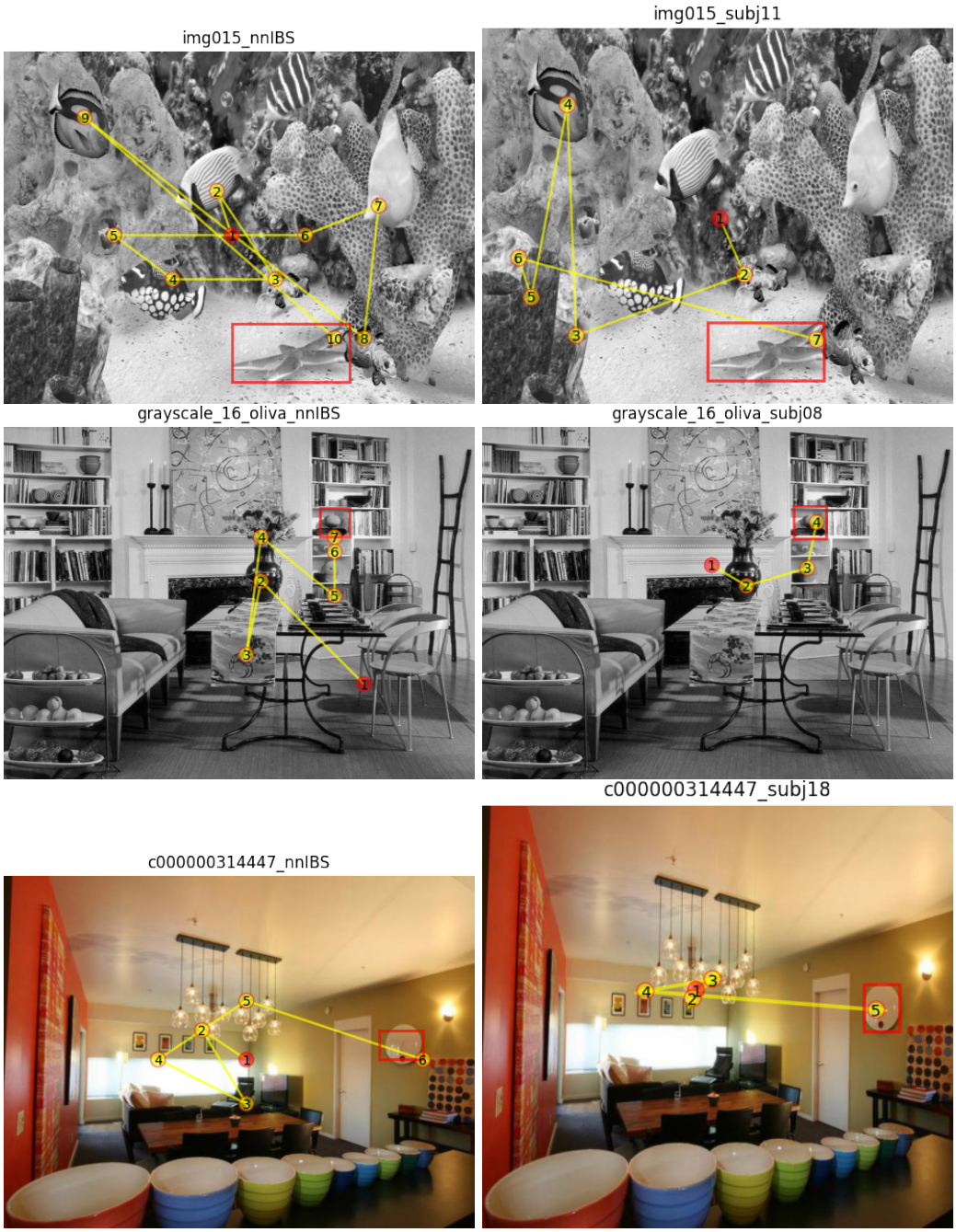


*Fig. 5.9:* Fila superior: *scanpath* generado por nnIBS en imagen img015 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 11 en la imagen img015 del *dataset Unrestricted* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra cerca de la diagonal, indicando que el *scanpath* del modelo se asemeja bastante al de los humanos. Fila media: *scanpath* generado por nnIBS en imagen grayscale_16_oliva del *dataset Interiors* (izquierda) y *scanpath* generado por el sujeto 8 en la imagen grayscale_16_oliva del *dataset Interiors* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra cerca de la diagonal, indicando que el *scanpath* del modelo se asemeja bastante al de los humanos. Fila inferior: *scanpath* generado por nnIBS en imagen c000000314447 del *dataset MCS* (izquierda) y *scanpath* generado por el sujeto 18 en la imagen c000000314447 del *dataset MCS* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra cerca de la diagonal, indicando que el *scanpath* del modelo se asemeja bastante al de los humanos. Hay distractores en la imagen, lo cual dificulta la tarea tanto al modelo como a los sujetos.

