UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

Tesis Doctoral

Integración escalable de realidad aumentada basada en imágenes y rostros

---

Autor
Nahuel A. MANGIARUA

Director
Jorge S. IERACHE

Director
María José ABÁSOLO

24 de Agosto de 2020

“ Good engineering involves compromise at every turn. A good, working, finished product is never pure by the standards of any one idiom or methodology. The art of good engineering is not the art of discovering and applying the one right idiom over all others. The art of good engineering is to know what your options are, and then to choose your trade-offs wisely rather than letting others choose them for you. ”

“La buena ingeniería involucra compromiso a cada paso. Un producto bueno, funcional y terminado nunca es puro bajo el estándar de ningún estilo o metodología. El arte de la buena ingeniería no es el arte de descubrir y aplicar el estilo que es correcto por sobre todos los otros. El arte de la buena ingeniería es conocer cuales son tus opciones, y luego elegir el punto de compromiso sabiamente en vez de permitir a otros elegir por ti.”

Thomas Becker
Octubre 15, 2007

UNIVERSIDAD NACIONAL DE LA PLATA

Abstract

Facultad de Informática

Doctorado en Ciencias Informáticas

Scalable integration of image and face based augmented reality

by Nahuel A. Mangiarua

Augmented Reality (AR) consists in the creation of an environment in which information and virtual elements fuse with the physical reality. It offers the user an enriched experience without interfering with its natural perception. This thesis develops around the scalable integration of two types of augmentation: arbitrary image based augmentation and face based augmentation with recognition capabilities and estimation of biometric parameters. It is intended to increase the scalability, that is the number of augmentation targets (images or faces) that a single application can have in its database, and to process without relying on an external online service during the exploitation phase.

It is proposed the use of face descriptors with certain morphological similarities to other popular image descriptors. This similarity allows turning the bottlenecks of both image and face based processes into the same problem. Given that the bottleneck identified in the AR process is the search for matches, that is, the search for an image or face in the database, it is proposed to obtain scalability through the use of advanced Approximate Nearest Neighbor (ANN) search algorithms. A series of quantitative performance studies over several ANN algorithms are carried out using a novel benchmarking scheme designed for this particular context. In contrast with other state-of-the-art studies HSNW results in the best alternative.

The design, prototype implementation and evaluation of an scalable architecture that makes use of parallel and asynchronous execution to achieve scalable augmentation of images, face recognition and biometric inference are presented.

UNIVERSIDAD NACIONAL DE LA PLATA

Resumen

Facultad de Informática

Doctorado en Ciencias Informáticas

Integración escalable de RA basada en imágenes y rostros

por Nahuel A. Mangiarua

La Realidad Aumentada (RA) consiste en la creación de un entorno en el que la información y los objetos virtuales se fusionan con la realidad, ofreciendo al usuario una experiencia enriquecida sin interferir con su percepción natural. Esta tesis se centra en la integración de dos tipos particulares de aumentación de la realidad: la basada en imágenes arbitrarias y la basada en rostros humanos, e incluyendo en esta última el reconocimiento facial de individuos y la estimación de parámetros biométricos. Se persigue por una parte la escalabilidad, en el sentido de no tener una limitación en la cantidad de objetivos de aumentación (imágenes o rostros) almacenados en una base de datos, y por otra parte, la independencia de servicios externos en línea a la hora de la explotación.

Se propone utilizar descriptores de rostros con ciertas similitudes morfológicas a otros populares descriptores de imágenes. Esta similitud permite tratarlos como un mismo problema. Dado que el cuello de botella identificado en el proceso de RA es la búsqueda de correspondencias, es decir la búsqueda de una imagen o rostro en la base de datos, se propone obtener escalabilidad mediante el uso de algoritmos avanzados de búsqueda aproximada de vecinos más cercanos o ANN. Se realizan estudios cuantitativos de desempeño de diversas implementaciones de algoritmos ANN utilizando un nuevo esquema de evaluación. Como resultado se establece a HNSW como el algoritmo más apropiado para la tarea.

Se consigue diseñar, implementar y evaluar una arquitectura integrada y escalable de RA basada en imágenes y reconocimiento de rostros con inferencia biométrica, basada en procesamiento paralelo y asíncrono.

A la memoria de Alicia Maccaniani

Agradecimientos

Quiero agradecer primero a Jorge, quien me dirigió, por incentivarme, motivarme, ayudarme, soportarme —en todo sentido— y brindarme sus valiosos consejos. A Maria Jose, mi codirectora por todas sus contribuciones. A los docentes de las materias de postgrado, por el apoyo y la buena onda, sé que no es fácil tenerme de alumno. A Martín por el apoyo logístico en la realización de los cursos y colaborar para que disponga del tiempo para avanzar con mi doctorado.

Por último no quiero dejar de extender mi agradecimiento a los doctores Leonid Boytsov y Masajiro Iwasaki que pese a exceder sus responsabilidades laborales y académicas —y vivir en zonas horarias muy distantes—, asistieron a este extraño en cuestiones técnicas a la hora de probar correctamente algoritmos de búsqueda de vecinos más cercanos aproximados, validar resultados y utilizar sus respectivas bibliotecas de código.

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

ANN	Approximate Nearest Neighbor, vecino más cercano aproximado
FERA	Facial Expression Recognition and Analysis
KP	KeyPoint, sinónimo de POI
ms	Milisegundo
NN	Nearest Neighbor, vecino más cercano
PnP	Perspective-n-Point
POI	Point Of Interest, punto de interés
RA	Realidad Aumentada
RANSAC	Random sample consensus
ROI	Region of interest, región de interés en una imagen
TEPA	Tarjetas de Emergencia Personal Aumentadas

Capítulo 1

Introducción

En la sección 1 de este capítulo introductorio se presenta la motivación mediante una breve introducción al estado del arte de las temáticas de la tesis: Realidad Aumentada basada en imágenes, reconocimiento de rostros y biometría, búsqueda en grandes volúmenes de datos, limitaciones de software disponible y aplicaciones de interés. En la sección 2 se define la problemática; en la sección 3 los objetivos y el alcance de este trabajo de tesis; en la sección 4 se describe la metodología seguida; y por último la sección 5 presenta estructura del resto del presente informe.

1.1 Motivación y áreas temáticas

1.1.1 Definición de Realidad Aumentada

La Realidad Aumentada (RA) se encuentra inmersa en lo que se ha llamado el continuo de la virtualidad, introducido por (Milgram y Kishino 1994) e ilustrado en la figura 1.1.

Figura 1.1. Gráfico del continuo de la virtualidad reconstruido sobre lo publicado por Milgram & Kishino.

Este continuo que se extiende desde el ambiente real que nos rodea hasta un ambiente completamente virtual, artificial. Sobre el mismo, se identifica la región de transición como la zona de Realidad Mixta. Esta zona a su vez puede dividirse en dos regiones. En la primera, partiendo desde el ambiente real, conviven distintos niveles de RA que

agregan elementos virtuales a la realidad física. En la segunda, la cual termina con el pasaje a los ambientes virtuales, se encuentra el espectro de la virtualidad aumentada donde objetos físicos son incorporados a ambientes mayoritariamente virtuales.

La RA consiste en la creación de un entorno en el que la información y los objetos virtuales se fusionan con la realidad, ofreciendo al usuario una experiencia enriquecida sin interferir con su percepción natural. La RA puede ser usada para expandir nuestros sentidos, define una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales, como pueden ser textos, imágenes, audio o videos para la creación de una realidad mixta en tiempo real (Abásolo Guerrero et al. 2011). Con la ayuda de la tecnología, la información sobre la realidad alrededor del usuario se convierte en interactiva y digital, pudiendo ser almacenada y recuperada como una capa de información por sobre la visión normal. La RA no siempre añade elementos al mundo real, sino que también puede ser usada para quitar información al mismo, quitando un objeto físico de la vista que es reemplazado por cierta información (Azuma 1997).

Una aplicación de RA, en la mayoría de los casos, está conformada mínimamente por un conjunto de cuatro elementos básicos que necesitan estar vinculados de cierta manera para que la aplicación logre su cometido. Los elementos en cuestión son:

- Un elemento que capture las imágenes de la realidad que están visualizando los usuarios, como por ejemplo, las cámaras presentes en las computadoras y en los dispositivos móviles.

- Un elemento sobre el que proyectar la integración de las imágenes reales con los contenidos virtuales. Se suelen utilizar las pantallas presentes en los dispositivos móviles y computadoras.

- Una arquitectura de software, embebida o no en un marco de trabajo, cuyo objetivo es el de interpretar la información del mundo real en la cual el usuario está inmerso, generar informaciones virtuales que cada servicio concreto necesite e integrarse de forma adecuada en el entorno.

- Un disparador o elemento que actúe como desencadenante de RA. Se utilizan distintos tipos de elementos tales como: localización (a través del uso de GPS), etiquetas o marcadores especiales con un detalle y contraste determinado, gráficos simples o imágenes arbitrarias con cierta complejidad en cuanto a su composición o incluso objetos reales.

Desde sus inicios la RA se ha reinventado a sí misma adoptando de manera continua nuevas técnicas y dispositivos de hardware a medida que se desarrollan. Parte de cuartos especialmente preparados con sensores y grandes marcadores hace algunas décadas, hasta seguimiento del movimiento del individuo (trayectoria de vuelo), reconocimiento óptico, posicionamiento por GPS o incluso aproximaciones por modelado 3D de ambientes en los últimos años. En (Krevelen, Rick, y Poelman 2010)

se puede encontrar una detallada comparación de las distintas técnicas con sus limitaciones, discutiendo posibles aplicaciones. A su vez, esta tecnología se ha expandido a multitud de campos de aplicación tales como: educación, salud, turismo, marketing y entretenimiento.

1.1.2 RA basada en imágenes

En particular, la RA basada en el reconocimiento de imágenes planas consiste en utilizar como disparador y anclaje al mundo físico, una imagen plana arbitraria que puede ser detectada, permitiendo la incorporación de información virtual de forma espacial y contextualmente coherente con la realidad. A diferencia de las imágenes denominadas marcadores, que poseen un patrón geométrico dominante para facilitar su detección y un código visual embebido para diferenciar uno de otro, la imagen arbitraria puede ser entendida como una fotografía, que captura y representa la realidad de un instante congelado en el tiempo. Si bien la realidad física que nos rodea se encuentra en continuo cambio, muchos de estos cambios son lo suficientemente graduales como para considerarlos constantes durante el tiempo de vida útil de una determinada aplicación que utilice esta tecnología. Se observa en la figura 1.2 un ejemplo de aplicación de este tipo de RA sobre un mural pintado en la vía publica.

Figura 1.2. Ejemplo de RA basada en imágenes arbitrarias.

La RA basada en imágenes depende usualmente de arquitecturas que combinan procesos locales a un dispositivo que analizan cuadros de un flujo de vídeo, con herramientas de autor remotas o en línea. En estos procesos, un usuario del sistema carga y configura un grupo de imágenes planas en una herramienta de autor, asociándolas con los elementos digitales que serán insertados como aumentaciones relativos a cada imagen.

La herramienta de autor analiza cada imagen en busca de puntos de interés (POI) que satisfagan ciertas condiciones que lo hagan fácil de volver a encontrar e identificar. Por cada punto, la herramienta computa y almacena un descriptor pseudo-invariante utilizando algoritmos como el popular SIFT (Lowe 2004) así como ORB (Rublee et al. 2011) o SURF (Bay et al. 2008). Al buscar por POI dentro de los cuadros de un flujo de vídeo, una aplicación separada puede detectar e identificar alguna de las imágenes cargadas por el usuario. Esta identificación a través de la búsqueda de correspondencias entre descriptores de POI puede realizarse de forma autónoma por la aplicación o dependiendo de servicios online relacionados a la herramienta de autor. Con la imagen, el objetivo de aumentación, reconocido se puede entonces aumentar la realidad con el contenido asignado al mismo.

En los últimos años el desarrollo en el área ha sido impulsado tanto por grupos de investigación como por instituciones comerciales privadas, mediante sus productos en forma de frameworks y sistemas de autor.

1.1.3 Escalabilidad y búsqueda de imágenes en grandes volúmenes de datos

La escalabilidad en términos de RA puede naturalmente relacionarse a múltiples aspectos de su ejecución. La acepción utilizada en el contexto particular de este trabajo de tesis, hace referencia específicamente a la cantidad de objetivos de aumentación que una arquitectura, u aplicación que la utilice, pueda contener y aumentar. En este sentido no se considera la cantidad de objetivos (imágenes o rostros) que se estén aumentando en simultáneo sobre un mismo cuadro de vídeo o imagen, sino al tamaño del repositorio de objetivos que un software puede reconocer y aumentar individualmente, sin depender de procesamiento externo.

La búsqueda o identificación de imágenes particulares en un gran volumen de datos es un área de investigación muy activa y de rápida aplicación a nivel productivo como podemos observar con el florecimiento de servicios de búsqueda online inversa incluidos en buscadores web como Google. Es decir, servicios que a partir de una imagen dada como criterio de búsqueda nos proporcionan otras similares o sitios web relacionados. En este contexto, los enfoques actuales consisten en el desarrollo de métodos de búsqueda aproximada predecibles, o en la reducción de dimensionalidad de la entrada o query. Los primeros permiten el uso de entradas de gran tamaño, como los propuestos en (Chum, Philbin, y Zisserman 2008; Grauman y Darrell 2007; G. Shakhnarovich, Viola, y Darrell 2003; Gregory Shakhnarovich, Darrell, y Indyk 2006) al sacrificar de manera predecible la precisión de la búsqueda, mientras que los segundos, buscan reducir el tamaño de la entrada con una cierta pérdida de información, pero permitiendo el cálculo de funciones de distancia entre elementos más robusta y por ende costosa, como los detallados en (Athitsos et al. 2004; Jegou, Douze, y Schmid 2008; Salakhutdinov y Hinton 2009; Torralba, Fergus, y Weiss 2008; Weiss, Torralba, y Fergus 2009).

Este comprende uno de los temas centrales en la propuesta de la tesis, en la cual se buscará una integración de técnicas de búsqueda a escala web existentes que cumpliendo con las restricciones de espacio y tiempo necesarios, en el contexto de la RA, permitan incrementar la escalabilidad, comparada con sistemas actuales en términos de cantidad de imágenes distintas que pueden ser identificadas automáticamente sin el agregado de un marcador adicional. En particular, se propone trabajar sobre el indexado inteligente de descriptores locales tanto para imágenes como para registros biométricos.

1.1.4 Detección y reconocimiento de rostros e inferencia biométrica

La detección de rostros en tiempo real en un flujo de vídeo, aunque aún un área de investigación activa, resulta posible y aplicable desde los aportes introducidos por el famoso trabajo de Viola y Jones (Viola y Jones 2001). En dicho trabajo, la combinación de un preprocesamiento sobre la imagen, la imagen integral, para acelerar el cálculo de sumatorias en sub-áreas junto con la utilización del método de entrenamiento AdaBoost (Freund y Schapire 1995), y la introducción de un esquema de clasificadores jerárquicos en cascada, permitieron por primera vez efectuar detección de rostros en tiempo real con precisión razonable. Siguiendo la tendencia general en el área de visión de computador, en los años posteriores muchos otros mecanismos basados en características (usualmente llamadas con el término en inglés, *features*) complejas o esquemas de aprendizaje de máquina simples se han propuesto, no sólo para aumentar el desempeño sino para incrementar el rango de ángulos y de variación lumínica aceptables, como los presentados en (Jegou, Douze, y Schmid 2008; Wang, Han, y Yan 2009; Waring y Liu 2005; Wu y Nevatia 2007).

En la actualidad, el problema de la detección de rostros es abordado mayormente mediante la aplicación de técnicas avanzadas de aprendizaje de máquina como se presenta en (Dantone et al. 2012), en particular destacan las redes neuronales convolucionales y sus arquitecturas derivadas como presentan (Girshick 2015; Li et al. 2015). Por su parte, (Zhang y Zhang 2010) resume y compara extensivamente trabajos relativamente recientes sobre la detección de rostros. Mientras que la detección facial en tiempo real no representa una gran dificultad técnica dada las prestaciones computacionales actuales como fue demostrado tiempo atrás en (Viola y Jones 2001), la integración de técnicas de reconocimiento de los mismos en el contexto de la RA para usuarios finales (al contrario de grandes sistemas centralizados como los instalados en un aeropuerto) aún reviste un desafío importante dada la limitada capacidad de procesamiento de las computadoras personales o dispositivos móviles.

En cuanto al reconocimiento o identificación de rostros previamente detectados, en los últimos años las aproximaciones han migrado a una fuerte utilización de esquemas de aprendizaje de máquina, en particular, a las redes neuronales profundas. Dichas arquitecturas de redes neuronales, como demuestran (Parkhi, Vedaldi, y Zisserman 2015; Schroff, Kalenichenko, y Philbin 2015; Taigman et al. 2014), pueden ser entrenadas para computar un único descriptor multidimensional a partir de la imagen del rostro de un individuo dado, con cierta robustez a la orientación y cambios en la iluminación.

Por su parte, la inferencia de parámetros biométricos a partir de imágenes es un campo con numerosas especializaciones. En particular, para en la detección de emociones a través del rostro encontramos un gran número de técnicas tanto a nivel experimental como a nivel productivo. (Corneanu et al. 2016) nos ofrece una extensiva introducción y

recorrido por los más importantes aportes hasta la actualidad a la vez que (Valstar et al. 2017) describe los esfuerzos por definir un marco de trabajo y evaluación estandarizado mediante las competencias FERA (Facial Expression Recognition and Analysis). Por otro lado, en («Affectiva» 2020; EmoVu, s. f.; «Kairos» 2020) entre muchos otros, ofrecen tanto servicios web como SDKs con la capacidad de estimar emociones, rangos de edad, etnia, género, etc. a partir de un rostro en una imagen o vídeo.

1.1.5 Limitaciones de herramientas de software

Los desarrolladores disponen de bibliotecas de programación o frameworks que proveen de acceso a funcionalidad propia de RA. En el universo de los frameworks disponibles para desarrolladores, se encuentra a grandes compañías como Apple con ARKit o Google con ARCore, entre otros. La mayoría de estos frameworks integran múltiples facilidades para desarrolladores, algunos incluyendo la aumentación de imágenes arbitrarias, otros la aumentación de rostros humanos, pero enfocándose principalmente en la aumentación de superficies y de ambientes en tres dimensiones.

Además de herramientas para desarrolladores con conocimientos de programación, hay disponibles herramientas de autor que ponen a disposición de usuarios finales no expertos la posibilidad de desarrollar aplicativos que hagan uso de la RA en un contexto delimitado. En este grupo se encuentra a («Augment» 2020; «Aumentaty» 2020; «Aurasma» 2020; «Zappar» 2020), entre otras herramientas de autor. Cabe destacar que en su amplia mayoría, estas herramientas fuerzan a los usuarios a mantener una conexión constante con el servicio online al momento de la explotación, es decir, la utilización de la aplicación que aumenta la realidad.

Muchos frameworks como sistemas de autoría y asistencia a la creación de aplicaciones de RA se enfocan en simplificar el acceso a la tecnología por parte de usuarios finales, salvando las limitaciones inherentes de escalabilidad al particionar los contenidos en unidades de tamaños reducidos que deben ser accedidos por separado, o dependiendo de un servicio online externo.

Sin embargo, los frameworks y herramientas disponibles hasta la fecha no hacen foco en la escalabilidad offline de las aplicaciones generadas, obligando tanto a usuarios como a desarrolladores a depender de forma constante de servicios externos en línea, muchas veces con un costo monetario adicional, los cuales pueden incurrir en dificultades de conectividad o elevada latencia, perjudicando la experiencia del usuario final. A su vez, ninguna de las alternativas permite la integración de RA basada en imágenes arbitrarias con tecnologías de reconocimiento facial escalable que permita el desarrollo de aplicaciones reactivas a la identidad o datos biométricos inferidos de las personas. La tabla 1.1 resume las características mencionadas para los principales

frameworks de RA de la actualidad así como de la propuesta cuyos objetivos se detallan en la próxima sección.

Framework	ARKit	ARCore	Vuforia	ARToolkit	Propuesta
Imágenes Arbitrarias	SI	SI	SI	SI	SI
Detección de rostros	SI	SI	NO	NO	SI
Reconocimiento de Rostros	NO	NO	NO	NO	SI
Cantidad de objetivos	No informado	1000	1000	No informado	1000+ (deseable)
Inferencia biométrica	NO	NO	NO	NO	SI
Costo	Entorno de desarrollo Apple	Gratuito	Comercial	Gratuito	Gratuito

Tabla 1.1. Resumen de las capacidades de aumentación de imágenes arbitrarias y reconocimiento de rostros de los frameworks más renombrados.

Según sus respectivas documentaciones a la fecha, si bien frameworks como ARKit («ARKit» 2020) y ARCore («ARCore» 2020) proporcionan capacidades de detección facial, con foco en la determinación de poses y mapeo 3D, ninguno de ellos integra la posibilidad de reconocimiento de individuos. Aunque la detección y determinación de la posición de rostros humanos es un paso previo necesario para el reconocimiento, esta última tarea reviste una gran complejidad y enfoques particulares que no son abordados por los frameworks mencionados.

Tanto Vuforia («Vuforia» 2020) como ARToolkit («ARToolKit» 2020) por su parte, hacen foco en el trabajo con marcadores e imágenes arbitrarias, con escalabilidad suficiente pero limitada, y sin capacidades de detección o reconocimiento de rostros humanos como puede observarse en la documentación citada. Particularmente, Vuforia complementa su framework con un servicio en línea que permite la detección de imágenes de forma remota a través de sus servidores. Dicho servicio naturalmente depende del acceso a la internet e incurre en costos monetarios por cada uso.

Por otro lado, la incorporación de inferencia de datos biométricos a partir de las imágenes observadas queda por fuera de las arquitecturas de los frameworks de RA mencionados y a completa responsabilidad de los desarrolladores.

1.1.6 Aplicaciones de RA basada en imágenes y rostros

Existen claros ejemplos de la potencialidad de la RA basada en grandes cantidades de imágenes arbitrarias o rostros humanos con capacidades de inferencia biométrica. Desde aplicaciones de recorrido virtual en grandes museos de arte o historia natural, hasta la aumentación de cada página de manuales didácticos junto con profesores y alumnos; el turismo en general, así como la arquitectura mediante la aumentación de fachadas de edificios o postales; aplicaciones de interacción social y didáctica entre miembros de una comunidad; aplicativos para las fuerzas de seguridad en búsqueda activa y constante de individuos con pedido de captura o sintomatologías preocupantes en contextos de pandemia; aplicaciones de asistencia personal o social que permitan una interacción colaborativa entre individuos en la vía pública.

A través de estos ejemplos, se evidencia la necesidad de una integración escalable de las tecnologías que facilite el desarrollo y evite las restricciones de usabilidad y los costos monetarios derivados de la dependencia con servicios externos online.

Entre los desarrollos propios de aplicaciones RA se hace mención a (J. S. Ierache, Mangiarua, y Bevacqua 2014) y (J. Ierache et al. 2015) donde se presenta un prototipo de sistema que hace uso de la biblioteca Vuforia y permite la generación, distribución y explotación de Catálogos de RA por parte de usuarios finales. Dichos catálogos se componen por un conjunto de marcadores que son aumentados con información provista por los usuarios al momento de su creación, la cual es visualizada en forma de RA utilizando una aplicación para teléfono inteligente que descarga la información de manera dinámica una sola vez y luego continúa con su funcionamiento de forma autónoma indefinidamente.

Existen también frameworks o herramientas especializadas, una suerte de híbridos entre biblioteca y herramienta de autor que permiten a usuarios finales o a desarrolladores no especializados en la RA crear fácilmente instancias de aplicaciones para un contexto específico. En esta categoría, para el contexto de la educación, se encuentran desarrollos propios como los publicados en (J. Ierache, Mangiarua, y Becerra 2014; N. Mangiarua et al. 2014) en los cuales se busca facilitar al personal docente la aumentación de distintos tipos de materiales con contenidos didácticos interactivos que permitan incrementar el interés de los estudiantes. En cuanto a la creación de juegos educativos, también llamada gamificación de la educación, se encuentran otros desarrollos propios como (Verdicchio et al. 2016; J. Ierache et al. 2018) donde se busca facilitar la absorción de contenidos educativos mediante actividades lúdicas que puedan realizarse incluso fuera del contexto de las aulas.

En (J. Ierache, Mangiarua, et al. 2016; J. Ierache, Verdicchio, et al. 2016; Montalvo et al. 2017) se demuestra cómo herramientas de RA pueden ser utilizadas de forma efectiva para asistir en el campo de la medicina, más específicamente en la emergentología, al proporcionar un acceso rápido a datos relevantes de un paciente, con

la posibilidad de inferir un estado de alerta previo mediante el uso de un sistema basado en conocimiento incorporado. Entre los casos de uso estudiados, también se contempla la utilización del sistema por parte del público general al momento de asistir a un individuo colapsado en la vía pública. Para sistemas de esta naturaleza, la capacidad de actuar aumentando directamente a un gran número de individuos por su rostro resulta de gran importancia.

1.2 Definición del problema

Tanto la RA basada en imágenes arbitrarias como el reconocimiento facial son utilizados por un número de sistemas, aplicaciones o frameworks en diversos campos de aplicación. Sin embargo, no existen en la actualidad frameworks cuyas arquitecturas integren la capacidad de reconocer imágenes y rostros de manera escalable, es decir con un número elevado de objetivos a aumentar. A su vez, ningún framework integra la capacidad de efectuar inferencia biométrica de información a partir de las imágenes percibidas, en particular la de rostros humanos. Estas capacidades se hacen accesibles a los desarrolladores en forma de frameworks separados o servicios en línea, sin integración con la RA, con un costo monetario adicional y susceptibles a problemáticas derivadas de las condiciones de conectividad como elevada latencia o pérdida momentánea de conexión, afectando negativamente la experiencia de un usuario final.

Si bien las técnicas de búsqueda de imágenes en grandes volúmenes de datos y los algoritmos que las facilitan son campos de investigación activos, aún no existe un gran número de trabajos enfocados en aplicar estas técnicas a incrementar la escalabilidad directa de la RA por reconocimiento de imágenes o reconocimiento facial. Es decir, aumentar la cantidad de imágenes o rostros que pueden ser identificados en una misma aplicación sin dependencia de un servicio online externo. Aunque (Takacs et al. 2008) describe la aplicación de técnicas de búsqueda a escala web en imágenes en el contexto de la RA, utiliza la técnica no para la determinación y anclaje de elementos virtuales sino para la recuperación de imágenes similares a lo observado.

1.3 Objetivos y alcance

Se plantea como objeto principal de este trabajo de tesis el diseñar una arquitectura escalable de RA basado en el reconocimiento visual monocular de imágenes y rostros humanos, con capacidad de inferencia de datos biométricos, que no haga uso de servicios externos para su etapa de explotación; en función a la propuesta comparativa presentada en la Tabla 1.1.

En este orden se plantean como objetivos particulares:

- Establecer los procesos y sus pasos necesarios para efectuar aumentación de imágenes, detección y reconocimiento de rostros e inferencia de información biométrica.
- Analizar comparativamente la complejidad computacional teórica y la carga de procesamiento empírica de cada paso de los procesos de RA, considerando en particular distintas variaciones de algoritmos disponibles para la búsqueda y descripción de POI.
- Diseñar una arquitectura que integre los procesos descritos, contemplando la ejecución paralela y/o asíncrona, identificando el o los pasos que resulten en el principal cuello de botella con respecto a la escalabilidad.
- Analizar comparativamente la velocidad y precisión de los algoritmos aplicables para aliviar o solventar el o los cuellos de botella detectados.
- Diseñar criterios de evaluación y conjuntos (sets) de datos de prueba para los algoritmos aplicables a los cuellos de botella que sean representativos del dominio de explotación propuesto.
- Incorporar al diseño un mecanismo de integración abierto que facilite el agregado futuro de algoritmos de inferencia biométrica adicionales en la arquitectura propuesta.

Se proyecta diseñar una arquitectura y desarrollar un prototipo que integre de forma escalable la RA basada en imágenes arbitrarias, el reconocimiento facial y la inferencia de datos biométricos.

Se analizará la complejidad computacional para identificar los cuellos de botella, analizando y comparando los algoritmos específicos existentes a fines de solventar las limitaciones de escalabilidad, evitando dependencias de sistemas externos durante la fase de explotación.

Particularmente se pretende determinar la idoneidad de las técnicas descritas en el resumen del estado del arte en orden a su funcionamiento en tiempo real dados los requerimientos específicos de la RA.

Mientras que no se busca competir con sistemas existentes en términos del refinamiento y calidad que estos han logrado con años de desarrollo continuo, se pretende demostrar que una integración de las tecnologías propuestas es posible, mientras se supera su escalabilidad.

Quedará por fuera del alcance de este trabajo de tesis el estudio de las cuestiones y particularidades de cada algoritmo de búsqueda de POI, descripción de POI, detección

de rostros, reconocimiento de rostros e inferencia biométrica utilizados. Se considerará sólo su aptitud relativa en términos de carga de procesamiento mínima requerida para alcanzar el objetivo de RA deseable bajo condiciones físicas (iluminación, oclusión, etc) esperables en un entorno de explotación parcialmente controlado.

1.4 Metodología utilizada

Este trabajo de tesis se encuentra encuadrado dentro de lo que se denomina como investigación en la ciencia del diseño (DSR por sus siglas en inglés). Dentro de este paradigma de investigación se considera al artefacto como pivote central del proceso de investigación, como explican (Gregor y Hevner 2013) entre otros. Este artefacto se sustenta sobre una base teórica de núcleo como plantean (Jones y Gregor 2007) mientras que a su vez constituye en sí mismo un aporte a reforzar o generar nuevas bases teóricas de distintos niveles de abstracción.

En este orden, el método de trabajo utilizado consta esencialmente de los siguientes pasos:

1. Estudiar el proceso de RA basado en imágenes: su complejidad teórica y pruebas experimentales para detectar cuellos de botella, en particular en torno a escalabilidad. También determinar descriptores aptos a utilizar.
2. Estudiar el proceso de RA basado en rostros con reconocimiento e inferencia biométrica: su complejidad teórica, detectar cuellos de botella tanto en escalabilidad como exigencias para tiempo real y determinar algoritmos a utilizar de acuerdo a sus descriptores.
3. Proponer arquitectura integrada de RA basada en imágenes y rostros. Optimizar mediante paralelización y asincronismo.
4. Realizar comparativa de algoritmos para aumentar la escalabilidad de la arquitectura propuesta.
5. Implementar la arquitectura en un prototipo demostrados y probar su desempeño.
6. Publicar resultados intermedios en todas las partes del proceso.

En particular, se adapta e incurre en un proceso iterativo en el cual ciertas decisiones de diseño son basadas en los resultados de evaluaciones con fuertes componentes cualitativos, a la vez que decisiones de diseño generan la necesidad de nuevos estudios cuantitativos que las evalúen.

1.5 Estructura de la tesis

Luego de la presente introducción que familiariza al lector en el contexto de la tesis, en el capítulo 2 se presenta el proceso de RA basada en imágenes. Desde su descripción general hasta una evaluación de desempeño, pasando por el análisis teórico de su complejidad computacional en cada paso.

El capítulo 3 describe de manera análoga el proceso de RA basado en el reconocimiento de rostros y la inferencia de información biométrica. Se presenta su estructura y elección de técnicas o algoritmos mientras se analiza la complejidad computacional teórica de cada paso.

El capítulo 4 describe la solución propuesta en forma de una arquitectura escalable que integra la RA basada en imágenes con el reconocimiento facial y la inferencia de información biométrica. Plantea los puntos sobre los cuales se toman decisiones de diseño para organizar el flujo de ejecución así como la distribución de la carga de procesamiento en múltiples hilos de CPU con uso de asincronía mientras se presentan los principales objetos de la implementación prototipo.

El capítulo 5 presenta los algoritmos de búsqueda aproximada de vecinos más cercanos en el contexto de la RA, tanto para la aumentación de imágenes como para el reconocimiento de rostros. Se discute el esquema de evaluación general para esta familia de algoritmos y se propone un esquema de evaluación específico bajo el cual se realizan una serie de estudios cualitativos sobre un número de algoritmos prometedores con el fin de identificar el más adecuado para la tarea.

El capítulo 6 resume y discute los resultados y las decisiones tomadas para el diseño de la arquitectura de integración escalable, presentando una validación sobre los tiempos de ejecución del prototipo completo bajo un nivel de carga elevado.

El capítulo 7 presenta una herramienta de autor que utiliza el concepto de templates para facilitar el trabajo de los usuarios frente al incremento de escala que permite la arquitectura propuesta así como los puntos de unión con la misma.

Finalmente, el capítulo 8 presenta las conclusiones y aportes de este trabajo de tesis a la vez que discute futuras líneas de trabajo que hacen a la continuidad de esta investigación.

Este documento además cuenta con seis apéndices. El apéndice A indica cómo acceder de manera digital al repositorio de código del proyecto completo. El apéndice B presenta la generación de datos utilizados en la comparativa de algoritmos ANN del capítulo 5. El apéndice C presenta el eje central del código utilizado a la hora medir el desempeño de algoritmos de ANN del capítulo 5. El apéndice D presenta el acceso a los datos base y el código para la confección del video simulado para pruebas del capítulo 5. El apéndice E presenta breves vídeos mostrando las capacidades de la arquitectura

integrada. Por último el apéndice F detalla una línea de trabajo en progreso que hace uso del producto de esta tesis en el contexto de la asistencia médica.

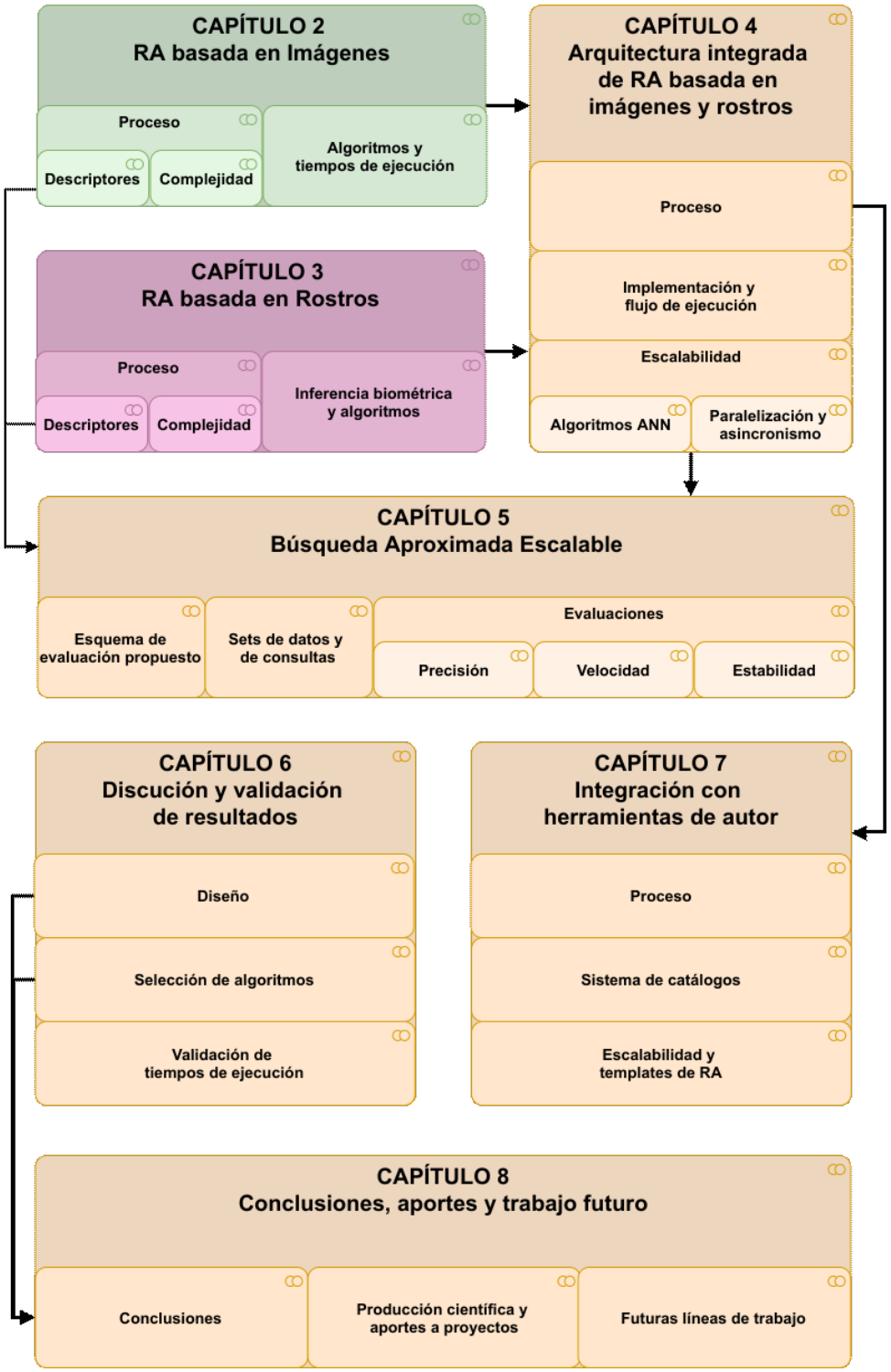

Figura 1.3. Diagrama de bloques ilustrando los capítulos de este escrito y sus principales temáticas.

Capítulo 2

Proceso de RA basada en imágenes

En el presente capítulo se definen los pasos necesarios para lograr la funcionalidad de aumentación de imágenes. Se define el proceso en la sección 1, analizando la complejidad teórica de cada paso en la sección 2. A continuación en la sección 3 se describen las cuestiones relevantes de los algoritmos de búsqueda de POI y extracción o descripción de características. La sección 4 finaliza el capítulo con un análisis de desempeño para este proceso y sus primeras conclusiones.

2.1 Descripción del proceso

Partiendo de objetivos de aumentación, el proceso o secuencia de pasos para la aumentación de imágenes consta de dos bucles independientes pero relacionados. El mismo se observa en la figura 2.1 y fue publicado en (N. Mangiarua, Ierache, y Abásolo Guerrero 2019).

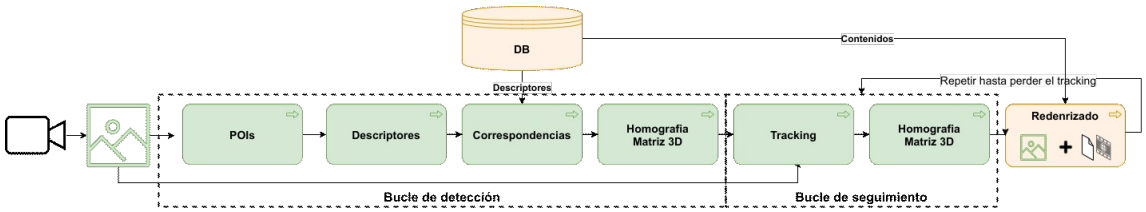

Figura 2.1. Diagrama conceptual del proceso para la aumentación basada en imágenes arbitrarias.

Bucle de detección

1. Detección de POI (también llamados features)
   Se ejecuta el algoritmo seleccionado (SIFT, ORB, etc) para buscar un número determinado de POI o features en un cuadro del vídeo proveniente de la entrada.
2. Cálculo de descriptores para cada POI (también llamado extracción de características)

Se ejecuta el algoritmo seleccionado (SIFT, ORB, FREAK, etc) para computar un descriptor por cada POI encontrado en el paso anterior.

3. Búsqueda de correspondencias
   Se buscan correspondencias entre descriptores de imágenes de la base de datos de imágenes objetivo y las del cuadro de vídeo actualmente siendo procesado.
4. Cálculo de Homografía y Matriz 3D
   Se computa una homografía con las correspondencias encontradas en el paso anterior. Los puntos que satisfacen la homografía son utilizados para estimar la matriz de transformación 3D.

Bucle de seguimiento

1. Tracking
   Con la información de la homografía del bucle de detección se buscan y computan nuevos descriptores en la región de la imagen detectada y se inicializa un algoritmo de seguimiento que actualiza la posición de cada punto sin necesidad de volverlo a buscar.
2. Cálculo de Homografía y Matriz 3D Idéntico al homónimo del bucle de detección.

Si bien el procesamiento requerido para la detección de una imagen objetivo de aumentación se puede ajustar a lo requerido para mantener una experiencia de usuario satisfactoria, el mismo puede incurrir en casi la totalidad del presupuesto temporal asignado. Es por esto que el proceso se divide en dos bucles independientes pero relacionados, el de detección y el de seguimiento. Estas series de pasos se ejecutan sobre una aplicación que puede correr sobre un dispositivo móvil, la cual captura un flujo de vídeo utilizando una cámara monocular.

En el bucle de detección, esta aplicación busca primero por puntos de interés (POI), también llamados puntos clave (KeyPoints o KP), utilizando los mismos algoritmos que se utilizaron al crear la base de datos de imágenes objetivo.

En segundo lugar, los puntos así encontrados son descritos con los mismos algoritmos que se utilizaron al crear la base de datos de imágenes objetivo.

El tercer paso del proceso toma los descriptores obtenidos y los coteja contra todos los almacenados previamente cuando el usuario carga las imágenes que desea aumentar. Los emparejamientos resultado son entonces filtrados, removiendo aquellos que no cuadren dentro de un plano sujeto a transformaciones afines.

Como cuarto y último paso de este bucle, a partir de los emparejamientos entre puntos obtenidos de las imágenes cargadas por el usuario y aquellos obtenidos de un cuadro del flujo de vídeo, se pueden calcular dos valores. En primer lugar se reconoce cuál de

todas las imágenes cargadas en la base de datos es la que efectivamente está siendo capturada como parte del flujo de vídeo, ya que cada descriptor es almacenado junto con la identidad de la imagen que lo origina. En segundo lugar se estima una matriz que describe el conjunto de transformaciones afines que asocian la posición actual de la imagen detectada en el mundo real con el origen del marco axial de la cámara a través del algoritmo denominado Perspective-n-Point (PnP). Esta matriz puede ser utilizada por sistemas de gráficos 3D para dibujar elementos virtuales sobre el flujo de vídeo posicionados de forma coherente.

Cabe destacar que, dado un POI de una imagen detectada en el flujo de vídeo, el descriptor computado no será idéntico al computado a partir de la imagen original cargada por el usuario en la herramienta de autor. Esto se debe a que los descriptores utilizados no son perfectamente invariantes ni completamente robustos a las distorsiones causadas más comúnmente por cambios irregulares en la iluminación, aberraciones ópticas o aberraciones cromáticas de los lentes de las cámaras, entre otras. Se experimentará para determinar los efectos que estas distorsiones inevitables puedan tener en el desempeño de distintas partes del proceso.

El bucle de seguimiento, mediante la utilización de algoritmos especializados en el seguimiento de puntos como el clásico algoritmo de flujo óptico local de (Lucas y Kanade 1981), es capaz de mantener y actualizar la posición de los objetivos de aumentación ya detectados por una fracción del costo computacional.

2.2 Detectores y descriptores de POI

Existen en la actualidad multitud de algoritmos utilizables como detectores y descriptores de POI que pueden dividirse en dos grandes grupos: los que producen descriptores continuos comparables con la distancia Euclidiana, y aquellos que producen descriptores binarios discretos comparables con la distancia Hamming o similares.

En particular se consideran los algoritmos:

- Continuos (flotantes):
  - SURF (Bay et al. 2008)
  - FAST (Rosten y Drummond 2006)
  - STAR (Agrawal, Konolige, y Blas 2008)
  - FREAK (Vandergheynst, Ortiz, y Alahi 2012)
- Discretos (binarios):
  - ORB (Rublee et al. 2011)

- BRISK (Leutenegger, Chli, y Siegwart 2011)

Como campo de investigación activo, nuevas aproximaciones y mejoras son publicadas con regularidad, en particular (Tareen y Saleem 2018) presentan un estudio comparativo no exhaustivo de la calidad y velocidad de casi lo totalidad de los algoritmos listados con anterioridad, pero con el foco en la tarea de costura de imágenes.

El algoritmo SIFT (Lowe 2004) con su descriptor continuo de 128 flotantes es ampliamente conocido, el mismo otorga el mayor nivel de robustez pero a costo computacional que supera ampliamente al resto y por eso no se considera.

Algoritmos como FAST, ORB y BRISK, incluyen tanto un detector como un descriptor. En el caso de FAST el descriptor se descarta de este estudio por no ser invariante a las rotaciones mientras que en el caso de ORB su descriptor no se recomienda por su tolerancia a cambios de tamaño de la imagen (distancia a la cámara) más limitada. Por esto se utilizan combinados con descriptores como FREAK.

Los dispositivos de captura de vídeo instalados en dispositivos móviles o computadoras personales trabajan en un rango de 24 a 30 cuadros por segundo, salvo excepciones de equipos profesionales o semiprofesionales. Esto significa que, si se pretende ejecutar estos algoritmos sin afectar la velocidad de cuadros por segundo percibida por el usuario, los mismos deben tardar en el peor de los casos, menos de 40 ms en completar.

2.3 Complejidad computacional

Con el objetivo de escalabilidad en mente, resulta necesario caracterizar la complejidad computacional relativa de todo el proceso a través de mediciones experimentales sobre cada paso. Una vez identificados los pasos dominantes, se deberán analizar las alternativas disponibles en la literatura, definir criterios de evaluación adecuados al contexto, seleccionando aquellos algoritmos que ostenten la menor complejidad.

Para el procesamiento de imágenes arbitrarias la complejidad de los algoritmos es la siguiente:

1. Algoritmos de detección de POI y extracción de características, dependen de:
    - Principalmente de la resolución en píxeles WxH de la imagen de entrada. Dicha resolución estará acotada por encima por la resolución del dispositivo hardware que capture el flujo de vídeo.
    - En menor medida de la cantidad N de puntos que se obtienen o el máximo que se fija si el algoritmo lo permite.

2. Algoritmo de cotejamiento, emparejamiento o búsqueda de vecinos más cercanos, dependen de:
    - El tamaño T de los descriptores en dimensiones (bits para binarios, números flotantes para continuos), los cuales son fijos para cada algoritmo.
    - De la cantidad N de descriptores computados sobre el cuadro del flujo de vídeo.
    - De la suma M de descriptores de cada imagen cargada por el usuario.
3. Algoritmos de homografía, dependen de:
    - La cantidad E de puntos emparejados en el paso anterior. Esta cantidad de puntos se encuentra acotada por encima por la cantidad N de puntos fijada en los algoritmos de detección de POI.
4. Algoritmo de estimación de matriz de transformación, dependen de:
    - La cantidad de puntos filtrados F, el cual estará limitado por encima por la cantidad N de puntos fijada en los algoritmos de detección de POI.