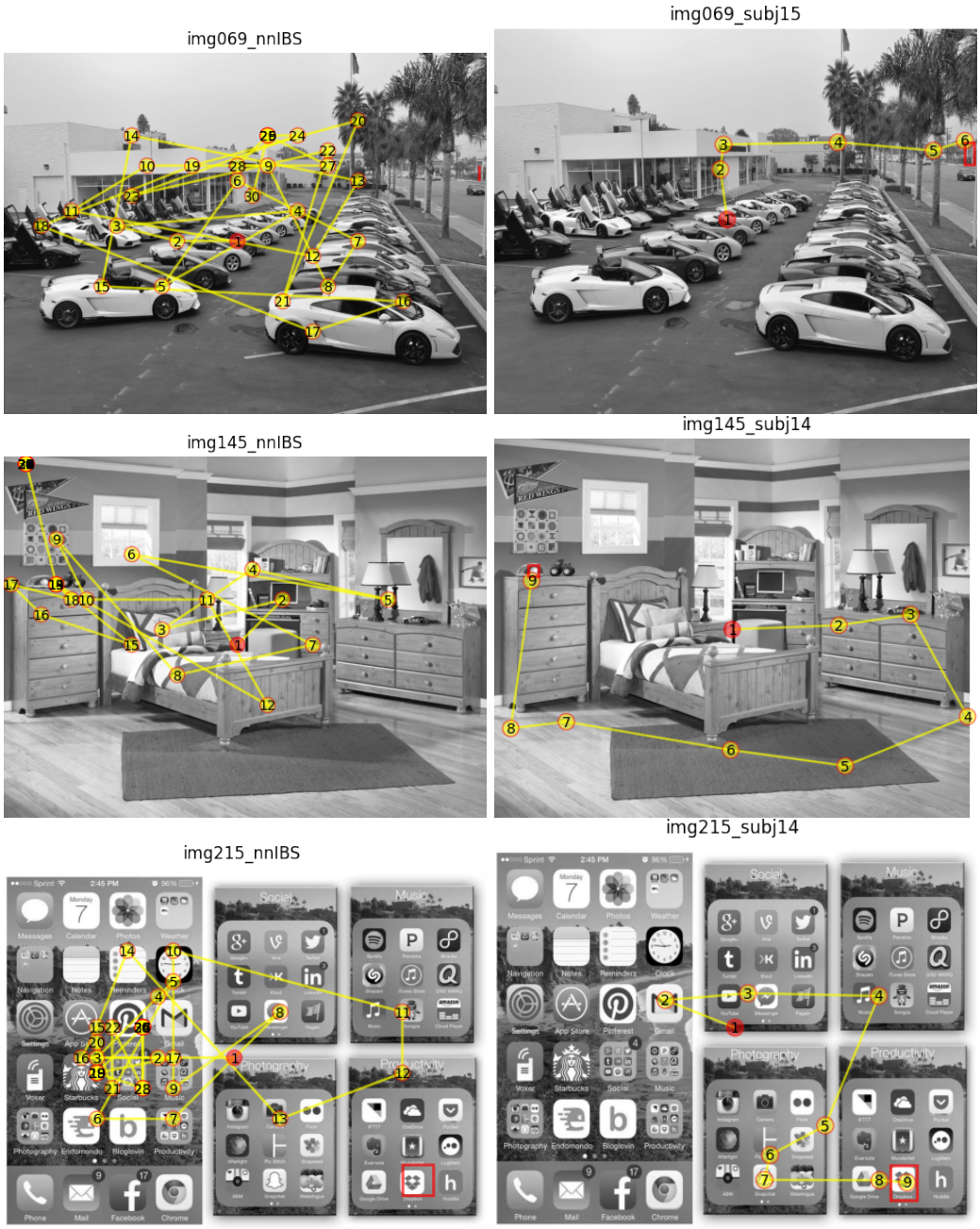


*Fig. 5.10:* Fila superior: *scanpath* generado por nnIBS en imagen img069 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 15 en la imagen img069 del *dataset Unrestricted* (derecha). El modelo no encuentra el objetivo, y este es demasiado pequeño. Fila media: *scanpath* generado por nnIBS en imagen img161 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 13 en la imagen img161 del *dataset Unrestricted* (derecha). El modelo no encuentra el objetivo. El objetivo no puede verse en la imagen, es posible que al ser tan pequeño haya habido algún problema al calcular sus límites. Fila inferior: *scanpath* generado por nnIBS en imagen img238 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 14 en la imagen img215 del *dataset Unrestricted* (derecha). El modelo no encuentra al objetivo. Hay muchos logos en la imagen, de los cuales hay algunos que pueden interpretarse como letras.


*Fig. 5.11:* Fila superior: *scanpath* generado por nnIBS en imagen img211 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 9 en la imagen img211 del *dataset Unrestricted* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo fue a buscar al objetivo a regiones de la imagen donde era imposible que esté. Fila inferior: *scanpath* generado por nnIBS en imagen grayscale_70_oliva del *dataset Interiors* (izquierda) y *scanpath* generado por el sujeto 37 en la imagen grayscale_70_oliva del *dataset Interiors* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo se distrajo con diversos objetos pero los sujetos no presentaron ese incoveniente. El modelo hizo sacadas bastante largas.


*Fig. 5.12:* Fila superior: *scanpath* generado por nnIBS en imagen m000000059868 del *dataset MCS* (izquierda) y *scanpath* generado por el sujeto 18 en la imagen m000000059868 del *dataset MCS* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo se distrajo con diversos objetos pero los sujetos no presentaron ese incoveniente. El modelo realiza sacadas bastante largas. Fila inferior: *scanpath* generado por nnIBS en imagen 000000045063 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 8 en la imagen 000000045063 del *dataset COCOSearch-18* (derecha). En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo se distrae con otros objetivos antes de encontrar el objetivo. La última sacada es bastante larga. El objetivo es grande.