De este análisis teórico de alto nivel de la complejidad de los pasos del proceso, y teniendo en cuenta el objetivo del incremento de la escalabilidad en términos de cantidad de imágenes y rostros, se desprende que el punto crítico o cuello de botella deberá eventualmente recaer en el paso de búsqueda de correspondencias al incrementar la escala.

2.4 Evaluación de desempeño

2.4.1 Objetivo del estudio

Dado un estimado teórico de la complejidad de cada paso del subproceso basado en imágenes, se propone un estudio para tomar una muestra de los tiempos de procesamiento efectivo de cada paso previamente publicado en (N. Mangiarua, Ierache, y Abásolo Guerrero 2019) que permita lograr dos objetivos puntuales.

En primer lugar, se busca determinar cuáles de los algoritmos mencionados son capaces de permitir la detección de una imagen objetivo dentro de una escena de prueba con características típicas de la explotación de la RA mientras el orden de magnitud de su tiempo de ejecución permita el procesamiento lineal en tiempo real (menos de 40 ms).

En segundo lugar, se busca obtener una medida relativa del tiempo de procesamiento de cada paso dadas las condiciones mencionadas. Es decir, evidenciar cómo se distribuye

la carga de procesamiento entre los pasos y cómo reacciona esta a la variable de cantidad de POI.

2.4.2 Diseño del estudio

Dada la naturaleza de ejecución en tiempo real de la RA, para comparar la idoneidad de cada algoritmo o combinación de algoritmos en la tarea, es necesario contrastar el tiempo de procesamiento que requieren contra la calidad de sus resultados. Para ello, se toma una imagen base que será el objetivo a detectar, como las que cargaría el usuario en un sistema completo, ilustrada en la figura 2.2. Esta imagen fue especialmente seleccionada por ser un buen objetivo de aumentación, es decir, reviste las características necesarias para el correcto funcionamiento de los algoritmos testeados como son el poseer zonas de alto contraste, líneas definidas y patrones irregulares entre otras. La misma cuenta con una resolución de 800 x 560 píxeles.

Luego se toma otra imagen, la de evaluación, que contenga a la imagen objetivo en un contexto realista, como se observa en la figura 2.3. En este caso la imagen objetivo se encuentra parcialmente ocluida, y con un ángulo de rotación cercano a los 45 grados lo cual representa un caso plausible pero difícil para un motor de RA basada en imágenes arbitrarias. La imagen de evaluación cuenta con una resolución de 1200 x 675 píxeles.

A continuación, se ejecuta el bucle de detección del subproceso basado en imágenes cuyo diagrama de actividad se detalla en la figura 2.5, para cada uno de los algoritmos con los que se cuenta con alternativas a estudiar. El resultado visual de cada ejecución se observa en la figura 2.4. Los algoritmos fueron seleccionados por la disponibilidad de una implementación estable en C++ integrada a la biblioteca OpenCV bajo la interfaz común cv::Feature2D al momento de realizado el estudio. Dichas implementaciones pueden encontrarse en los módulos features2d («OpenCV: 2D Features Algorithms» 2020) y xfeatures2d («OpenCV: Experimental 2D Features Algorithms» 2020). En el código de prueba se utiliza el algoritmo seleccionado para encontrar y calcular POI los cuales luego se utilizan para computar un recuadro envolvente a la imagen objetivo detectada.

Figura 2.2. Imagen objetivo que será buscada durante el experimento.

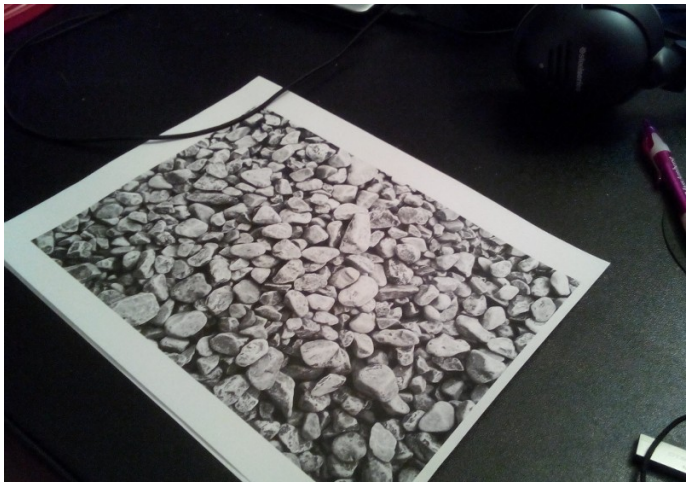

Figura 2.4. Imagen de evaluación, conteniendo a la imagen objetivo en perspectiva y con oclusión parcial.

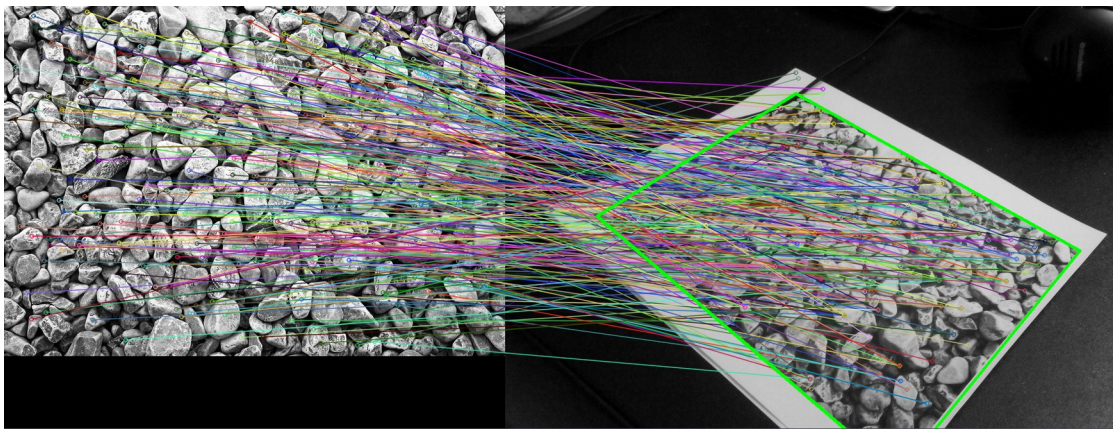

Figura 2.3. Resultado visual de la ejecución de la evaluación con un algoritmo dado.

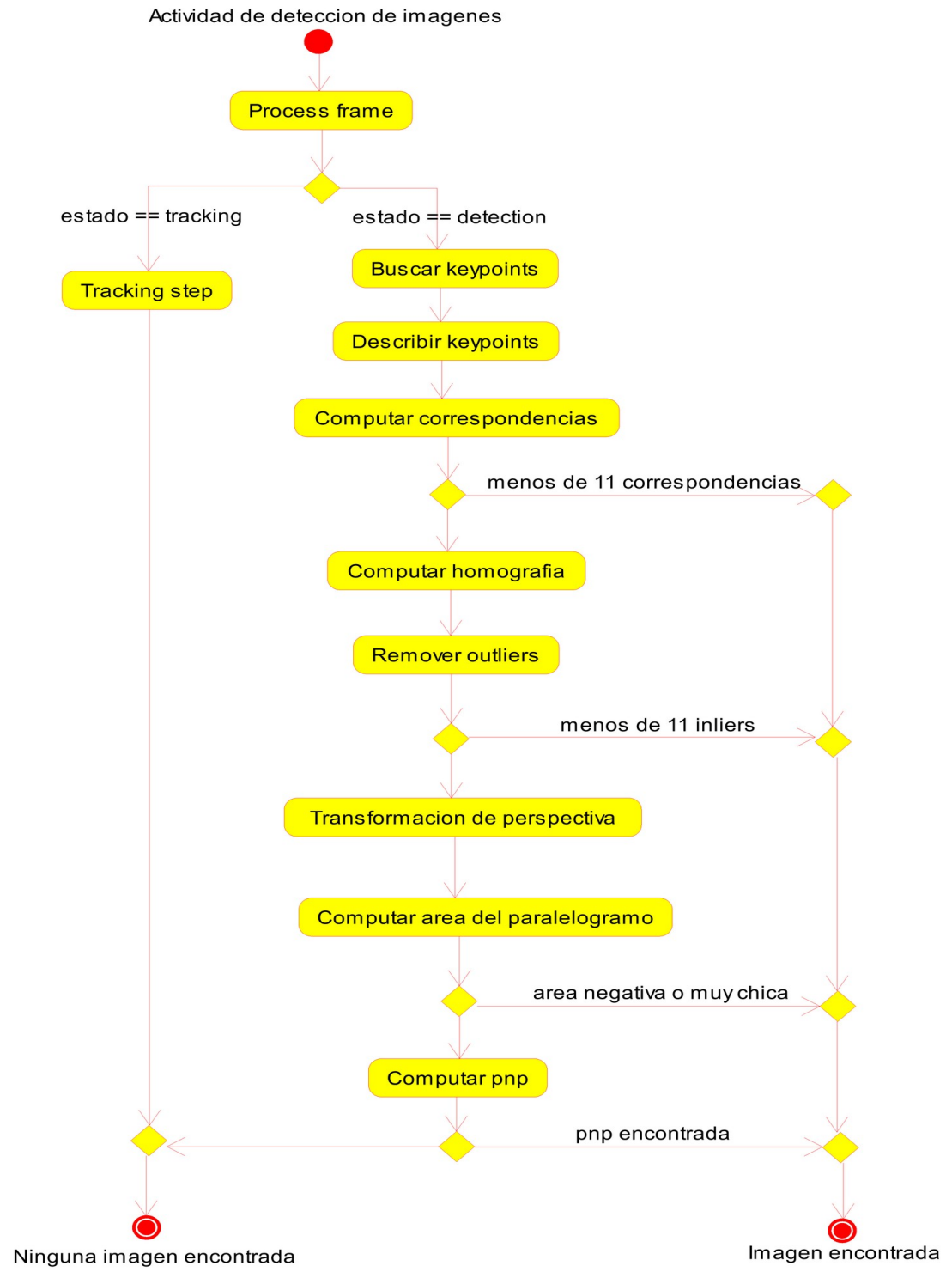

Figura 2.5. Diagrama de actividad del bucle de detección del subproceso de imágenes.

Cada algoritmo cuenta con un número de parámetros que afectan tanto sus resultados como la velocidad de ejecución, en cada caso, se ajustan los parámetros lo mínimo posible de forma tal que el tiempo de ejecución se reduzca a la vez que los resultados obtenidos se mantengan satisfactorios. La satisfacción de los resultados se determina empíricamente observando si el recuadro envolvente a la imagen objetivo está correctamente situado. Dado que la RA se construye para ser percibida por los humanos a través de sus sentidos, no resulta en un aporte el calcular la precisión a un nivel que sea imperceptible a la vista.

En todos los casos, el código es ejecutado en una computadora portátil con procesador Intel Core i7 4810MQ corriendo a 2.8Ghz. Los tiempos son medidos en microsegundos y reportados en milisegundos.

2.4.3 Resultados y conclusiones

Como resultado de la ejecución del código de pruebas para cada algoritmo o combinación de ellos, se observa en la tabla 2.1 la cantidad de POI y los tiempos de ejecución de cada paso, tanto en términos absolutos como relativos.

En línea con los resultados publicados en (Rublee et al. 2011) y los valores actualizados que presentan (Tareen y Saleem 2018), se observa que ORB proporciona una buena relación entre calidad y velocidad de ejecución por debajo de los 40 ms, en especial cuando se lo aparea con el descriptor FREAK.

De los datos se desprende que FAST necesita de un gran número de POI para alcanzar un resultado satisfactorio debido a la baja calidad de los mismos, por debajo del 2% de correspondencias. Esto incurre en un alto tiempo para la descripción del total de esos puntos y todos los pasos subsiguientes.

Por su parte, si bien el algoritmo SURF fue diseñado con el objetivo de tiempo real en mente, las restricciones temporales más acotadas de este estudio junto con la imposibilidad de controlar la cantidad de POI a detectar de forma explícita lo vuelven un candidato pobre para el proceso de RA en cuestión. Aunque es posible ajustar sus parámetros para reducir los tiempos, en particular el Hessian threshold, la validación cruzada con otra imagen objetivo indica que dicho ajuste sólo resulta válido para la imagen objetivo original y por lo tanto no puede ser utilizado en el contexto general. Paradójicamente, la alta calidad de los POI y descriptores SURF, con un 95% de coherencias, implica que el algoritmo de homografía, utilizando la técnica RANSAC deba procesar un gran número de elementos, incrementando su tiempo de procesamiento de manera significativa. Aunque se considerara recortar esta cantidad de POI una vez detectados, el tiempo inicial de detección excede con creces el presupuesto temporal.

Algoritmo	**Orb+Freak**	**Star+Freak**	**Brisk**	**Surf**	**Fast+Freak**
Cantidad POIs	500	503	349	267	1524
Cantidad de Correspondencias Correctas	64	39	40	255	29
% Correspondencias Correctas	12,80%	7,70%	11,50%	95,50%	1,90%
Detección POI	10,18 ms 57%	14,31 ms 47%	10,99 ms 35%	86,61 ms 51%	0,79 ms 0%
Descripción POI	2,77 ms 15%	1,58 ms 5%	0	35,91 ms 21%	4,16 ms 1%
Búsqueda Correspondencias	4,61 ms 26%	10,71 ms 35%	16,49 ms 53%	11,29 ms 7%	413,33 ms 98%
Homografía	0,39 ms 2%	3,59 ms 12%	3,53 ms 11%	35,76 ms 21%	5,1 ms 1%
Tiempo Total	17,95 ms	30,19 ms	31,01 ms	169,57 ms	423,38 ms

Tabla 2.1. Resultados completos de la evaluación de los pasos de RA por imágenes para varias combinaciones de algoritmos de detección y descripción de POI.

Finalmente, dados los tiempos de ejecución observados se establece que los pasos ampliamente dominantes resultan los de búsqueda y descripción de POI así como el de emparejamiento. Considerando que estos dos primeros pasos dependen principalmente de parámetros controlables, como la resolución de las imágenes con los que se los alimenta, una vez seleccionado uno apropiado, no influirán en la escalabilidad del sistema en cuanto a la cantidad de objetivos de aumentación. Por otro lado, el paso de emparejamiento o búsqueda de vecinos más cercanos depende directamente del número de objetivos de aumentación, es decir, de la cantidad de imágenes arbitrarias que el usuario de la arquitectura pretenda aumentar. Se presentará en próximas secciones los estudios pertinentes a la selección de los algoritmos óptimos para este paso crucial.

Capítulo 3

Proceso de RA basada en reconocimiento de rostros e inferencia biométrica

Comenzando con este tercer capítulo se encuentra en la sección primera la descripción del proceso que permite la funcionalidad de aumentar rostros sobre la base de su reconocimiento y la inferencia de valores biométricos. A continuación la sección 2 presenta los algoritmos utilizados en cada paso mientras que la sección 3 analiza de manera teórica y alto nivel la complejidad computacional de cada paso involucrado en el proceso.

3.1 Descripción del proceso

Para lograr la detección, reconocimiento y aumentación de rostros humanos se propone un proceso análogo al de la RA basado en imágenes arbitrarias. Se parte de una base de datos de los rostros que se pretende sean reconocidos y aumentados por el sistema. La herramienta computa y almacena un descriptor para cada individuo utilizando una red neuronal convolucional profunda entrenada específicamente para la tarea de reconocimiento facial.

Al igual que con la RA basada en imágenes, el tiempo de procesamiento de este proceso es reducido mediante la utilización de dos bucles (de detección y de seguimiento) independientes pero relacionados.

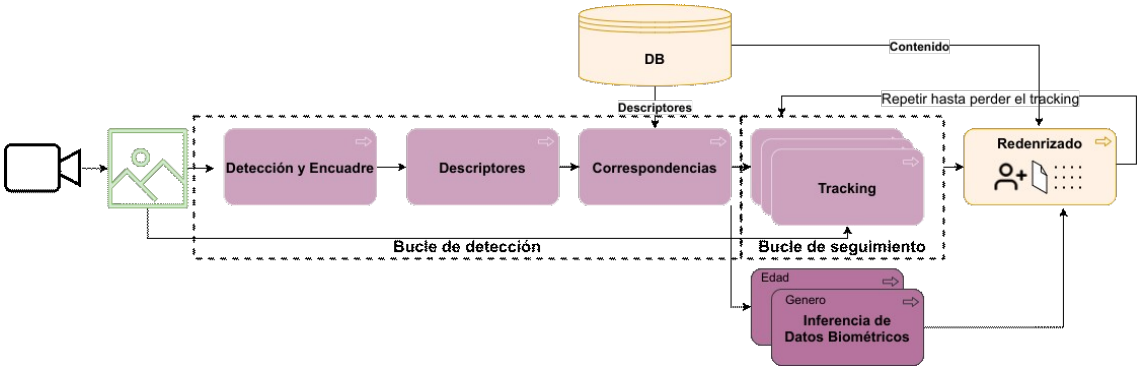

Figura 3.1. Diagrama conceptual del proceso para la aumentación basada en rostros.

Bucle de detección

1. Detección de rostros
   Se ejecuta el algoritmo de detección seleccionado para encontrar todas las instancias de rostros humanos en la imagen y calcular su posición o encuadre.
2. Cálculo de descriptores para cada rostro
   Se ejecuta el algoritmo seleccionado para computar un descriptor por cada rostro encontrado en el paso anterior.
3. Búsqueda de correspondencias
   Se buscan correspondencias entre descriptores de rostros objetivo y las del cuadro de vídeo actualmente siendo procesado.
4. Inferencia de datos biométricos. Se dispara la ejecución asíncrona en hilos de ejecución independientes de los distintos algoritmos de inferencia de datos a partir de imágenes.

Bucle de seguimiento

1. Tracking
   Con la información de encuadre del bucle de detección se inicializa en un hilo de ejecución independiente un algoritmo de seguimiento que actualiza la posición de cada rostro sin necesidad de volverlo a buscar.

Como se observa en la figura 3.1 el efecto de aumentación se logra utilizando otro programa de software el cual captura un flujo de vídeo monocular. En el primer bucle se alimenta un cuadro del flujo de vídeo a un algoritmo de detección de rostros. El mismo se encarga de detectar la presencia de rostros humanos dentro del cuadro y precisar su posición y encuadre.

Como segundo paso, se recortan regiones de interés (ROI) para cada rostro detectado, las cuales serán alimentadas a la red neuronal que producirá un descriptor para cada una.

Como tercer paso dichos descriptores serán luego cotejados con los almacenados previamente para buscar coincidencias, teniendo en cuenta que no serán idénticos debido no sólo a distorsiones en la imagen sino a posibles alteraciones en el ángulo de captura y expresiones de los rostros. En particular, la red neuronal utilizada para la implementación prototipo intenta mantener una distancia euclidiana máxima de 0,6 unidades para rostros de un mismo individuo.

El bucle de seguimiento, mediante la utilización de algoritmos de seguimiento de objetos genéricos como los implementados en la biblioteca OpenCV, es capaz de mantener y actualizar la posición de los rostros, objetivos de aumentación ya detectados por una fracción del costo computacional como se observará en la sección 5.3.

3.2 Detección y descripción de rostros

Como se introduce en el capítulo primero, la detección de rostros en tiempo real en un flujo de vídeo resulta posible y aplicable desde hace casi dos décadas. Desde entonces, las técnicas utilizadas han evolucionado pero manteniendo un fuerte foco en la utilización del aprendizaje de máquina. Aunque se pretende mantener la flexibilidad en la elección del algoritmo de detección de rostros mediante la incorporación de una interfaz abstracta, en particular se considera para el desarrollo del prototipo demostrador el detector por red neuronal provisto por OpenCV. El mismo, basado en el framework Caffe por (Jia et al. 2014) otorga un nivel de robustez elevado mientras que su tiempo de ejecución se mantiene entre los límites aceptables establecidos en 40 ms por pruebas anteriores.

Según lo investigado en cuanto a las técnicas y algoritmos para el reconocimiento o identificación de rostros, ninguna alternativa que pueda trabajar eficazmente partiendo de una sola instancia o imagen por individuo y que sea de uso general, es decir que no deba ser reentrenada o ajustada para un set de rostros específicos, logra una velocidad de procesamiento adecuada para la ejecución en tiempo real. Con esta limitación en mente, se consideran especialmente apropiadas las redes neuronales convolucionales entrenadas con el término de error de tripletas introducido por (Schroff, Kalenichenko, y Philbin 2015) Esta técnica de entrenamiento plantea describir el rostro de un individuo mediante un vector multidimensional continuo donde se busca minimizar la distancia intra-clase (mismo individuo) mientras maximiza la distancia inter-clase (distintos individuos). Dentro de esta familia de redes, los descriptores resultado de procesar distintas imágenes del rostro de un mismo individuo sólo pueden estar a una distancia euclidiana máxima menor a 1.2 unidades, es decir el vector resultado se distribuye en una hiperesfera de radio 0.6 unidades. Adicionalmente los vectores que generan constan de 128 números flotantes revistiendo una similitud de tamaño y variabilidad con los descriptores para imágenes SIFT. Si bien la evaluación de distintos modelos entrenados

para reconocimiento facial en términos de desempeño relativo escapa al alcance de esta tesis, se considera especialmente el modelo provisto por la biblioteca dlib («dlib» 2020).

3.3 Inferencia biométrica

Por su parte, la inferencia de parámetros biométricos a partir de imágenes es un campo con numerosas especializaciones y avances continuos. A su vez, buena parte de las técnicas disponibles incurren en tiempos de ejecución no compatibles con la restricción de tiempo real. Dados estos motivos, se propone la incorporación de una interfaz abstracta que otorgue flexibilidad para agregar o modificar algoritmos de inferencia a la vez que permita la ejecución asíncrona de ser necesario.

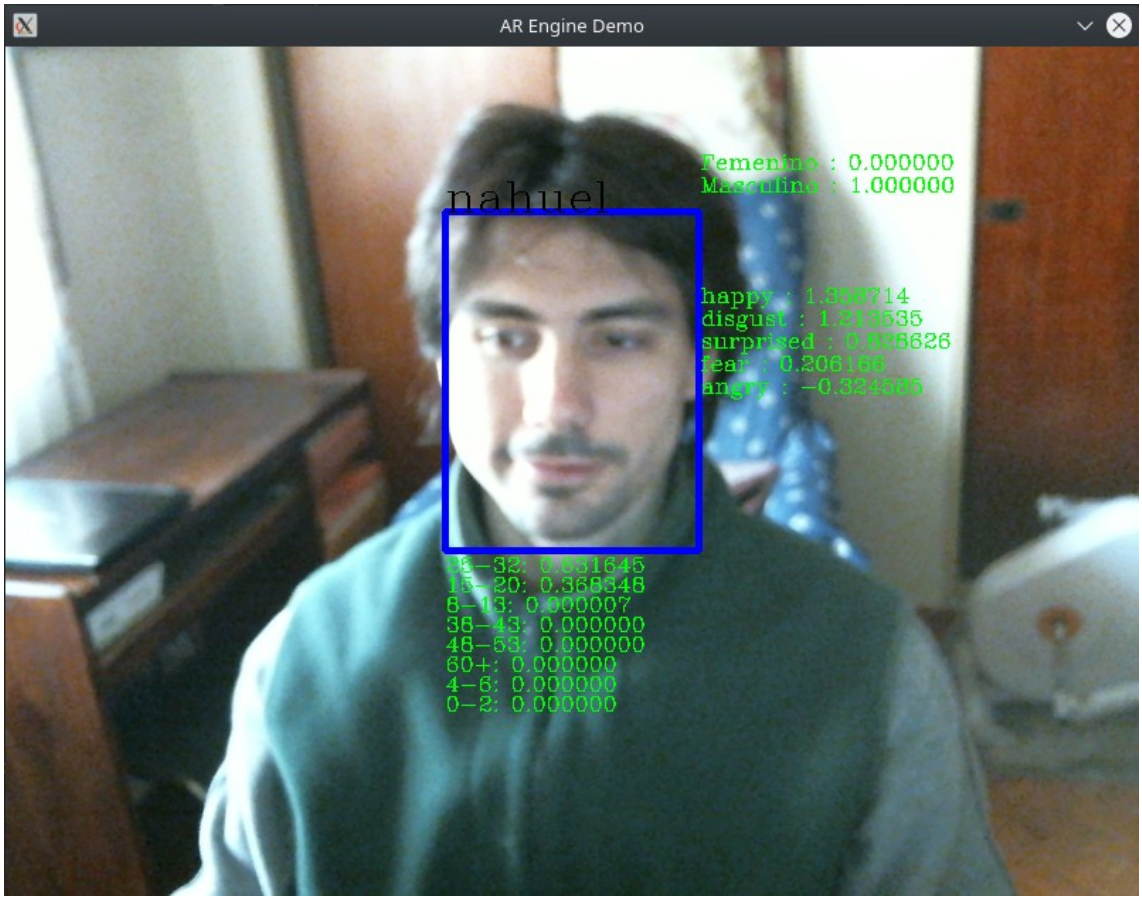

Figura 3.2. Ejemplo de detección, reconocimiento e inferencia biométrica a partir del rostro. Se incluyen los valores relativos para distintas categorías de los algoritmos de inferencia.

En la figura 3.2 se observa el proceso de detección, reconocimiento e inferencia biométrica en funcionamiento. Aunque tanto el desarrollo como la evaluación de

algoritmos de inferencia biométrica escapa al alcance de la presente tesis, se integran y utilizan los siguientes tres algoritmos:

- Inferencia de género (parte superior izquierda de la figura): red neuronal convolucional profunda basada en la arquitectura Inception (Szegedy et al. 2014) entrenada para (Lapuschkin et al. 2017).
- Inferencia de edad (parte inferior de la figura): red neuronal convolucional profunda también basada en la arquitectura Inception (Szegedy et al. 2014) entrenada para (Lapuschkin et al. 2017)
- Inferencia de expresión facial (lateral derecho de la figura): red neuronal convolucional profunda basada en la arquitectura RESNet (He et al. 2015) y disponible en («Facial expression Recognition ResNet» 2020)

3.4 Complejidad computacional

Para el procesamiento de rostros, la complejidad de los algoritmos se describe a continuación:

1. Algoritmo de detección de rostros
    - Depende principalmente de la resolución en píxeles WxH del cuadro del flujo de vídeo pero varía fuertemente según el algoritmo utilizado. De igual manera, esta resolución está acotada por encima por la resolución de la cámara utilizada.
2. Red neuronal que computa un descriptor para cada rostro
    - Depende de la resolución de la ROI utilizada como entrada pero es fija para cada arquitectura de red particular.
    - Depende de la cantidad de rostros R que hayan sido detectados en el paso de detección.
3. Algoritmo de cotejamiento
    - Depende del tamaño T del descriptor computado el cual es fijo para cada arquitectura de red.
    - Depende de la cantidad de individuos N de la base de datos de rostros a ser reconocidos por el sistema.
4. Algoritmos de inferencia de informaciones
    - En general, dependen de la resolución de las ROI a utilizar.

En este caso, la complejidad computacional resulta difícil de determinar analíticamente dada la naturaleza de los algoritmos de aprendizaje de máquina que se proponen utilizar. En particular, los niveles de performance informados por sus respectivos autores indican que ninguno de los modelos de reconocimiento e inferencia considerados permitirían una ejecución en tiempo real para el bucle de detección.

Dado que la cantidad de rostros humanos que se pueden encontrar en un determinado cuadro de un flujo de vídeo es inherentemente limitado, es posible adaptar la implementación para lidiar inteligentemente con un número razonable de casos máximos simultáneos sin afectar la escalabilidad.

Con este proceso también se hace uso de algoritmos de seguimiento (tracking) que evitan el reprocesamiento del proceso completo en cada cuadro del flujo de vídeo, reduciendo su tiempo de procesamiento.

Capítulo 4

Propuesta de arquitectura integrada de RA basada en imágenes y rostros

En este capítulo se presenta la integración de los procesos propuestos con anterioridad en una arquitectura escalable única. La sección 1 introduce la arquitectura y explicita su estructura general mientras que la sección 2 discute la organización del flujo de ejecución de acuerdo a las restricciones temporales y consideraciones especiales. A continuación, la sección 3 adentra en la implementación de un prototipo demostrador mientras que la sección 4 detalla decisiones de diseño en torno a la aplicación de paralelismo y asincronía. Finalmente, la sección 5 discute una propuesta complementaria a esta integración para permitir que herramientas de autor manejen el incremento de escala en términos de cantidad de objetivos de aumentación.

4.1 Integración de los procesos

Al utilizar un proceso para aumentación de rostros como el propuesto, queda en inmediata evidencia la similitud con el proceso de aumentación de imágenes presentado. Dada lo análogo de los procesos, los mismos pueden ser combinados en una arquitectura integrada, unidos principalmente por el paso de cotejamiento como fue publicado en (N. A. Mangiarua, Ierache, y Abasolo 2020). Esta arquitectura integrada permite la aumentación escalable en simultáneo de imágenes y de rostros mientras se realiza inferencia de datos biométricos como se observa en la figura 4.1. Naturalmente, no existe razón para mezclar descriptores extraídos de imágenes con los computados a partir de rostros, pero ambos tipos pueden ser cotejados por el mismo algoritmo y estructuras de datos, el cual resulta de vital importancia para lograr la escalabilidad del sistema.

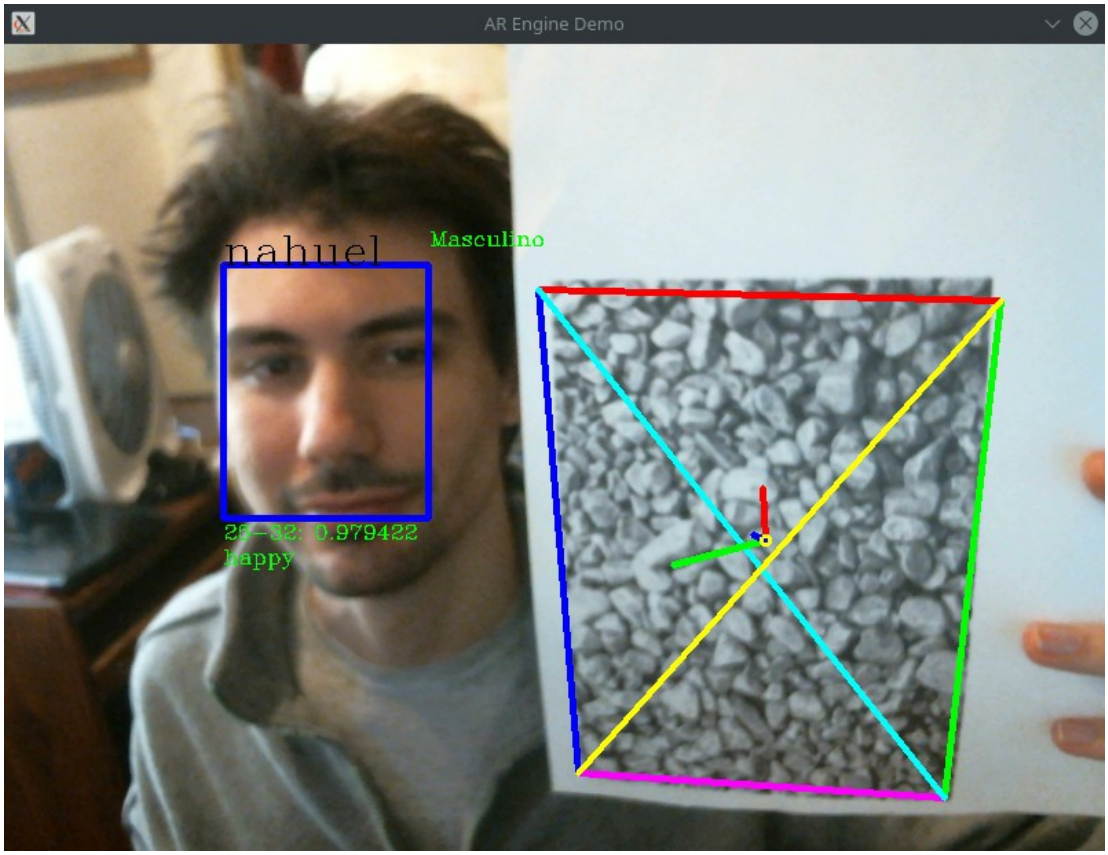

Figura 4.1 Captura de la implementación prototipo de la arquitectura integradora en funcionamiento. Se puede apreciar la combinación de aumentación de imágenes a la derecha y aumentación de rostros con reconocimiento (nombre sobre el rostro) e inferencia biométrica de género (arriba a la derecha), probabilidad de rango de edad (abajo del rostro) y expresión facial más probable (debajo del rango de edad).

El diagrama conceptual de la arquitectura completa considerando las interfaces y datos de entrada puede apreciarse en la figura 4.2. Aquí se observan los principales pasos propios de la arquitectura (en verde y en morado) y los pasos o artefactos externos conectados (en beige).

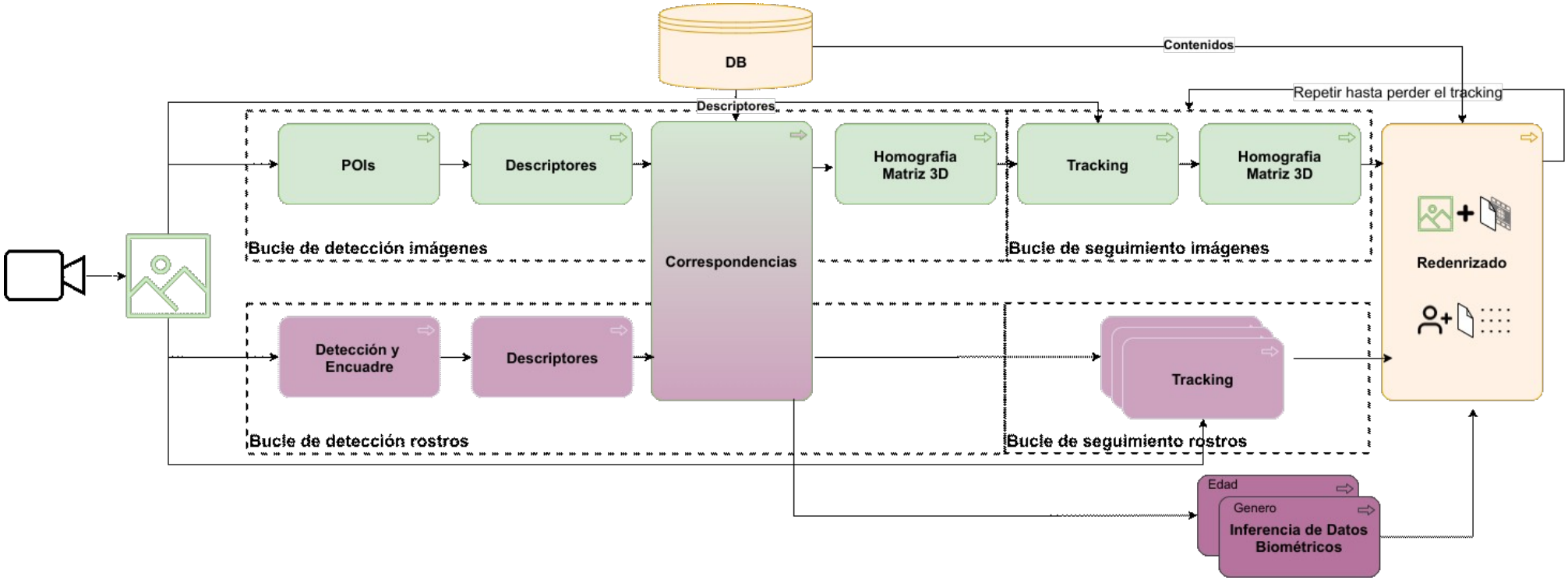

Figura 4.2. Diagrama conceptual de la arquitectura integrando los procesos de RA basada en imágenes RA basada en rostros. Este diagrama no pretende especificar aun órdenes de ejecución ni paralelismo.

4.2 Organización del flujo de ejecución

Dada esta arquitectura, resulta tentador el intentar correr ambos de los ahora subprocesos de forma paralela con cada cuadro del flujo de vídeo, haciendo uso de múltiples núcleos de CPU. Sin embargo, al tener en cuenta las características del funcionamiento del hardware al momento de ejecutar estos algoritmos se puede deducir que no sería apropiado. Por un lado, un gran número de los algoritmos son computacionalmente intensivos en cuanto al uso de ancho de banda de memoria, llegando a saturarlo. Ejecutar más de uno de estos algoritmos a la vez sólo incurriría en una contención por el recurso limitado, incrementando la cantidad de fallos de caché y reduciendo el desempeño de ambos. Por el otro, varios algoritmos ya poseen la capacidad de ejecutar en paralelo.

Además, ejecutar cada subproceso de manera secuencial uno detrás del otro para cada cuadro del flujo de vídeo, restringiría aún más el limitado tiempo del que se dispone para completar cada paso. Un decremento del presupuesto temporal para cada paso de los subprocesos incurriría en una indeseada pérdida de calidad y escala.

Se propone entonces la ejecución entrelazada de los bucles de cada subproceso, aprovechando la observación de que, con un flujo de vídeo de al menos 24 cuadros por segundo, las diferencias entre lo visualizado en un cuadro y el siguiente serán mínimas. Incluso en los casos donde se produzca un eventual movimiento rápido de la cámara se dará principalmente alguno de los dos siguientes casos. El primero es que haya objetos desplazándose a la misma velocidad y dirección (que el usuario intenta seguir) y por lo tanto no revistan un gran cambio en la imagen. El segundo es que, si cambiara la imagen por completo, la limitación de la percepción del cerebro humano de trece cuadros por segundo producirá el efecto conocido como desenfoque por movimiento (motion blur) impidiendo que el usuario perciba el deterioro de la calidad de la aumentación durante ese periodo de movimiento.

Dados dos bucles por subproceso, detección de imágenes, seguimiento de imágenes, detección de rostros y seguimiento de rostros, sólo uno de los mismos ejecutará para cada cuadro del flujo de vídeo, intercalando los relacionados con imágenes y los relacionados con rostros. El flujo de ejecución de alto nivel de estos pasos, ilustrado en la figura 4.3, destaca los principales puntos de decisión del proceso integrado. Adicionalmente se destaca la delegación de los procesos de inferencia e identificación a hilos de ejecución paralelos.

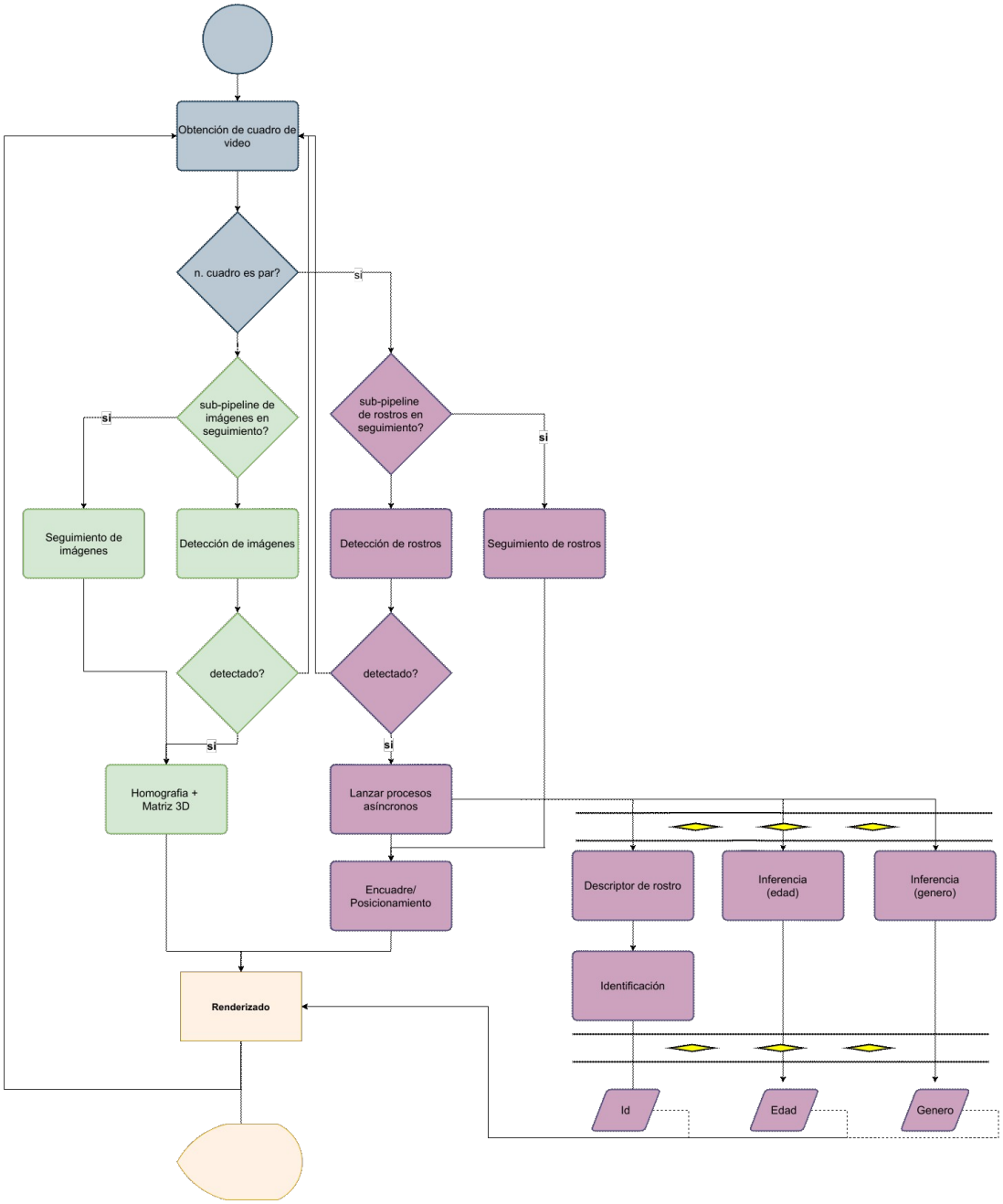

Figura 4.3. Diagrama de flujo simplificado de la arquitectura integrada. En verde se observa el subproceso basado en imágenes, en morado se observa el subproceso basado en rostros, mientras que los demás nodos pertenecen a partes comunes o ajenas a la arquitectura propuesta.

Cabe destacar la incorporación de esquemas de ejecución asíncronos que aprovechen las capacidades de hardware moderno. Esto no sólo otorga flexibilidad para la selección de algoritmos de inferencia sino también la posibilidad de generar un marco de extensibilidad que permita a consumidores de la arquitectura incluir inferencias adicionales de forma transparente a la arquitectura a través de una interfaz común. Los detalles de esta distribución en hilos de CPU será tratada con más detalle en secciones subsiguientes.

4.3 Implementación prototipo

Como artefacto central producto de este trabajo de tesis se implementa la arquitectura propuesta en lenguaje C++ por su eficiencia de ejecución. En el Apéndice A se detalla la ubicación del repositorio de código así como la documentación generada al respecto.

Se presenta el diagrama de colaboración con las principales clases de código en la figura 4.4. En el mismo se observa la distribución de tareas en sus clases pertinentes así como la relación que existe entre ellas a fines de facilitarle al lector la comprensión del código involucrado en experimentos y evaluaciones posteriores.

Cabe destacar algunas decisiones de implementación particulares que fueron aplicadas en este prototipo. En primer lugar, se implementa el subproceso de imágenes y el subproceso de rostros en forma de sub-engines con capacidad de funcionamiento autónoma si así fuera requerido. Esto permite probar cada parte de forma independiente así como aplicar modificaciones o mejoras en una sin afectar en forma directa la otra. Por otro lado, se abstraen mediante interfaces la mayoría de los algoritmos de cada paso de los subprocesos como por ejemplo FaceTracker y FaceDetector. Esto no sólo permite la realización de pruebas sino que facilita la actualización de los algoritmos.

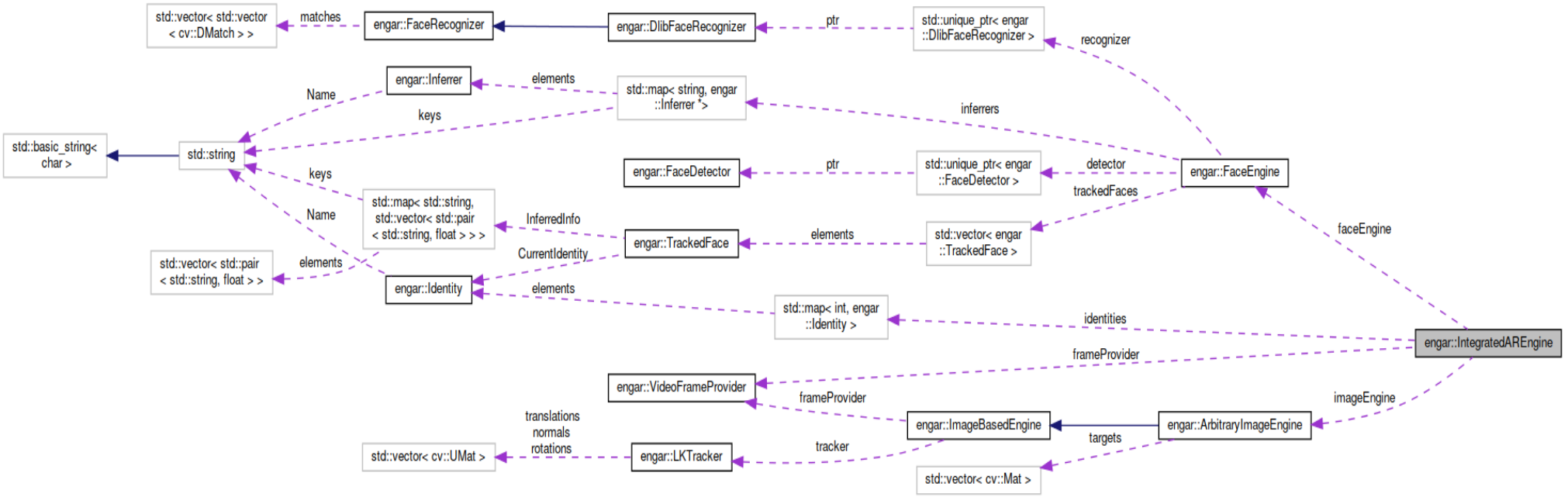

Figura 4.4. Diagrama de colaboración de la clase IntegratedArEngine que condensa el núcleo del prototipo de la arquitectura implementado en C++.

4.4 Distribución de la ejecución sobre hilos de CPU

Aunque los pasos del subproceso basado en imágenes logran cumplir con la restricción de tiempo de ejecución como se discute en capítulos anteriores, si se quisiera dar soporte al seguimiento de múltiples imágenes objetivo en simultáneo, resulta recomendable una configuración donde cada una de estas imágenes es seguida utilizando un hilo de CPU ejecutando en paralelo como se observa en la figura 4.5.