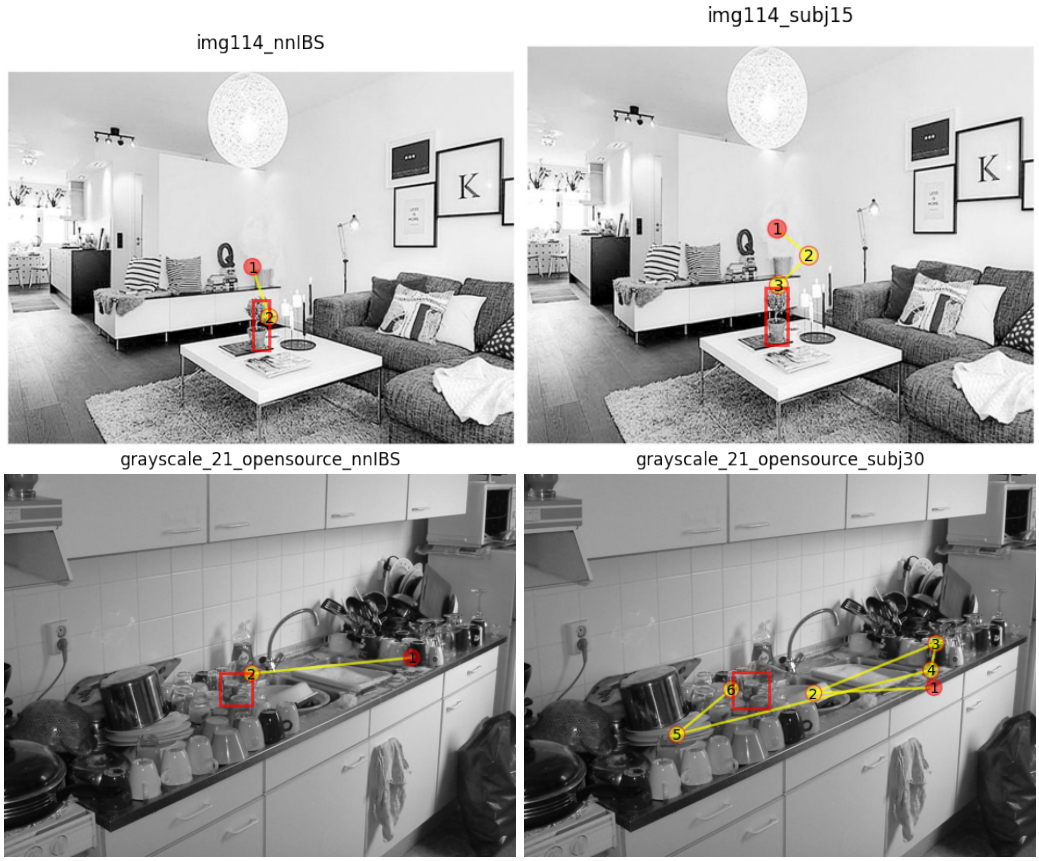


*Fig. 5.13:* Fila superior: *scanpath* generado por nnIBS en imagen img114 del *dataset Unrestricted* (izquierda) y *scanpath* generado por el sujeto 15 en la imagen img114 del *dataset Unrestricted* (derecha). El modelo encuentra el objetivo en una sacada, y este está cerca del centro. Fila inferior: *scanpath* generado por nnIBS en imagen grayscale_21_opensource del *dataset Interiors* (izquierda) y *scanpath* generado por el sujeto 30 en la imagen grayscale_21_opensource del *dataset Interiors* (derecha). El modelo encuentra el objetivo en una sacada, y este está cerca del centro.

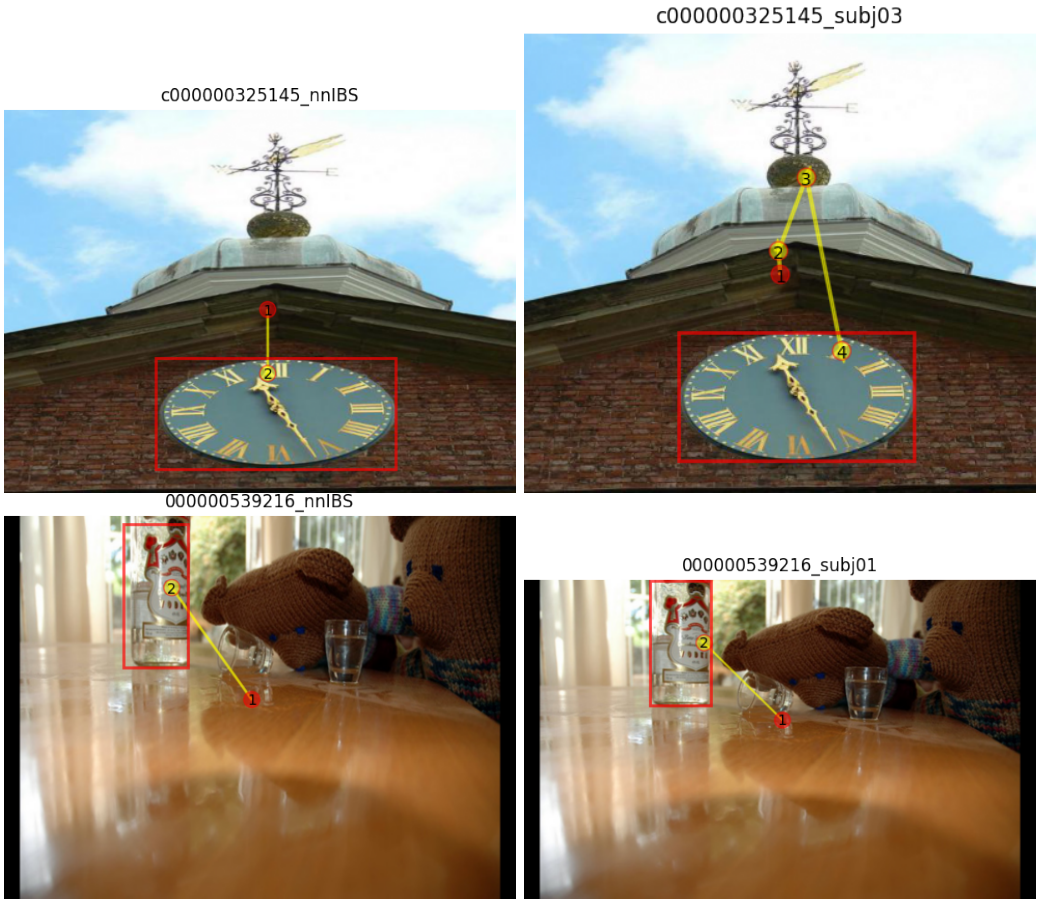


*Fig. 5.14:* Fila superior: *scanpath* generado por nnIBS en imagen c000000325145 del *dataset MCS* (izquierda) y *scanpath* generado por el sujeto 3 en la imagen c000000325145 del *dataset MCS* (derecha). El modelo encuentra el objetivo en una sacada. El objetivo es muy grande y se encuentra cerca del centro (y de la fijación inicial). Fila inferior: *scanpath* generado por nnIBS en imagen 000000539216 del *dataset COCOSearch-18* (izquierda) y *scanpath* generado por el sujeto 1 en la imagen 000000539216 del *dataset COCOSearch-18* (derecha). El modelo encuentra el objetivo en una sacada. El objetivo es grande y está cerca del centro (y de la fijación inicial).

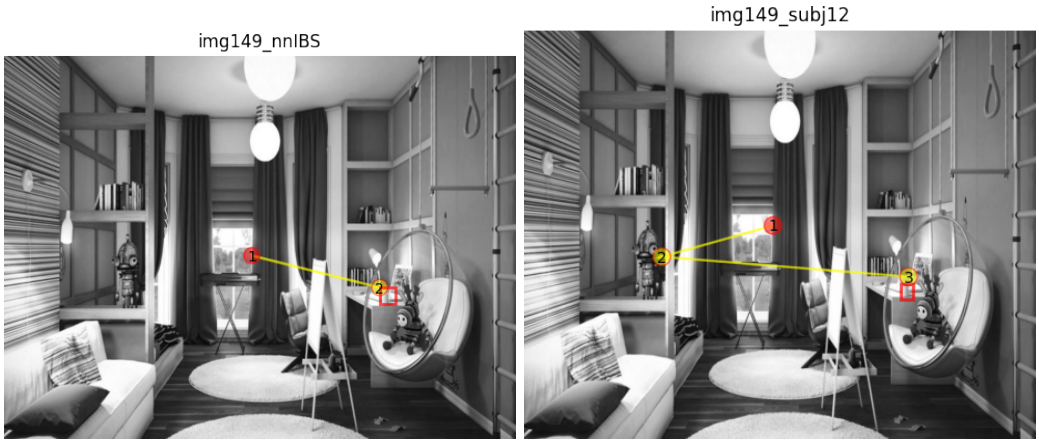


*Fig. 5.15:* A) *scanpath* generado por nnIBS en imagen img149 del *dataset Unrestricted*. B) *scanpath* generado por el sujeto 12 en la imagen img149 del *dataset Unrestricted*. El modelo encuentra el objetivo en una sacada, no queda claro el por qué siendo que el objetivo es pequeño y está alejado del centro.