Además, existen partes del subproceso de rostros que por sí mismas exceden el presupuesto temporal, como los algoritmos de descripción de rostro y los de inferencia de datos biométricos. Como ya fue mencionado, esto imposibilita la ejecución secuencial de todo un subproceso como parte del hilo de CPU principal.

La solución propuesta para solventar esta dificultad se observa en la figura 4.6 y consiste en sólo detectar los rostros y encuadrarlos para ofrecer un feedback visual al usuario de forma secuencial en el hilo de ejecución principal, y luego delegar la ejecución de estos algoritmos más demandantes a hilos de CPU independientes. Estos hilos se procesarán mayormente en simultáneo con los bucles de seguimiento (tracking), los cuales tienen una baja demanda de procesamiento en sí mismos, evitando la saturación del CPU y el ancho de banda de la memoria. A medida que los algoritmos completan su ejecución, sus resultados se agregan a la aumentación.

También existe la incertidumbre sobre la cantidad de rostros que podrá estar presente en una misma imagen. Lo recomendable, es limitar la cantidad máxima de rostros a seguir según las capacidades de hardware donde corra la arquitectura.

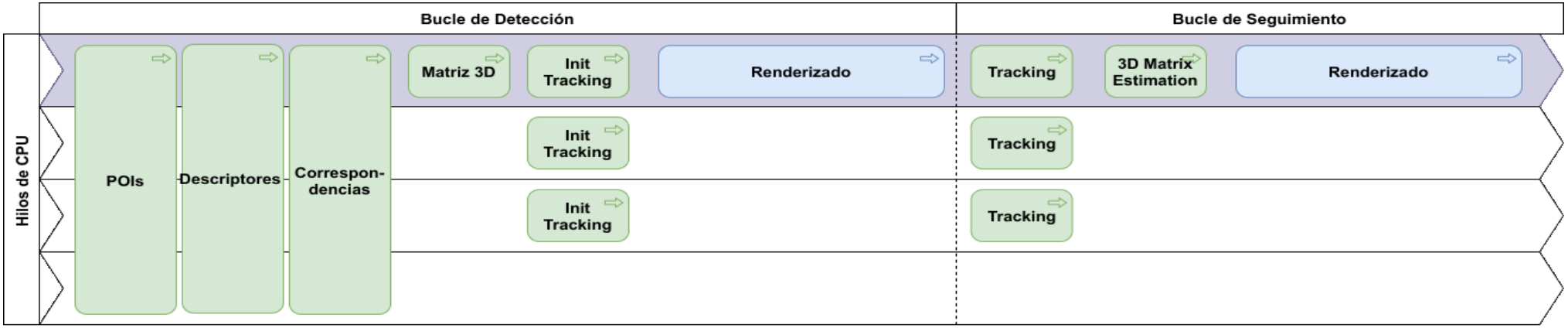

Figura 4.5. Distribución de la ejecución de algoritmos del subproceso de *imágenes* sobre distintos hilos de CPU. El andarivel resaltado representa el hilo de ejecución principal de la aplicación.

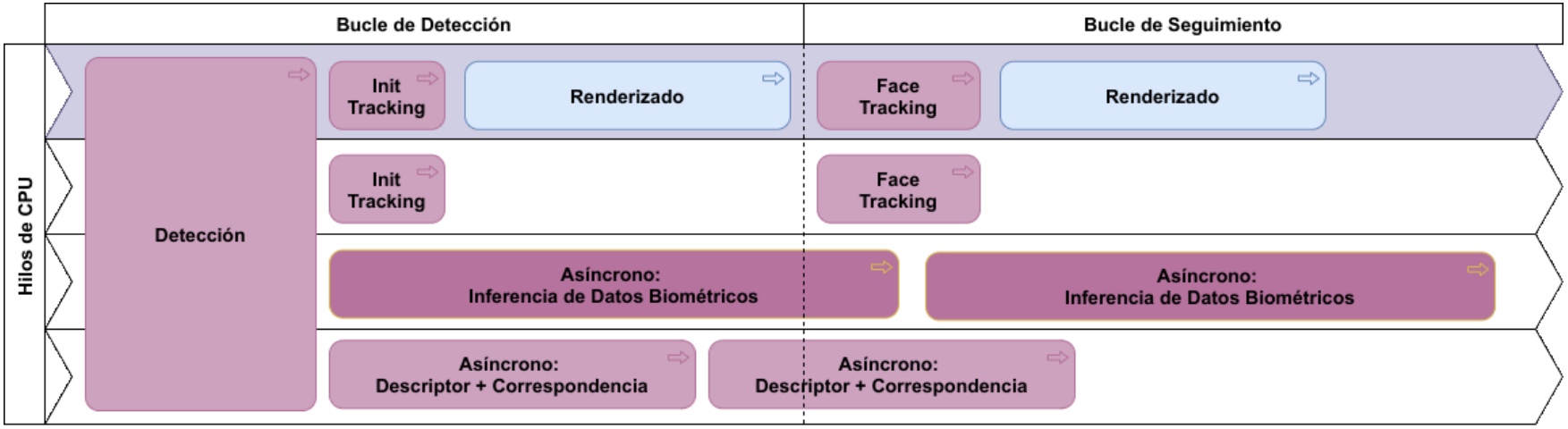

Figura 4.6. Distribución de la ejecución de algoritmos del subproceso de rostros sobre distintos hilos de CPU. El andarivel resaltado representa el hilo de ejecución principal de la aplicación.

Capítulo 5

Evaluación de algoritmos de búsqueda aproximada de vecinos más cercanos en el contexto de la RA

En este capítulo se estudian una serie de aspectos de los algoritmos de búsqueda aproximada de vecinos más cercanos - en inglés *Approximate Nearest Neighbour (ANN)*- relevantes para su utilización en el contexto de la RA. Primeramente se desarrolla el contexto y la necesidad de la utilización de algoritmos ANN avanzados en la sección 1. En las secciones 2 y 3, se discute el esquema de evaluación tradicional para esta familia de algoritmos y se propone un nuevo esquema específico para este contexto particular. A continuación, las secciones 4 a 7 presentan evaluaciones cuantitativas sobre un número de algoritmos de esta familia en el marco del esquema propuesto. En particular se estudian: el efecto de la presencia de verdaderas correspondencias, el nivel de variación entre descriptores, el efecto de esta variación en los descriptores para los dos primeros vecinos más cercanos y el desempeño con datos con bajo nivel de distorsión. Finalmente se presenta una validación del desempeño general y su evolución según el tamaño del set de datos en las secciones 8 y 9.

5.1 Búsqueda aproximada de vecinos más cercanos

Observando que los descriptores computados tanto para los rostros humanos como para los POI sobre imágenes no son perfectamente invariantes, resulta inapropiado utilizar para este paso algoritmos de búsqueda de correspondencias exactas, como una clásica tabla de hashing. En cambio, resulta indispensable la utilización de algoritmos de búsqueda aproximada de vecinos más cercanos o ANN.

Los algoritmos de ANN son un área de investigación activa con nuevas aproximaciones y algoritmos siendo desarrollados todos los años. Pese a esto, los estudios comparativos extensivos son escasos, ya que cada autor realiza una comparación con sólo un subgrupo manejable de algoritmos al momento de publicar sus aportes. En particular, el

único estudio extensivo encontrado es (Aumüller, Bernhardsson, y Faithfull 2018); ampliamente referido por otros autores y desarrolladores.

Dado que la complejidad computacional sobre los algoritmos que se encargaran del cómputo de las correspondencias tiene una relación directa con la escala, es decir la cantidad de objetivos de aumentación con los que pueda trabajar el sistema, se vuelve necesario realizar un estudio más profundo sobre sus características y comportamientos con el fin de seleccionar el más apropiado para la arquitectura. En particular se considera a los siguientes algoritmos: ANNG (M. Iwasaki 2010), ONNG (Masajiro Iwasaki y Miyazaki 2018), FLANN (Muja y Lowe 2009), HNSW (Malkov y Yashunin 2016) implementado en (Boytsov y Naidan 2013), ANNOY («Annoy» 2020) and KGRAPH («KGraph» 2020). Los mismos son seleccionados por la disponibilidad de una implementación en lenguaje C++ y su buen desempeño de acuerdo a los estudios citados. En todos los casos los algoritmos fueron envueltos con la interfaz de abstracción DescriptorMatcher de OpenCV con el fin de homogeneizar su utilización.

5.2 Esquema de evaluación tradicional

Resulta usual en la literatura del tema, realizar la comparación de algoritmos de ANN utilizando como métrica principal la cantidad de verdaderos aciertos (recall) para al menos los diez primeros vecinos más cercanos a un elemento de búsqueda (query). Como se observa en análisis previos, los pasos de cómputo de homografía y estimaciones de matrices no hacen un aporte dominante al tiempo de ejecución siempre y cuando se mantenga controlado el número de correspondencias con las que se trabaja. Es por esto que resulta conveniente trabajar con sólo los dos primeros vecinos más cercanos para cada descriptor obtenido durante la ejecución de los primeros pasos del subproceso. Esta limitación permite mantener controlados los tiempos de ejecución de pasos subsiguientes así como obtener un nivel de robustez basal bien mediante el test de distancia propuesto por Lowe en [59] o bien alimentando ambas correspondencias de cada punto al algoritmo de homografía y permitir que RANSAC determine la opción correcta para cada uno.

Pero si se limita a los algoritmos de ANN a sólo los dos primeros vecinos más cercanos, surge la pregunta de cuán significativos resultan los resultados comparativos previamente publicados cuando se los somete a esta limitación. Adicionalmente, tanto (Muja y Lowe 2014) como (Mikolajczyk y Matas 2007), entre otros, notan que el desempeño de muchos algoritmos de ANN cambia de acuerdo a la presencia o no de lo que denominan una verdadera correspondencia, es decir, una coincidencia exacta o casi exacta entre el elemento de consulta (usualmente llamado por su forma del inglés *query*) y alguno de los elementos contenidos en el set de datos sobre los que se busca.

Habiendo identificado al paso de búsqueda de correspondencias como el cuello de botella principal a la hora de escalar a un mayor número de objetivos de aumentación (imágenes y rostros cargados por el usuario) y dado que los descriptores utilizados en la arquitectura tienen un alto nivel de invarianza y robustez al cambio, se vuelve de interés estudiar si esta particularidad afecta o no de manera significativa el desempeño de los algoritmos de ANN.

5.3 Propuesta de evaluación específica para el contexto de la RA

Se propone un esquema de evaluación para algoritmos de búsqueda ANN específico para el contexto de la RA. El mismo implica la evaluación de esta familia de algoritmos utilizando datos donde la variación entre los elementos query y los elementos de entrenamiento, base u originales sea reducida. A partir de estos sets de datos, se propone que los algoritmos de búsqueda de ANN sean comparados por su recall en los dos primeros vecinos más cercanos y teniendo en cuenta como estos valores evolucionan de acuerdo a la cantidad de datos utilizados.

De acuerdo al esquema propuesto, a continuación se describe una serie de experimentos y sus resultados. Comenzando por la medición del desempeño según la presencia de verdaderas correspondencias se buscará determinar la magnitud del efecto descrito dadas las condiciones particulares de la RA. Luego, se determina el nivel de variación entre descriptores correctamente emparejados y el efecto de distintos niveles de variación sobre el desempeño de los algoritmos en general y específicamente para bajos niveles de variación. Finalmente se medirán las fluctuaciones en el desempeño de los algoritmos de acuerdo a la cantidad de datos con el fin de probar su estabilidad ante el incremento de la escala.

Estos resultados son obtenidos al ejecutar código en un sólo hilo de CPU en un equipo corriendo Ubuntu Linux 19.10, equipado con un procesador AMD Ryzen 5 1600X (12Mb L3 cache) y 16Gb de RAM corriendo a 2667 Mhz con un ancho de banda aproximado de 20Gbps.

5.4 Conjuntos o sets de datos y de consultas

Los algoritmos avanzados de búsqueda ANN obtienen sus características y mejoras a través de la construcción de una estructura de datos usualmente llamada índice. Dicho índice debe ser construido en base a un conjunto de datos particulares y sólo será válido en relación a este, por lo cual es usual denominar al proceso como entrenamiento. En

los subsiguientes experimentos se utilizan una serie de términos y conjuntos de datos que se presentan a continuación.

Un conjunto o set denominado de datos o de entrenamiento constituye una cantidad determinada de vectores sobre los cuales se realizará una búsqueda de los valores (vecinos) más cercanos a un vector externo llamado query, según distintas métricas de distancia. Es sobre estos conjuntos que los distintos algoritmos de búsqueda ANN se entrenan y construyen sus índices. Como convención de nombres se utilizará letra mayúscula para denominarlo.

Por otro lado un conjunto o set denominado de consultas (queries) constituye una cantidad determinada de vectores que serán tomados de uno en uno como elemento query en la búsqueda de sus vecinos más cercanos (vecinos que serán elementos de un set de datos o de entrenamiento). Salvo cuando se indique lo contrario, cada set de consultas se encuentra asociado con un set de datos particular. Como convención de nombres se utiliza letra minúsculas para denominarlos.

En los experimentos se manejan dos tipos de conjuntos de consultas denominados disjuntos y distorsionados. En el primer tipo, el set de consultas y el set de datos pueden ser conjuntos disjuntos, en cuyo caso ninguno de los elementos del set de pruebas se encuentra en el set de datos En el segundo tipo, los elementos del set de consultas se forman a partir de aplicar un cierto grado de distorsión a elementos presentes en el set de datos asociado. Como convención de nombres en los set de consultas distorsionados se utiliza el sufijo “-d” seguido de un número que indica el porcentaje de distorsión aplicado.

La distorsión sobre los elementos se realiza utilizando el código incluido en el apéndice B según la métrica de distancia. Para distancia euclidiana, la distorsión es aplicada al sumar o restar un pequeño número aleatorio de acuerdo al nivel de distorsión deseado, a cada dimensión del elemento. Para distancia Hamming, la distorsión se aplica invirtiendo al azar un número de bits.

Gran parte de los diversos conjuntos que se derivan o construyen para cada experimento utilizan como base a SIFT1M introducido en (Jégou, Douze, y Schmid 2011) y a SIFT10M introducido en (Dua y Graff 2017). Ambos consisten en una lista de 1 millón y 10 millones de vectores SIFT respectivamente. Cada vector consta de 128 dimensiones cuyos valores se encuentran discretizados entre 0 y 255. Adicionalmente el set SIFT1M incluye 10.000 vectores extra (no presentes en el millón mencionado) considerados en su conjunto como set de consultas disjunto provisto.

Los emparejamientos correctos - en inglés *ground truth* - entre vectores query y sus vecinos más cercanos son re computados en cada caso utilizando el algoritmo de fuerza bruta que es ciento por ciento preciso.

5.5 Desempeño según la presencia de una verdadera correspondencia, distintos niveles de distorsión y diferente cantidad de vecinos buscados

Partiendo de las observaciones de Muja (Muja y Lowe 2014) y Mikolajczyk (Mikolajczyk y Matas 2007) entre otros, se propone corroborar con una experiencia reducida el efecto de la existencia o no de una verdadera correspondencia así como el efecto que producen distintas proporciones de cambio entre descriptores. A la vez, se propone evaluar si la cantidad de vecinos más cercanos para un mismo elemento query requerido a los algoritmos de ANN afectan su desempeño. Es decir, si cada orden de cantidad de vecinos más cercanos reviste un mismo nivel de recall.

5.5.1 Diseño del experimento

Se preparan una serie de tres set de datos o entrenamiento y seis sets de queries o consultas como se observa en la tabla 5.1.

Entrenamiento 200.000 elementos	**Pruebas Muestreo 10.000 elementos**			
	Disjunto	**Distorsionados**		
		0%	10%	20%
SIFT1M	sift1m	sift1m-d0	sift1m-d10	sift1m-d20
SIFT10Ms1	sift10ms1			
SIFT10Ms2	sift10ms2			

Tabla 5.1. Resumen de los sets de entrenamiento y sus sets de consultas correspondientes.

El primer set de datos consiste en los primeros 200.000 vectores de SIFT1M mientras que el segundo y el tercero son dos conjuntos disjuntos que consisten en 200.000 vectores tomados secuencialmente de SIFT10M.

Asociados al primer set de entrenamiento se definen cuatro set de consultas de 10.000 elementos cada uno. Uno de ellos (sift1m) es el provisto con SIFT1M en un archivo separado. Los otros tres sets de consultas se generan tomando los primeros 10.000 elementos de SIFT1M y aplicándoles respectivamente 0%, 10% y 20% de distorsión.

Asociados al segundo y al tercer set de entrenamiento se construyen sets de consulta disjuntos tomando muestras únicas al azar de SIFT10M no incluidos en ningún otro set.

Se selecciona para esta prueba los algoritmos HNSW y FLANN, y se mide el recall de cada orden de vecino más cercano para cada juego de consultas con su set de entrenamiento correspondiente para distancia Euclidiana.

5.5.2 Resultados obtenidos

Como se observa en las figuras 5.1 y 5.2, el nivel de recall de un algoritmo no resulta constante para distintos órdenes de vecinos. Es decir, dado un elemento query, la probabilidad de que un determinado algoritmo ANN como HNSW o FLANN devuelvan el correcto primer vecino más cercano no es la misma que la probabilidad de que devuelvan el segundo vecino más cercano a dicho elemento query.

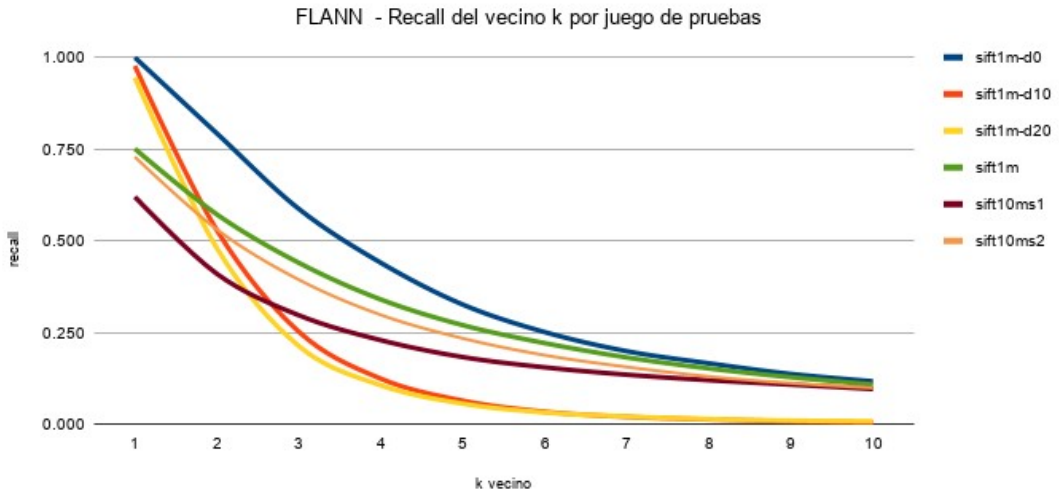

Figura 5.1. Curvas de recall para cada orden de vecino k por set de consultas para algoritmo FLANN.

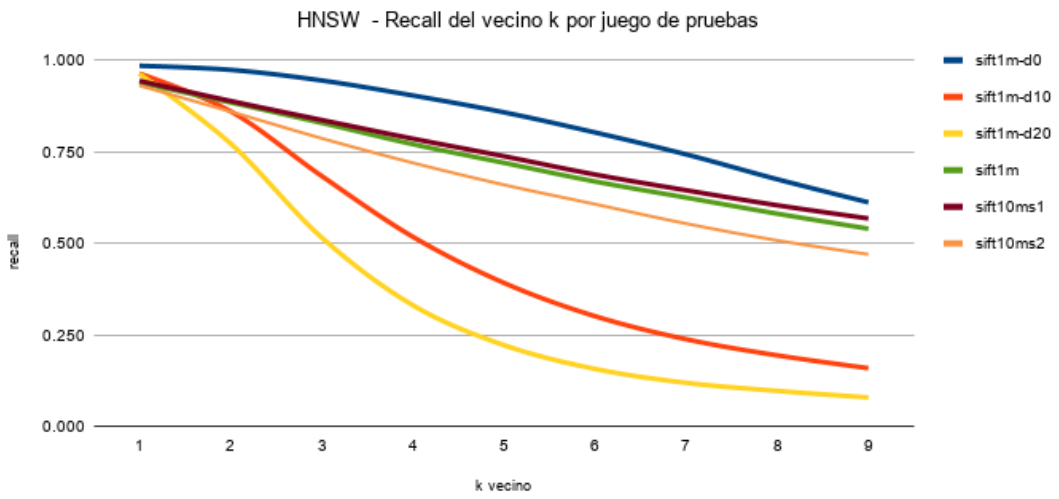

Figura 5.2. Curvas de recall para cada orden de vecino k por set de consultas para algoritmo HNSW.

De igual forma, se observa que la presencia de una verdadera coincidencia para cada elemento query como en el set de consultas con 0% de distorsión, altera significativamente la curva de recall por orden de vecino.

También resulta de relevancia la observación de que la mayor cantidad de variabilidad se produce entre el primer y cuarto vecino más cercano, zona clave en el contexto de la RA donde suele trabajarse con no mucho más de dos vecinos más cercanos.

5.6 Nivel de variación entre descriptores correctamente emparejados

5.6.1 Diseño del experimento

Para determinar el efecto de la similitud entre el elemento query y alguno de los elementos del set de datos en el contexto de la RA, primero se debe determinar el nivel de similitud observado entre descriptores computados sobre los objetivos de aumentación y aquellos correctamente identificados sobre los cuadros de vídeo procesados. Se propone simular el uso de una aplicación de RA basada en imágenes bajo condiciones de explotación semi-controladas, asumiendo una buena iluminación uniforme y mínima oclusión de la imagen objetivo.

Para ello se dispone la ejecución del código correspondiente al bucle de detección del subproceso de imágenes con un algoritmo representativo de los de tipo continuo y otro representativo de los de tipo binario. Se seleccionan los algoritmos SIFT y ORB correspondientemente y se genera un flujo de vídeo simulado a partir de tres elementos. En primer lugar se utiliza una imagen con relativa alta complejidad en cuanto a los elementos y variaciones visuales que presenta, para ser utilizada como fondo. Esta imagen aumentará la dificultad de la tarea al presentar posibles puntos de confusión para los algoritmos utilizados. El segundo elemento es una imagen con muy baja complejidad visual, por ejemplo, una imagen de papel blanco. Esta imagen hará las veces de marco, permitiendo un distanciamiento entre el fondo y los objetivos de aumentación. Este distanciamiento es comúnmente recomendada por los sistemas de RA ya que otorgan un margen de error para las estimaciones y facilitan al usuario el enfoque de la imagen objetivo de aumentación. El tercer elemento consiste en una serie de imágenes de prueba correspondientes a obras de arte gráfico (pinturas, dibujos, esbozos, etc). Utilizando el código incluido en el Apéndice C, cada imagen de un subconjunto del set de prueba se combina con el segundo elemento, aplicando una transformación afín de cambio de perspectiva sumado a una rotación al resultado de la fusión. Dicho resultado parcial es luego inserto sobre el primer elemento de fondo como se observa en la figura 5.3. Así mismo, esto se ilustra a modo de ejemplo en el vídeo disponible en el apéndice E. La rotación varía en intervalos de 10 grados por cuadro durante un lapso de 18 cuadros por imagen.

El set de imágenes de prueba consiste de trabajos publicados en («Wikiart» 2020) y compilados por («Kaggle» 2020). De dicha serie se extrae una muestra de 5000 elementos para los cuales el algoritmo SIFT, con sus parámetros por default, logre encontrar al menos 190 POI. Este filtrado elimina aquellas imágenes que, por sus

características propias, sea altamente probable que no puedan ser utilizadas como objetivos de aumentación de forma robusta. Sin embargo este filtro mínimo no garantiza que las imágenes que lo superan realmente representen buenos objetivos de aumentación.

Como paso previo a la ejecución del bucle de detección, se pre-calculan hasta 200 POI por cada una de las 5000 imágenes de prueba, incorporándolas a un algoritmo ANN por fuerza bruta, constituyendo el set de datos.

Figura 5.3. Ejemplo de cuadro del flujo de vídeo simulado. Se observa la imagen de prueba principalmente en el cuadrante superior izquierdo, enmarcada por el segundo elemento con textura de papel blanco.

El bucle se ejecutará una vez por cuadro para las primeras 200 imágenes del set de datos. Como resultado de la ejecución del código se medirán los siguientes valores:

- La cantidad de cuadros del flujo de vídeo simulado donde se detecta la imagen objetivo correcta.
- La cantidad de correspondencias correctas (juzgadas por la estimación del algoritmo homografía y pnp) entre descriptores query (aquellos obtenidos desde un cuadro del flujo de vídeo) y los descriptores del set de datos (aquellos pre-calculados)

- La distancia media entre el elemento query y su elemento correspondiente en el set de datos para cada correspondencia correcta.

5.6.2 Resultados obtenidos

Considerando el tamaño de los descriptores SIFT de 128 números enteros entre 0 y 255, la distancia euclidiana máxima entre dos elementos resulta de la ecuación: $\sqrt{256^2 * 128}$.

Por otro lado, de los descriptores ORB de 256 bits, se obtiene que la distancia Hamming máxima posible entre dos elementos es 256.

Como resultado del experimento se obtiene que la distancia promedio entre descriptores SIFT y sus correctas correspondencias con el descriptor del mismo POI en el set de datos es menor al 5% de la distancia máxima posible. Por otro lado la distancia para descriptores ORB es de menos del 15% de la distancia máxima.

Análogamente, los descriptores calculados para rostros en el subproceso correspondiente consideran, por definición de sus creadores, que correspondencias correctas y confiables sólo pueden estar a una distancia euclidiana máxima menor a 1.2 unidades (se distribuyen en una hiperesfera de 0.6 unidades de radio).

Esta similitud, que resulta esperable, proporciona un marco con el cual analizar el desempeño de los distintos algoritmos de ANN para el contexto particular de la RA.

5.7 Efecto de la variación de los descriptores para los dos primeros vecinos más cercanos

Como se menciona previamente, en el contexto de la RA basada en imágenes y rostros la variación entre los elementos query y los elementos en el set de datos es reducida. A la vez, la cantidad de vecinos más cercanos que se exige a los algoritmos es mucho más limitada que los valores usualmente utilizados en el común de las comparaciones de la literatura. Se propone entonces la siguiente experiencia para analizar comparativamente los efectos de las particularidades de este contexto sobre un grupo ampliado de algoritmos de ANN. Los mismos fueron seleccionados principalmente por la disponibilidad de una implementación estable en lenguaje C++ el cual garantiza un mejor desempeño al de implementaciones en lenguajes de nivel superior como podrían ser Java o Python.

5.7.1 Diseño del experimento

La tabla 5.2 presenta los tres conjuntos de datos de entrenamiento y doce conjuntos de consulta utilizados en esta experiencia. Se utilizan los mismos tres conjuntos de datos de entrenamiento basados en SIFT1M y SIFT10M, así como los seis conjuntos de consulta utilizados en la experiencia 5.5, adicionando seis nuevos sets de consulta aplicando diferentes niveles de distorsión de 0%, 10% y 20% al segundo y tercer conjunto de entrenamiento con el código del Apéndice B. Estos conjuntos serán utilizados con la distancia euclidiana para los algoritmos ANN.

Por otra parte, a partir de los conjuntos de entrenamiento y de consultas previos se generan nuevos conjuntos que serán utilizados con la distancia de Hamming. Para generar los nuevos conjuntos de entrenamiento se toman sólo los primeros 32 bytes (256 bits) de cada vector. Para los sets de consulta, primero se toman los primeros 32 bytes de cada vector y luego se les aplica el porcentaje de distorsión correspondiente.

Los tiempos se miden en microsegundos y se reportan en milisegundos, directamente sobre la llamada a los algoritmos, evitando posibles overheads introducidos por la abstracción.

Entrenamiento 200.000 elementos	**Pruebas Muestreo 10.000 elementos**			
	Grupo Disjunto	**Distorsionados**		
		0%	**10%**	**20%**
		Grupo-d0	**Grupo-d10**	**Grupo-d20**
SIFT1M	sift1m[1]	sift1m-d0	sift1m-d10	sift1m-d20
SIFT10Ms1	sift10ms1	sift10ms1-d0	sift1ms1-d10	sift1ms1-d20
SIFT10Ms2	sift10ms2	sift10ms2-d0	sift1ms2-d10	sift1ms2-d20

Tabla 5.2. Resumen de los sets de entrenamiento y sus sets de consultas correspondientes para distancia euclidiana. Cada set de consulta se encuentra asociado al set de entrenamiento de su misma fila. Cada columna de entrenamiento representa un grupo cuyos resultados son reportados como promedio.

5.7.2 Resultados obtenidos

Para cada uno de los cuatro grupos, se ejecuta el experimento con sus tres sets de consulta y se reporta a continuación el recall promedio resultante.

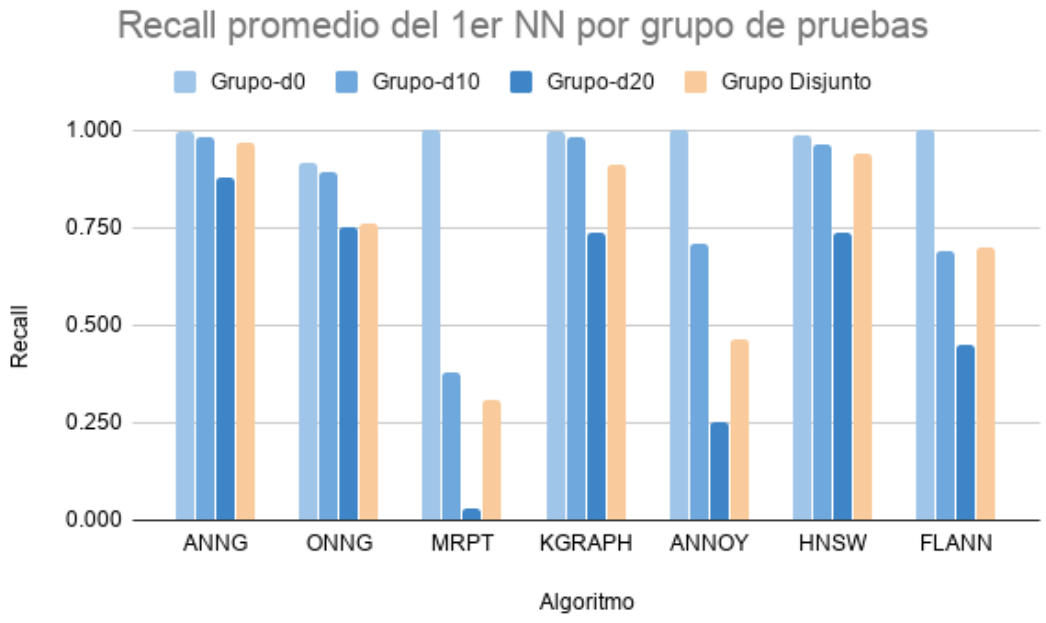

Figura 5.4. Recall promedio del primer vecino más cercano para cada algoritmo en cada set de consulta. Tonos de azul corresponden a set de consultas con distorsión, anaranjado para set de consultas totalmente disjunto.

1 provisto como set de consulta junto con SIFT1M

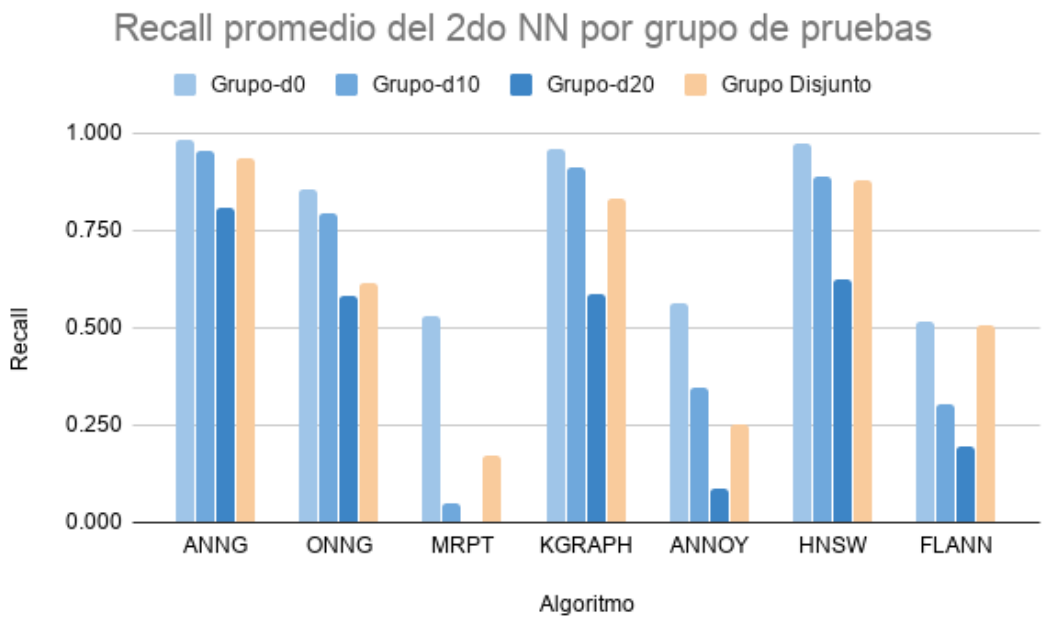

Figura 5.5. Recall promedio del segundo vecino más cercano para cada algoritmo en cada set de consulta. Tonos de azul corresponden a set de consultas con distorsión, anaranjado para set de consultas totalmente disjunto.

En las figuras 5.4 y 5.5 se observa cómo para la distancia euclidiana, casi todos los algoritmos obtienen peores resultados para los sets disjuntos, es decir, aquellos donde no existe una verdadera correspondencia entre el elemento query y los elementos del set de entrenamiento, que para los sets con 10% o menos de distorsión.

Estos resultados pueden ser parcialmente explicados no sólo por la cantidad de distorsión o cambio sino por la distribución del cambio entre los elementos de consulta y de entrenamiento. Para la distancia euclidiana, cuando los elementos de consulta son un simple extracto de descriptores disjuntos del set de entrenamiento, la distancia resulta aleatoria pero usualmente dominada por una gran diferencia concentrada en pocas dimensiones. Por otro lado, los descriptores generados por los algoritmos a partir de los cuadros del flujo de vídeo se distorsionan de una forma más suave, distribuyendo el cambio en un gran número de dimensiones.

Para la distancia Hamming, los resultados difieren del comportamiento observado previamente para los algoritmos ANNG y su derivado ONNG, como se observa en las figuras 5.6 y 5.7. Para todos los otros casos, el recall obtenido es menor en los juegos de consulta disjuntos. Cabe destacar que los valores absolutos de recall no son de importancia para este experimento sino la relación entre los resultados obtenidos para cada set de consulta con cada algoritmo.

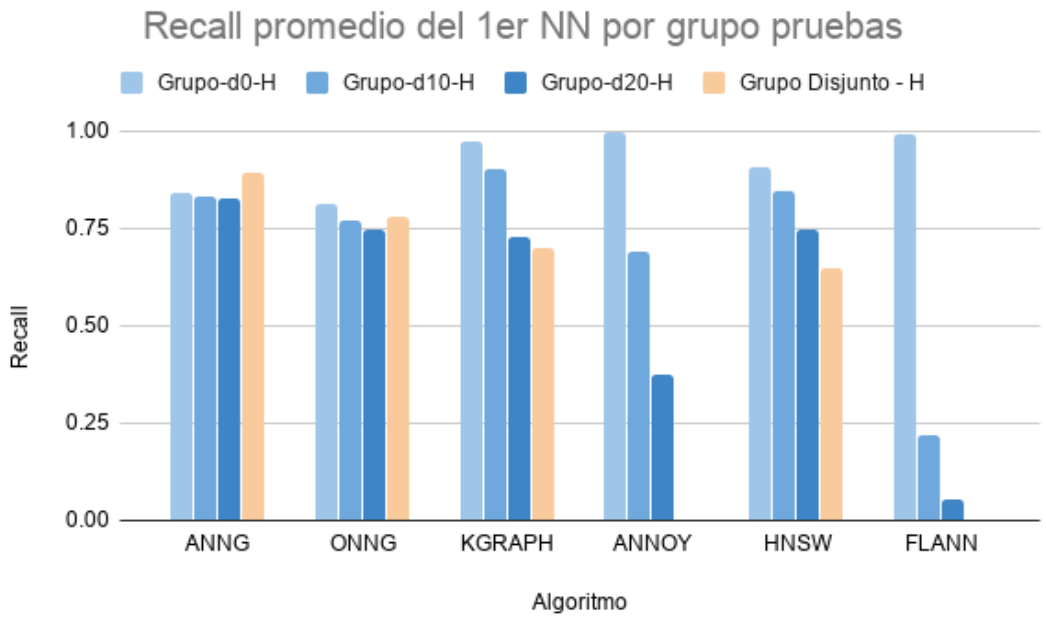

Figura 5.6. Recall promedio del primer vecino más cercano para cada algoritmo en cada set de consulta. Tonos de azul corresponden a set de consultas con distorsión, anaranjado para set de consultas totalmente disjunto.

En el caso de la distancia Hamming, la cercanía de los bits invertidos que constituyen la distorsión entre los elementos query y los elementos de entrenamiento parece tener influencia pero la confirmación o refutación de esta observación requiere de experimentación específica que excede el alcance de este trabajo de tesis.

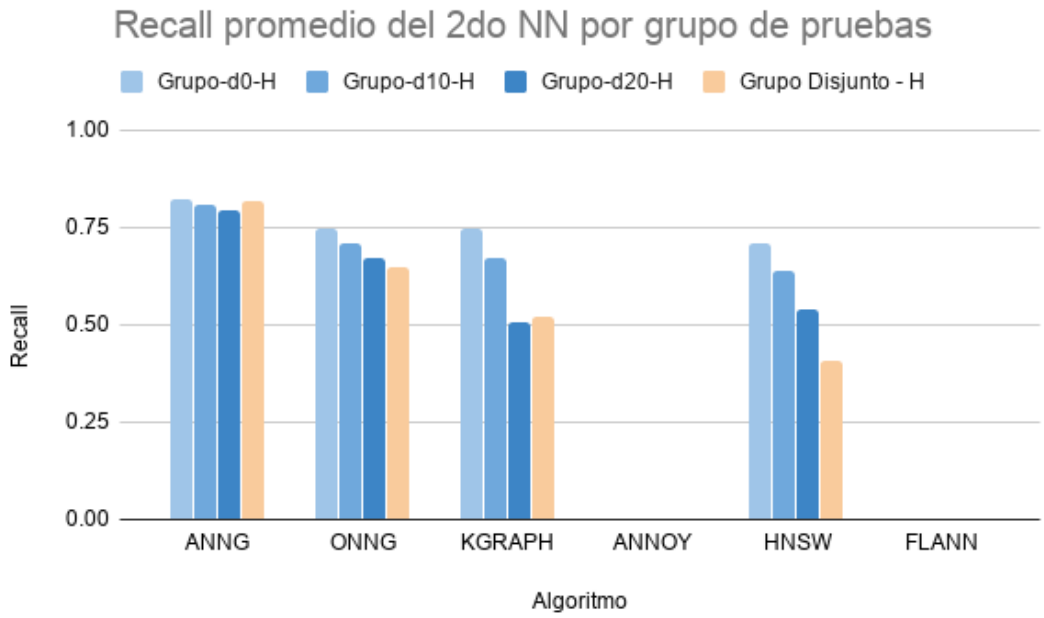

Figura 5.7. Recall promedio del primer vecino más cercano para cada algoritmo en cada set de consulta. Tonos de azul corresponden a set de consultas con distorsión, anaranjado para set de consultas totalmente disjunto.

5.8 Desempeño con datos con bajos niveles de distorsión

Dado que la similitud entre elementos de consulta y entrenamiento afecta su desempeño, como ilustran los resultados anteriormente presentados, se procede a compararlos bajo estas condiciones específicas.

5.8.1 Diseño del experimento

La tabla 5.3 resume los conjuntos de datos para probar y comparar el desempeño de los distintos algoritmos de ANN mencionados. Se utiliza el set SIFT1M en su totalidad como set de entrenamiento y se generan dos sets de consultas a partir de este. Al primero, para evaluar la distancia euclidiana, se le aplica una distorsión del 5%. Al segundo, para evaluar distancia Hamming, se le aplica una distorsión del 15%.

Entrenamiento 1.000.000 elementos	**Pruebas Muestreo 10.000 elementos**	
	Distorsión de 5% (l2)	**Distorsión de 15% (H)**
SIFT1M	sift1m-d05	sift1m-d15-H

Tabla 5.3. Sets de entrenamiento y consultas para le evaluación con bajos niveles de distorsión.

Se ejecuta el código del archivo bench.cpp del repositorio digital apéndice A, cuyo extracto principal se encuentra a disposición en el apéndice D, para cada algoritmo múltiples veces, ajustando sus parámetros de forma de obtener empíricamente los mejores resultados para cada nivel de procesamiento. En el mismo, se utiliza el algoritmo seleccionado para buscar los primeros dos vecinos más cercanos en el set de entrenamiento de cada elemento en el set de consultas.

Las métricas capturadas son el recall del primer y segundo vecinos más cercanos así como los tiempos de ejecución en microsegundos, reportados en mili-segundos o bien como queries por segundo.

5.8.2 Resultados obtenidos

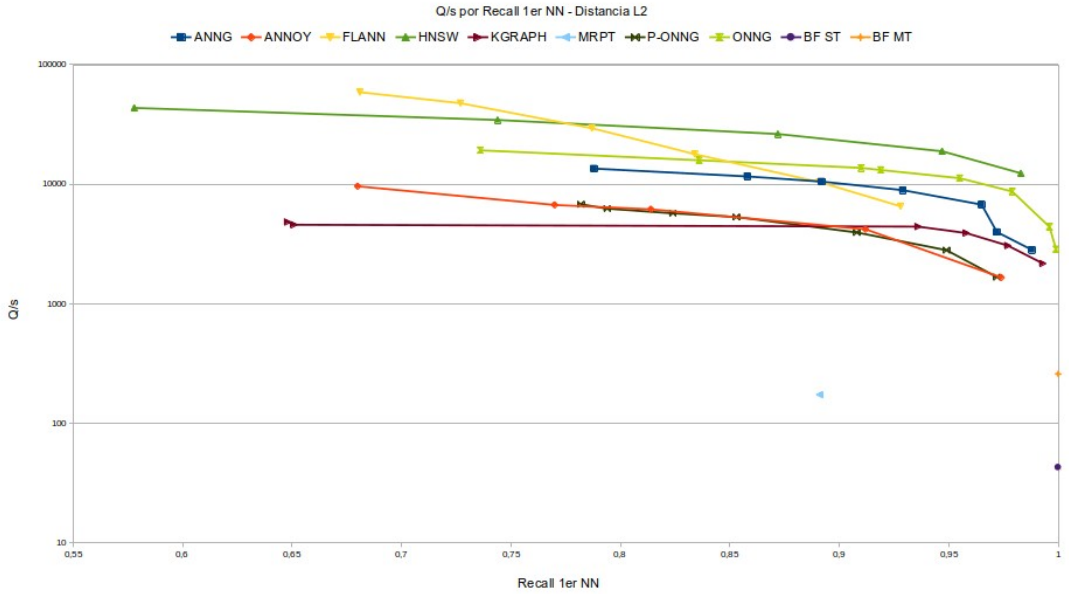

Figura 5.8. Cantidad de queries por segundo por recall del primer vecino más cercano, para distancias Euclidianas con un set de consultas con una distorsión del 5%.

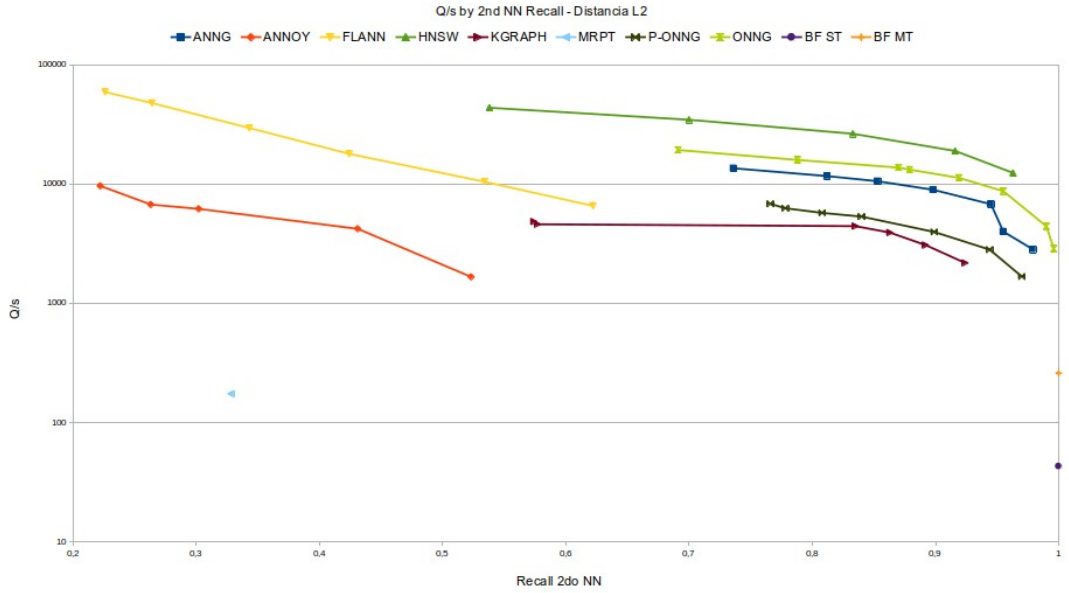

Figura 5.9. Cantidad de queries por segundo por recall del segundo vecino más cercano, para distancias Euclidianas con un set de consultas con una distorsión del 5%.

En la figura 5.8 se aprecia cómo varios algoritmos intercambian lugares para el recall del primer vecino más cercano (NN), especialmente en el rango de 80-90% y como algunos alcanzan resultados claramente superiores que cuando se los consulta con un número mayor de NN y datos de consulta disjuntos como los reportados en (Aumüller, Bernhardsson, y Faithfull 2018). El algoritmo P-ONNG (pseudo-onng) es un método alternativo implementado por el autor de ONNG para construir un índice ONNG de forma directa en vez de partiendo de un índice ANNG.

Por otro lado los resultados para el segundo NN como se observa en la figura 5.9 ilustra un comportamiento diferente donde cada algoritmo mantiene su orden relativo, con FLANN y ANNOY decayendo drásticamente. En general HNSW despliega una mejora significativa sobre ONNG a pesar de encontrarse casi a la par cuando son testeados en contextos más generales como los ya mencionados.