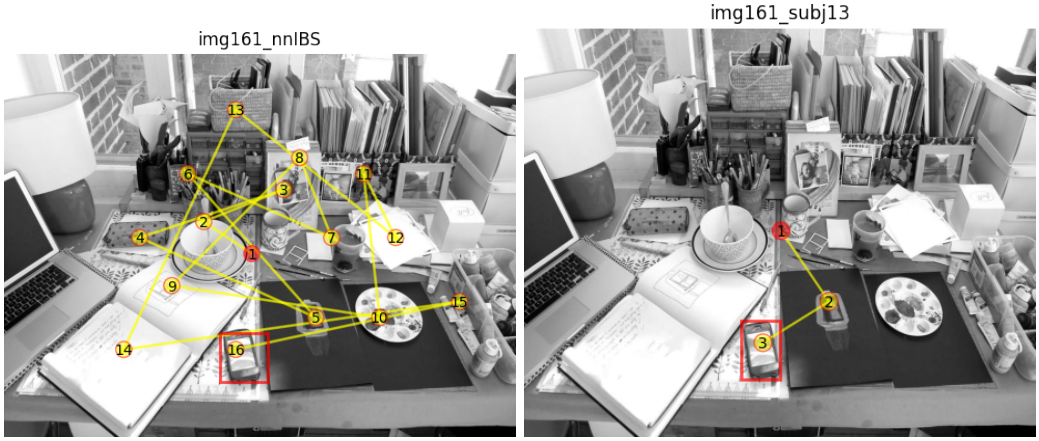


*Fig. 5.16:* A) *scanpath* generado por nnIBS en imagen img161 del *dataset Unrestricted*. B) *scanpath* generado por el sujeto 13 en la imagen img161 del *dataset Unrestricted*. En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. El modelo se distrajo con diversos objetos pero los sujetos no presentaron ese incoveniente.

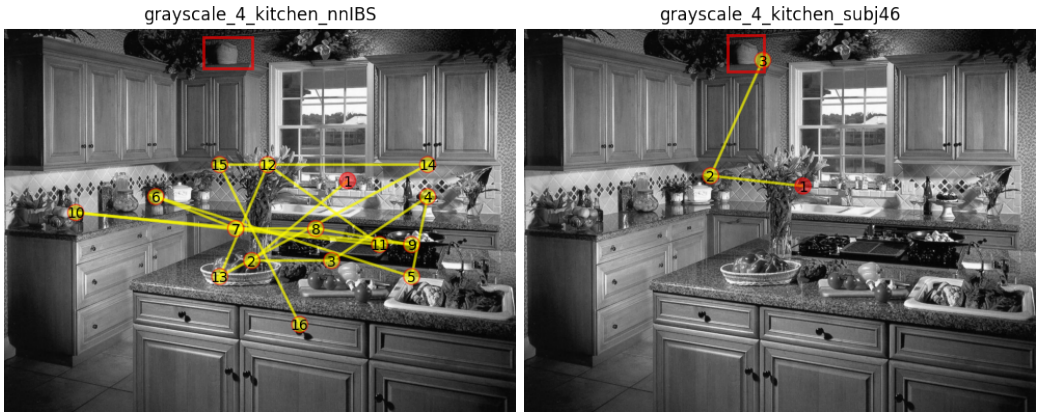


*Fig. 5.17:* A) *scanpath* generado por nnIBS en imagen grayscale_4_kitchen del *dataset Interiors*. B) *scanpath* generado por el sujeto 46 en la imagen grayscale_4_kitchen del *dataset Interiors*. El modelo no encuentra el objetivo y sesga su búsqueda al centro de la imagen.


*Fig. 5.18:* A) *scanpath* generado por nnIBS en imagen grayscale_45_oliva del *dataset Interiors*. B) *scanpath* generado por el sujeto 16 en la imagen grayscale_45_oliva del *dataset Interiors*. El modelo no encuentra al objetivo y sesga su búsqueda hacia el centro de la imagen.