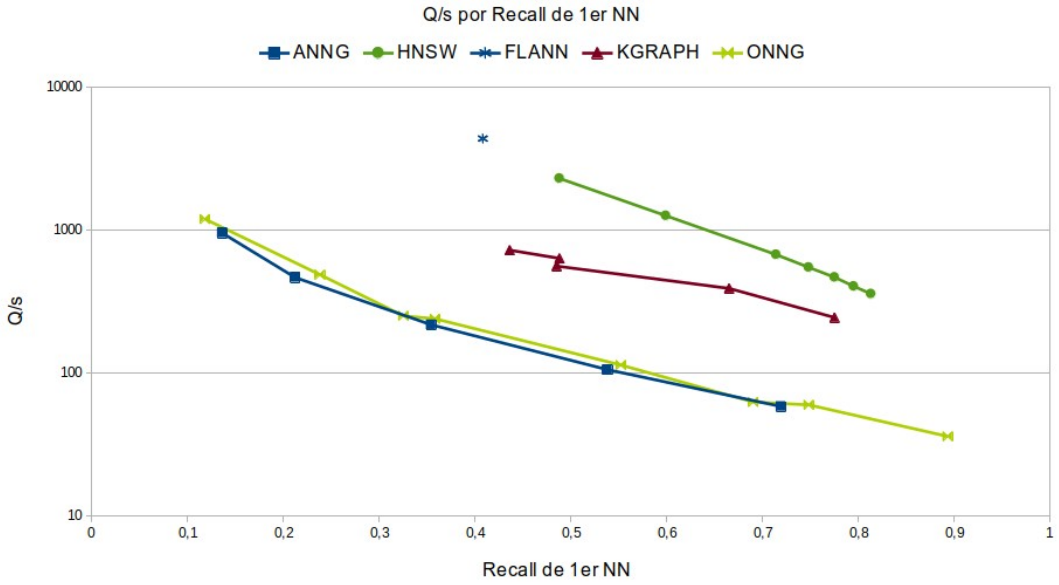

Figura 5.10. Cantidad de queries por segundo por recall del primer vecino más cercano, para distancias Hamming con un set de consultas con una distorsión del 15%.

En la figura 5.10 se observan resultados análogos para la distancia Hamming, con 15% de distorsión. Dado que ahora cada bit, en vez de cada byte/número, representa una dimensión, el incremento en la dimensionalidad se hace notar mientras que los niveles de desempeño de todos los algoritmos caen.

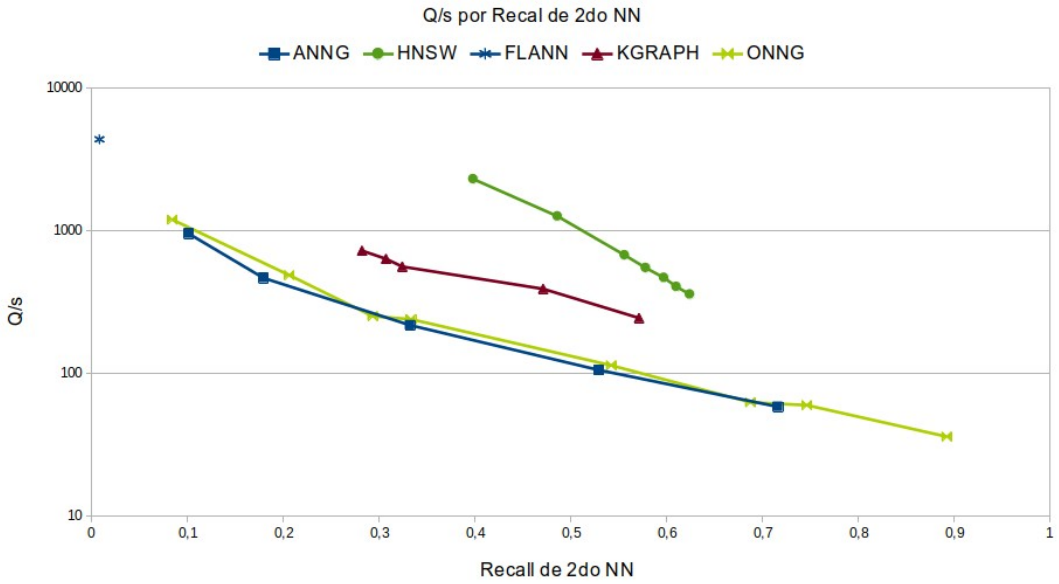

Figura 5.11. Cantidad de queries por segundo por recall del segundo vecino más cercano, para distancias Hamming con un set de consultas con una distorsión del 15%.

En este caso, se observa al algoritmo KGraph sobrepasando a ONNG por un margen considerable tanto para el primer como para el segundo vecino más cercano. Por su lado los algoritmos FLANN y ANNOY revisten una caída dramática de su recall a partir del segundo NN como se observa en la figura 5.11. En el caso de FLANN, se valida este resultado corriendo los mismos tests sobre dos implementaciones, la provista por OpenCV y la implementación oficial disponible al público en Github. En ambos casos los resultados fueron casi idénticos. Para el algoritmo ANNOY se contactó a sus desarrolladores pero no se obtuvo respuesta sobre el tema al momento de este escrito.

Adicionalmente, se puede observar en las figuras 5.12 y 5.13 el corrimiento promedio en el índice de los vecinos más cercanos obtenidos por los algoritmos, es decir, dado un elemento query cual es la diferencia en la posición entre los vecinos obtenidos por el algoritmo y los verdaderos vecinos más cercanos. Por ejemplo, dado el elemento query A se obtienen los vecinos E y F como candidatos a primero y segundo más cercanos por el algoritmo ANN. Sin embargo, los elementos E y F son en verdad el quinto y sexto elemento más cercano a A, por lo tanto la distancia en el índice será de 3 unidades.

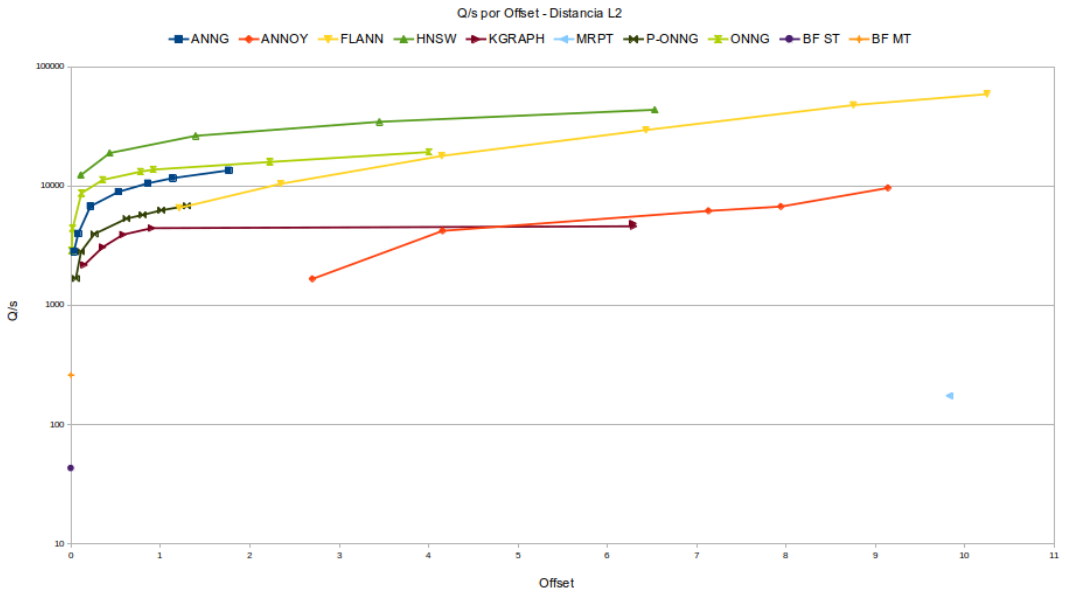

Figura 5.12. Cantidad de queries por segundo por corrimiento promedio de 1er y 2do NN para distancia Euclidiana.

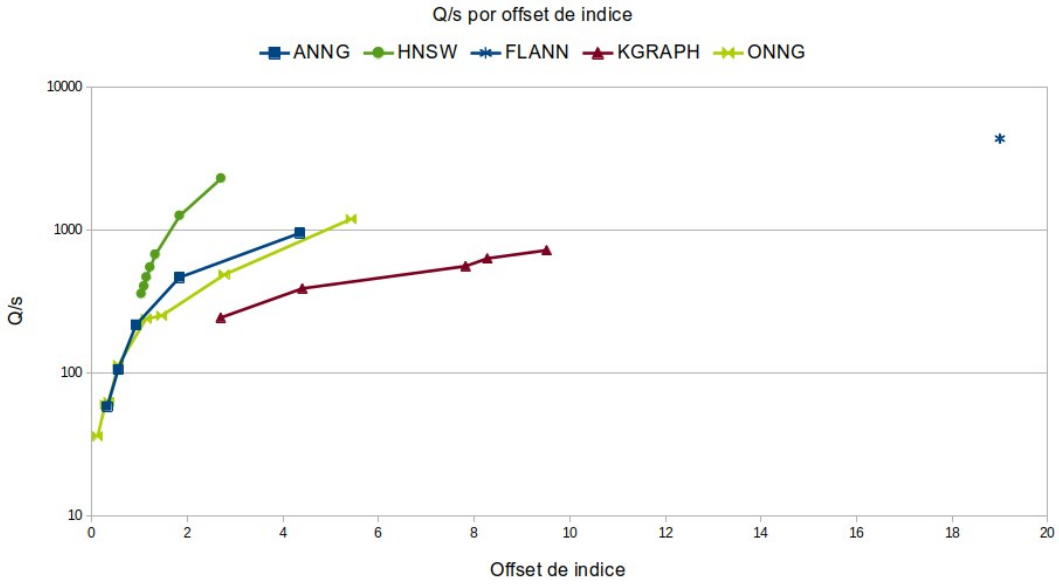

Figura 5.13. Cantidad de queries por segundo por corrimiento promedio de 1er y 2do NN para distancia Hamming.

Si se compara la cantidad de queries por segundo por la métrica de corrimiento se puede observar nuevamente una superioridad del algoritmo HNSW por sobre su primer

competidor ONNG. Si bien este último es capaz de lograr corrimientos muy reducidos, el impacto en su velocidad de ejecución resulta importante.

5.9 Validación del desempeño

Si se cruzan los resultados de cantidad de cuadros del flujo de vídeo simulado correctamente detectados por cada algoritmo del estudio 5.6 anteriormente detallado con la cantidad de queries por segundo que alcanzan los mismos en el estudio 5.8 para la misma configuración de parámetros, se podrá medir la relación final real entre la velocidad de los algoritmos y su desempeño en el subproceso de imágenes propuesto.

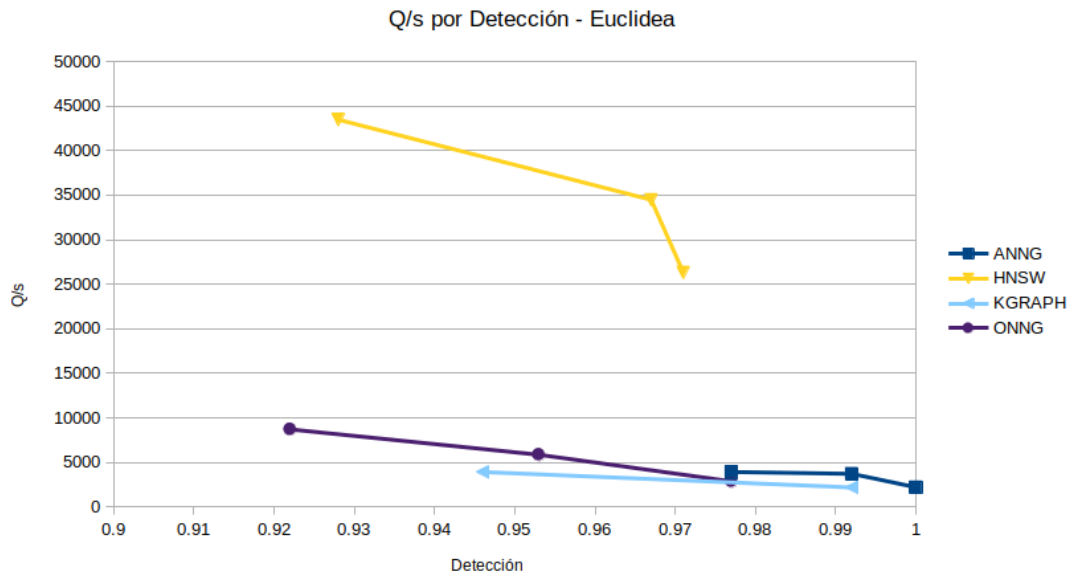

Figura 5.14. Cantidad de queries por segundo por el nivel de detección relativo. Se toma el mayor número de detecciones como 100% y ajusta el resto de los valores de forma relativa.

Se aprecia en la figura 5.14 la cantidad de queries por segundo para un subconjunto de algoritmos que mostraron propiedades interesantes en experiencias anteriores. La cercanía entre ANNG, KGRAPH y ONNG es en parte explicada por el alto nivel de redundancia contemplado en el subproceso de imágenes al utilizar hasta 200 POI por imagen objetivo. Aun así, ONNG observa una curva con pendiente más pronunciada permitiendo aumentar significativamente la velocidad si se sacrifica un pequeño porcentaje de precisión. Por otro lado se confirman las observaciones anteriormente realizadas que predecían una superioridad del algoritmo HNSW.

En general los resultados obtenidos indican que los esquemas de comparación tradicionales para algoritmos ANN pueden no resultar completamente representativos de su desempeño al aplicarlos al contexto de la RA y que las consideraciones propuestas permiten predecir y comparar con mayor exactitud.

5.10 Desempeño según el tamaño del set de datos

Ya se ha determinado que el nivel de variación en los descriptores y la cantidad de vecinos más cercanos reducida necesarios para el contexto de la RA afectan significativamente la velocidad y desempeño de los algoritmos de ANN evaluados.

5.10.1 Objetivo del estudio

Para lograr el objetivo de escalabilidad de la arquitectura integrada, resulta de interés evaluar el impacto tanto en recall como en velocidad de ejecución del incremento en la cantidad de elementos producido por el incremento de la cantidad de objetivos de aumentación.

5.10.2 Diseño del experimento

Partiendo del set de datos SIFT1M se preparan cinco set de consulta de tamaño variable a incrementos de 200.000 elementos sobre los cuales se aplica una distorsión de 5% cuando se evalúa la distancia Euclidiana y de 15% cuando se evalúa la distancia Hamming, como se resume en la tabla 5.4.

Entrenamiento	**Pruebas**									
	Distorsión de 5%					**Distorsión de 15%**				
SIFT1M	200K	400K	600K	800K	1M	200K	400K	600K	800K	1M

Tabla 5.4. Resumen de set de consulta con tamaños variables.

Se ejecuta el código del archivo bench.cpp del repositorio digital («Repositorio» s. f.) midiendo el recall del primer y segundo vecinos más cercanos así como el tiempo de ejecución en microsegundos, reportado en mili-segundos. En todos los casos se utiliza una computadora con Ubuntu Linux 19.10 equipada con una CPU AMD Ryzen 5 1600X (12 Mb L3 cache) y 16 Gb de RAM corriendo a 2667 Mhz con un ancho de banda aproximado de 20 Gbps.

Para cada algoritmo, los parámetros fueron seleccionados en forma empírica para producir la mejor relación entre velocidad y recall pero sin superar los 0.2 milisegundos por query siempre que sea posible o bien dejado al algoritmo en caso de poseer opción de autoconfiguración.

5.10.3 Resultados obtenidos

En las figuras 5.15 y 5.16 se observa la evolución de los tiempos de ejecución por query a medida que el set de datos de entrenamiento utilizado es de mayor tamaño. Los tiempos de ejecución absolutos de cada algoritmo no son importantes en este caso sino la tendencia de la curva de cada uno. Un algoritmo con buena escalabilidad debe observar una complejidad computacional altamente sub-lineal.

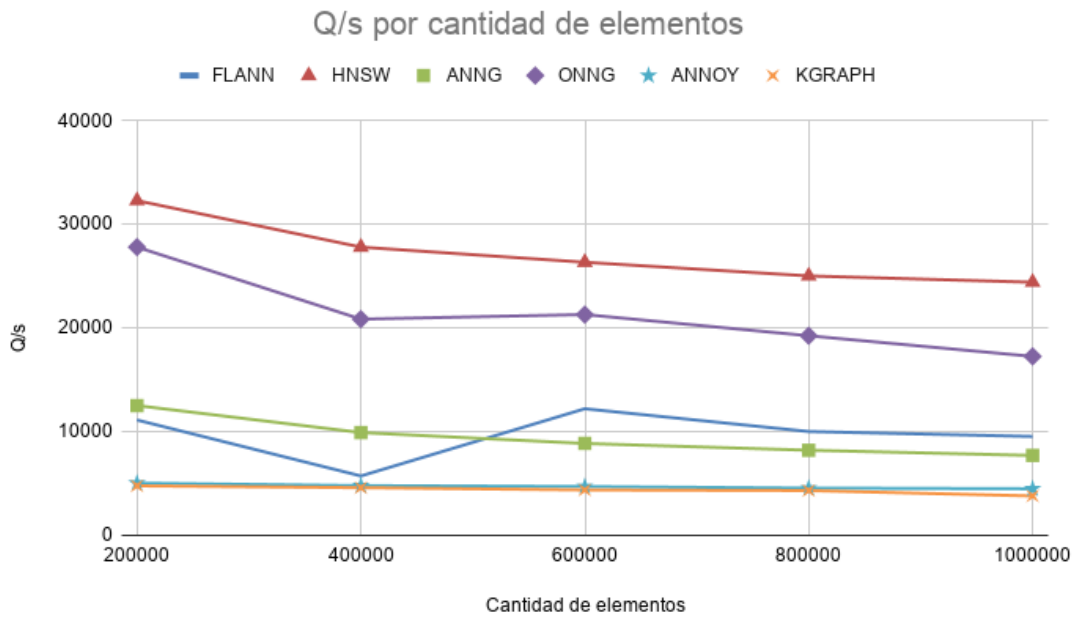

Figura 5.15. Curva de cantidad de queries por segundo de cada algoritmo para cada set de entrenamiento con tamaños incrementales. Se utiliza distancia Euclidiana.

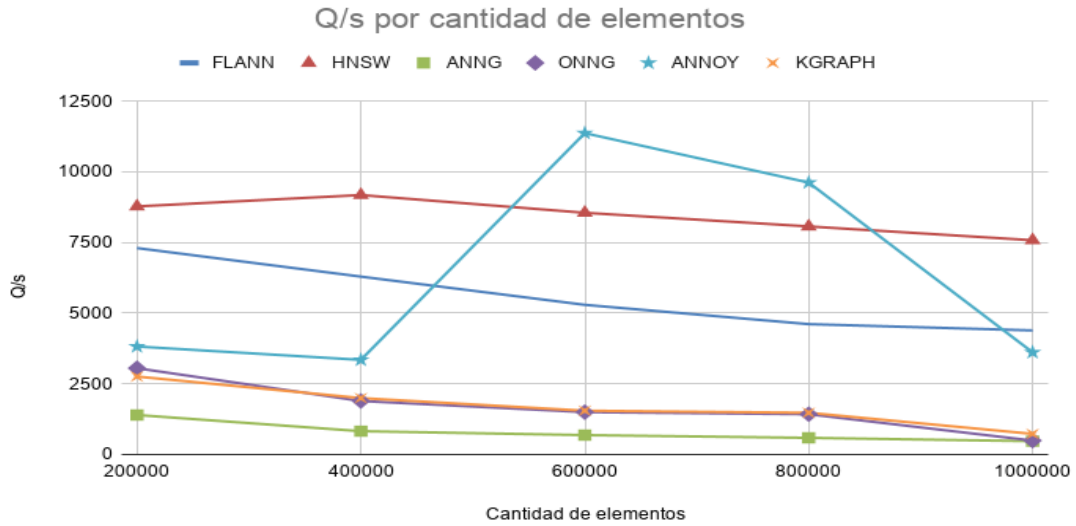

Figura 5.16. Curva de cantidad de queries por segundo de cada algoritmo para cada set de entrenamiento con tamaños incrementales. Se utiliza distancia Hamming.

Se destaca que los valores absolutos no son de importancia en el contexto de este experimento sino la tendencia y estabilidad de las curvas de cada algoritmo.

Se observa una relativa estabilidad y baja pendiente para la mayoría de los algoritmos con la excepción de FLANN y ANNOY. Esto se puede explicar parcialmente por la reducida cantidad de parámetros, las funciones de autoajuste de los mismos y la naturaleza aleatoria de la construcción de sus índices.

Adicionalmente resulta de importancia observar la estabilidad y evolución del recall de los diferentes algoritmos al trabajar con tamaños incrementales de set de entrenamiento.

Como se observa en las figuras 5.17, 5.18, 5.19 y 5.20 la mayoría de los algoritmos evaluados presenta un nivel de estabilidad en su desempeño aceptable, con pendientes fuertemente sublineales. Nuevamente el algoritmo ANNOY presenta ciertas inestabilidades mientras que cabe destacar que si bien ANNG resulta en el algoritmo más estable, se debe recordar que su relación entre tiempo de procesamiento y recall es notablemente inferior al de otros como ONNG o HNSW.

Si bien los valores absolutos de recall no son tan importantes para este experimento en particular, cabe resaltar que el bajo desempeño de los algoritmos FLANN y ANNOY para el segundo vecino más cercano en distancia Hamming, es consistente con observaciones previas.

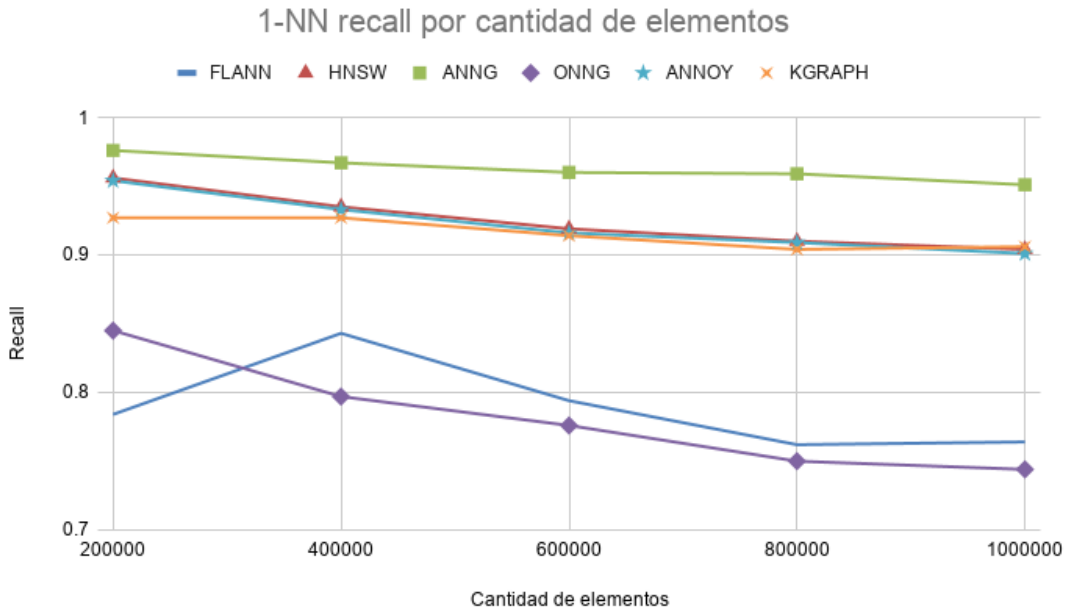

Figura 5.17. Curvas de evolución del recall para el primer vecino más cercano para cada set de entrenamiento de tamaño incrementales, utilizando distancia Euclidiana.