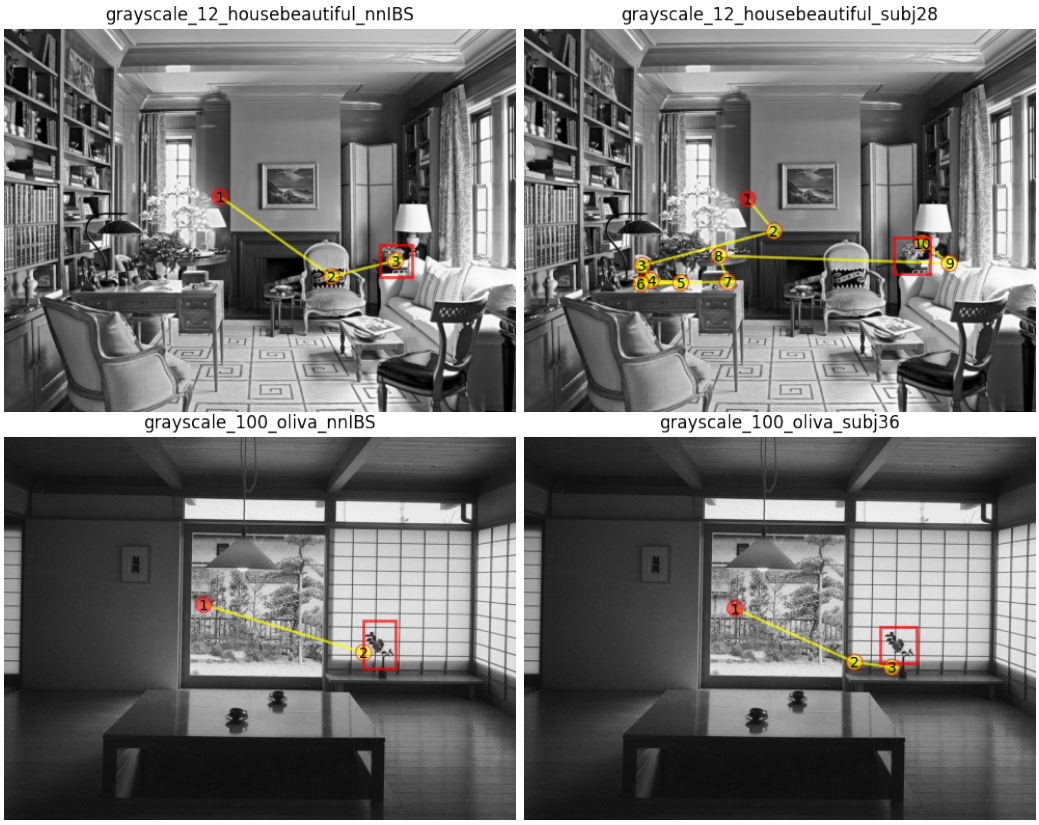


*Fig. 5.19:* Fila superior: *scanpath* generado por nnIBS en imagen grayscale_12_housebeautiful del *dataset Interiors* (izquierda) y *scanpath* generado por el sujeto 28 en la imagen grayscale_12_housebeautiful del *dataset Interiors* (derecha). El modelo encuentra el objetivo en dos sacadas, aún cuando este no se encuentra cerca del centro y hay muchos distractores en la imagen. Fila inferior: *scanpath* generado por nnIBS en imagen grayscale_100_oliva del *dataset Interiors* (izquierda) y *scanpath* generado por el sujeto 36 en la imagen grayscale_100_oliva del *dataset Interiors* (derecha). El modelo encuentra el objetivo en una sacada.


*Fig. 5.20:* A) *scanpath* generado por nnIBS en imagen grayscale_10_housebeautiful del *dataset Interiors*. B) *scanpath* generado por el sujeto 30 en la imagen grayscale_10_housebeautiful del *dataset Interiors*. En el gráfico de MultiMatch el punto correspondiente a esta imagen se encuentra alejado de la diagonal, indicando que el *scanpath* del modelo dista bastante del de los humanos. Al modelo se le dificulta más la tarea que a los sujetos. Hay sacadas largas por parte del modelo.