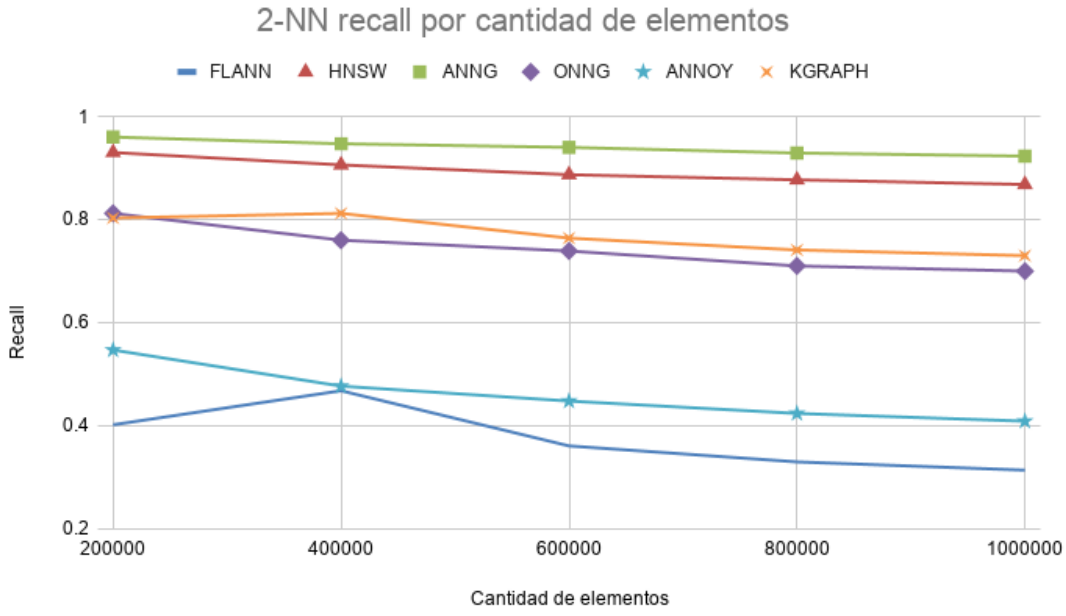

Figura 5.18. Curvas de evolución del recall para el *segundo* vecino más cercano para cada set de entrenamiento de tamaño incrementales, utilizando distancia Euclidiana.

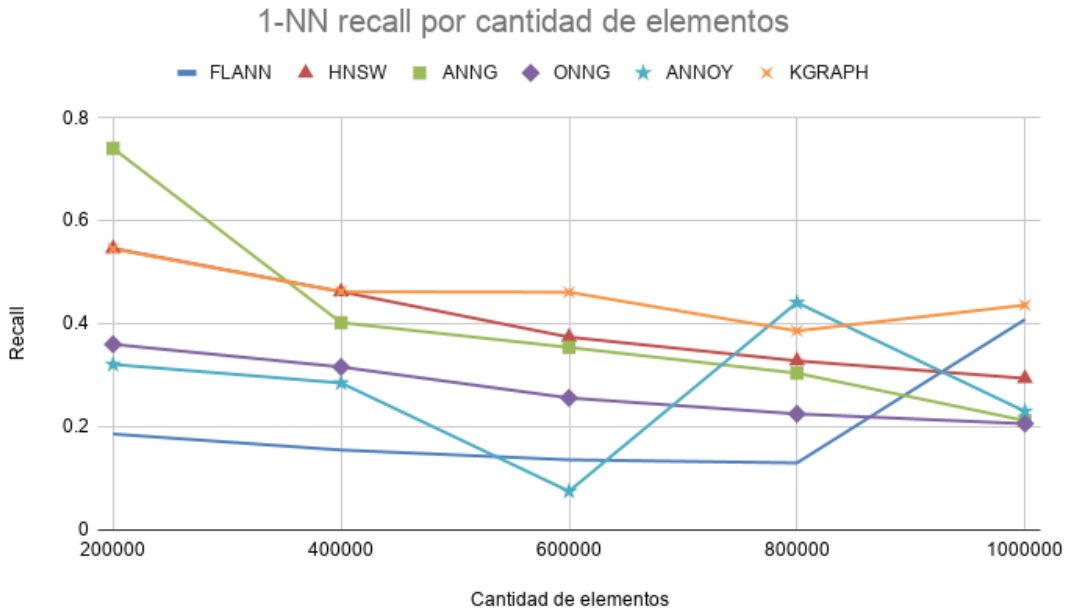

Figura 5.19. Curvas de evolución del recall para el *segundo* vecino más cercano para cada set de entrenamiento de tamaño incrementales, utilizando distancia Hamming.

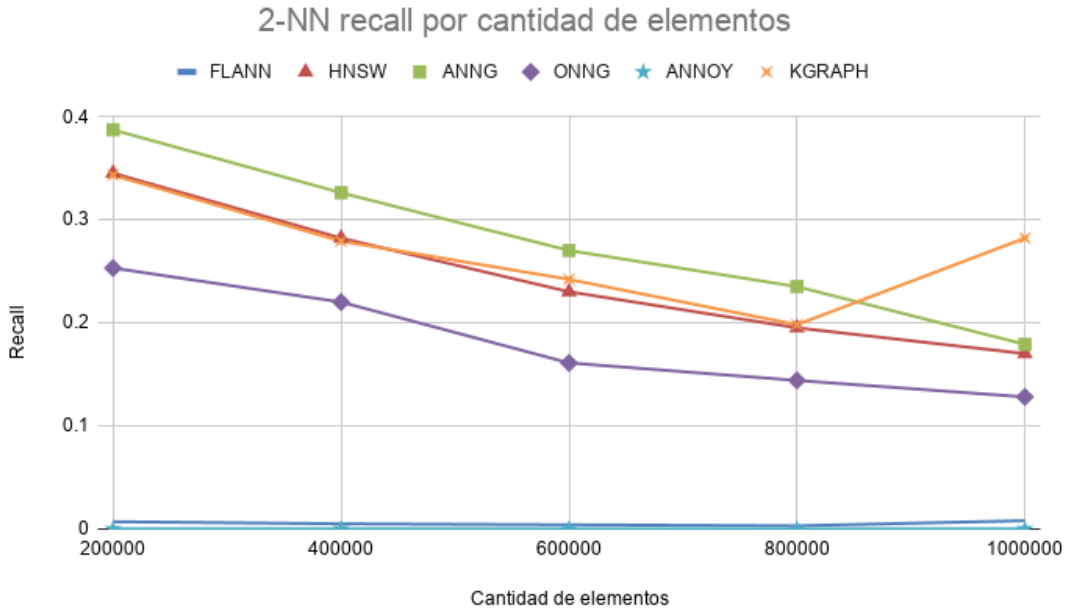

Figura 5.20. Curvas de evolución del recall para el *segundo* vecino más cercano para cada set de entrenamiento de tamaño incrementales, utilizando distancia Hamming.

Capítulo 6

Discusión y validación de resultados

En la primera sección de este capítulo se integran y resumen las decisiones de diseño fundamentadas en las distintas evaluaciones y análisis sobre las partes y pasos de los procesos integrados. En particular se discute la elección de algoritmos en torno al esquema de evaluación propuesto y la organización del flujo de ejecución a fines de aliviar el cuello de botella principal para lograr un sistema de RA integrado escalable. Luego, en la sección 2, se presenta una validación de los tiempos de ejecución del prototipo completo, integrando aumentación de imágenes arbitrarias, detección y reconocimiento de rostros e inferencia biométrica con 5.000 objetivos de aumentación.

6.1 Discusión de los resultados

Durante el transcurso de este trabajo se desarrolló un software prototipo experimental sobre el cual se realizó una serie de experiencias. De los datos recopilados fue posible inferir distintas afirmaciones que dieron lugar a una arquitectura de integración escalable de la RA basada en imágenes y rostros. La misma, al ser aplicada como framework otorga a un desarrollador la capacidad de aplicar aumentación de imágenes arbitrarias así como la detección de rostros y el reconocimiento facial. A su vez, integra una algoritmia escalable probada con 5.000 imágenes (equivalente a 1.000.000 de rostros), superando lo probado o soportado por otros sistemas existentes. Finalmente, la arquitectura también integra facilidades de estimación de información a partir de rostros humanos como rango etario, género y expresión facial, ofreciendo una interfaz que permita la integración de futuros algoritmos. Esta tecnología de rápida expansión resulta clave en contextos de RA como lo demuestra la actual necesidad de obtener la identidad y la temperatura corporal de un individuo minimizando el contacto y cercanía física lo más posible debido a la pandemia que nos aqueja.

6.2 Diseño de la integración y distribución del flujo de ejecución

Dado el subproceso propuesto para la aumentación de imágenes arbitrarias, de los resultados experimentales se obtiene que tanto el algoritmo ORB como la combinación del algoritmo ORB y el algoritmo FREAK otorgan una relación de precisión sobre velocidad satisfactorio a las necesidades de escalabilidad y restricciones temporales planteadas. Si bien los algoritmos probados fueron los que cuentan con una implementación de software estable, abierta y eficiente en C++, los mismos no representan una lista exhaustiva y por lo tanto se plantea la posibilidad de mejora al incluir nuevos algoritmos que revistan propiedades prometedoras cuando existan implementaciones estables, abiertas y eficientes.

De igual forma, de los resultados experimentales se obtiene que el principal cuello de botella al momento de incrementar la cantidad de imágenes objetivos de aumentación reviste en el paso de búsqueda de correspondencias entre descriptores.

En cuanto al subproceso propuesto para la aumentación de rostros se determina utilizar redes neuronales convolucionales entrenadas con el error por tripletas para el reconocimiento debido a las particularidades de los descriptores que producen, volviéndolos similares a los descriptores SIFT en cuanto a la búsqueda de correspondencias se refiere. Se establece también que el cuello de botella recae en el mismo paso de búsqueda de correspondencias y que los pasos de reconocimiento así como la inferencia biométrica deben ejecutarse en paralelo de forma asíncrona para lograr mantener la restricción de tiempo real.

Se resuelve entonces que el cuello de botella general para la arquitectura integrada en términos de cantidad de objetivos de aumentación recae en la búsqueda de correspondencias. El mismo se alivia mediante la utilización de algoritmos de búsqueda ANN de alto desempeño que son evaluados especialmente.

El ahorro de tiempo de procesamiento obtenido por el alivio al cuello de botella en el paso de búsqueda de correspondencias así como la utilización de bucles de seguimiento y esquemas asíncronos son los que permiten entonces la utilización de ambos tipos de RA bajo una misma arquitectura integradora a la vez que se ofrece un potencial de escalabilidad superador.

6.3 Marco de evaluación para algoritmos de búsqueda de correspondencias (ANN)

De los resultados experimentales se obtiene que las métricas de comparación populares para los algoritmos ANN no resultan completamente representativas para su uso en el

contexto particular de la RA. Se plantea entonces un marco de evaluación específico. Este nuevo marco propone medir el desempeño de los algoritmos mediante el recall de sólo los dos primeros vecinos más cercanos evaluando sobre datos confeccionados especialmente.

Los mismos se confeccionan de la siguiente manera:

1. Dado un conjunto de datos general compuesto por descriptores, se toma una muestra de los mismos la cual será la base de los conjuntos de datos de consulta.
2. Los datos de la muestra se distorsionan aplicando un porcentaje de cambio de acuerdo a la métrica de distancia y algoritmos a utilizar en la evaluación. Inicialmente se propone un 5% de distorsión para distancias Euclidianas y un 15% de distorsión para distancias Hamming, sin embargo, se recomienda ajustar estas proporciones si se utilizara algoritmos distintos a los evaluados en el presente trabajo.

6.4 Selección de algoritmos ANN

Bajo el marco de evaluación propuesto se observa que determinados algoritmos exhiben un comportamiento diferente al observado bajo un marco más tradicional y genérico como los citados con anterioridad. En particular, se predice que HNSW debería otorgar una mejor relación de recall por velocidad de ejecución, superando a otros algoritmos como ONNG pese a demostrar resultados casi equivalentes en los marcos de evaluación citados.

6.5 Validación de tiempos de ejecución del prototipo completo

Habiendo determinado la combinación de algoritmos apropiada para cada paso de los subprocesos integrados así como el algoritmo de mejor desempeño a la hora de aliviar el cuello de botella, se procede a validar los tiempos de ejecución. Se ejecuta el prototipo completo con 5000 imágenes de obras de arte descritas en la sección 5.2.2, una imagen adicional que presenta buenas cualidades para su aumentación y 4 rostros como objetivos de aumentación, representando un nivel de carga muy elevado. Cabe destacar que cada imagen es representada por 200 descriptores, mientras que un rostro es representado por un único descriptor bajo la familia de redes neuronales seleccionada. Esto implica una relación de equivalencia en términos de procesamiento para la búsqueda de correspondencias de 200 a 1.

El prototipo completo responde a la arquitectura integrada ilustrada en la figura 4.2 permitiendo la aumentación de imágenes arbitrarias, detección de rostros,

reconocimiento facial e inferencia de datos biométricos. En este prototipo demostrador la inferencia de datos biométricos se compone de tres inferencias, rango de edad, género y expresión facial.

Se resume en la figura 6.1 las estadísticas de tiempos de ejecución del prototipo completo ejecutando en una computadora de escritorio con Ubuntu 19.10, un procesador Ryzen 5 1600x y 16Gb de RAM a 2667Mhz. Se resalta en negrita tres valores en particular:

- la jerarquía Detect-Avg-Match, midiendo el tiempo promedio utilizado para encontrar correspondencias sobre POI de imágenes que gracias a la utilización del algoritmo HNSW resulta en valores menores a otras categorías pese a ser identificado como el cuello de botella
- la jerarquía Track-Avg que mide el tiempo promedio que demora el algoritmo de seguimiento de imágenes para un cuadro
- la medición Face track time que mide el tiempo promedio que demora el algoritmo de seguimiento de rostros para un cuadro

Los resultados experimentales sobre el bucle de detección completo del subproceso basado en imágenes valida las predicciones del marco de evaluación propuesto, observando una superioridad del algoritmo HNSW en la tarea de búsqueda de correspondencias. Cabe destacar que los resultados obtenidos para el procesamiento de imágenes con descriptores SIFT y distancia euclidiana resultan extrapolables al caso de búsqueda de correspondencias de descriptores de rostros humanos, ya que ambos revisten el mismo orden de posible distorsión.

```
Image Engine Statistics
Avg Good Matches: 10
  Avg Distance: 28.2941
  Avg Min Distance: 15.75
  Avg Max Distance: 33.375
Avg Frame time: 13.451445 ms
Avg FPS: 74.341461 fps
Max Frame time: 44 ms
Detect times:
  Avg = 21.979849 ms
    Detect = 13.841746 ms
    Match = 7.752278 ms
      Homography = 0.011495 ms
Track times:
  Avg = 2.337142 ms
  Min = 1 ms
  Max = 4 ms
Grab frame: 0.000000 ms
Copy frame: 0.002207 ms
Tracker: 1.664224 ms
Homography: 0.196161 ms
Outlier clean up: 0.000004 ms
Shape checks: 0.001035 ms
Perspective transform: 0.000028 ms
3D outlier clean up: 0.000902 ms
Solve PnP: 0.462043 ms
Draws: 0.000000 ms
Swaps: 0.000173 ms
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Face Engine Statistics
Face detect time:
      Avg = 48.370454 ms
Face track time:
      Avg = 5.517200 ms
```

Figura 6.1. Estadísticas de tiempos de ejecución del prototipo demostrador de la arquitectura completa ejecutando una prueba ad hoc con 5001 objetivos de aumentación. Las sangrías o jerarquías representan la descomposición del tiempo observado, por ejemplo: el tiempo Detect-Avg ~= 22ms se compone por approx. 14ms de detección (Detection) y 8ms de búsqueda de correspondencias (Match).

Capítulo 7

Integración con herramientas de autor

7.1 Procesos de una herramienta de autor

Las herramientas de autor facilitan la creación y explotación de aplicaciones de RA sin la necesidad de contar con conocimientos de programación. Proveen de una interfaz amigable a los usuarios para generar las bases de datos que almacenarán las imágenes o rostros objetivo de aumentación de cada aplicación, y asociarles la información añadida.

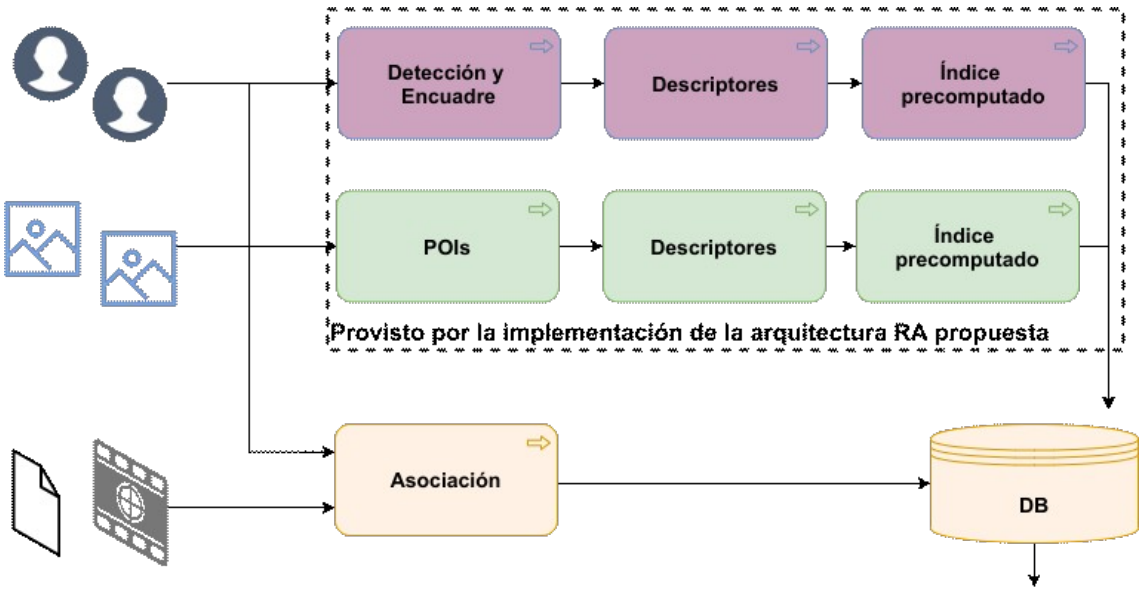

Figura 7.1. Diagrama conceptual de la funcionalidad de las herramientas de autor en relación con la propuesta de arquitectura integrada al proveer índices precalculados como parte de la base datos.

Como se observo en los capítulos 2 y 3, los procesos para la aumentación tanto de imágenes como de rostros hacen uso de bases de datos con los descriptores de cada objetivo de aumentación. Como se observa en la figura 7.1, estos valores deben ser computados utilizando exactamente los mismos algoritmos y configuraciones que utiliza la arquitectura integrada propuesta.

Adicionalmente, los algoritmos encargados de la búsqueda de correspondencias entre descriptores requieren de un pre-procesamiento o generación de alguna clase de estructura de datos normalmente denominada como índice. Dicha generación debe realizarse sólo una vez para cada conjunto de objetivos de aumentación y su complejidad computacional es elevada. Por estas razones se propone la realización de dichas tareas en las herramientas de autor, sacando provecho del gran poder de procesamiento elástico disponible en la nube, como se comunica en (N. Mangiarua, Ierache, y Abásolo Guerrero 2019). Para ello, se requiere de funcionalidad de generación, lectura y escritura de estructuras de datos como parte de la implementación de la arquitectura integrada que pueda ser utilizada tanto por las herramientas de autor como por las aplicaciones de explotación.

7.2 Herramientas de autor: Sistema de Catálogos

Entre las diversas herramientas de autor existentes destacamos un desarrollo propio que llamamos Sistema de Catalogos Virtuales Aumentados. Dicho sistema de catálogos consiste en dos aplicaciones, una herramienta de autor web y un visor para dispositivos mobiles como se observa en la figura 7.2.

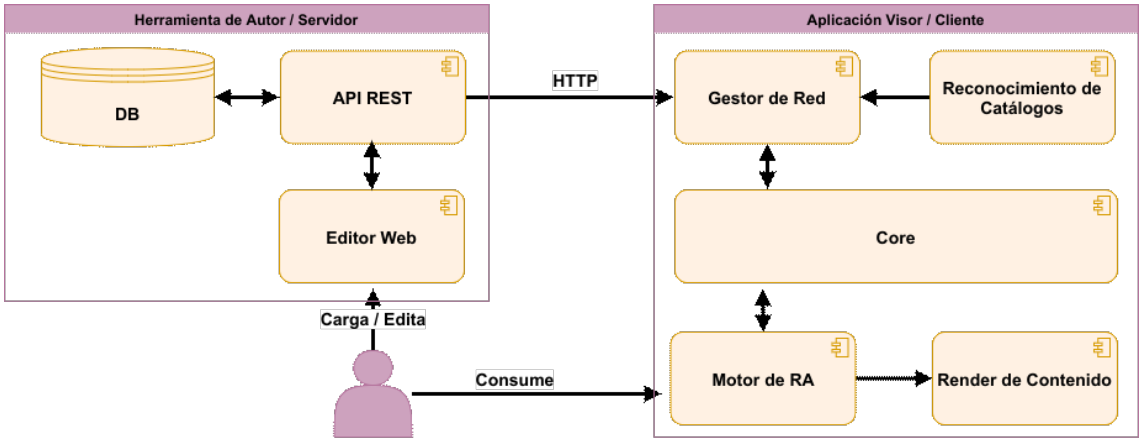

Figura 7.2. Diagrama conceptual de la arquitectura del sistema de catálogos virtuales aumentados.

La aplicación web cuenta con una típica arquitectura en capas desarrollada con las tecnologías Spring y Hibernate. Cuenta con un front-end para usuarios finales que permite la carga de objetivos de aumentación y sus contenidos a aumentar en unidades de trabajo que se denominan “catálogos”.

Por su parte, el visor o aplicación cliente cuenta con una arquitectura modular basada en eventos desarrollada sobre la herramienta Unity3D. La misma permite la descarga de un catálogo mediante la lectura de un código QR y la subsiguiente aumentación de los objetivos y contenidos que este incluya.

Exponiendo un API Rest, el sistema también permite interactuar con aplicaciones clientes externas, facilitándo dichos conjuntos de objetivos de aumentación o catálogos. Este esquema cliente-servidor bajo un esquema de API fácilmente extensible, permite su adaptación para la explotación en contextos específicos como los de la emergentología o la educación gamificada. Cabe destacar que el API Rest expone todos los datos presentes en la aplicación web, permitiendo a dichos clientes operar de forma completamente independiente, si así lo quisieran, una vez realizada la descarga inicial.

7.3 Escalabilidad en la herramienta de autor mediante templates

Una problemática observada durante la explotación del sistema de catálogos y extrapolable a las herramientas de autor en general, es como mejorar la usabilidad cuando la cantidad de objetivos de aumentación y/o contenidos se incrementa, es decir, cuando crece la escala del sistema. En situaciones donde el usuario pretende aumentar cientos o miles de objetivos, resulta de interés el reducir la carga de trabajo al momento de incorporar cada uno de los objetivo al sistema. De igual forma, dada la escalabilidad de la arquitectura integradora propuesta, se incrementa la necesidad de contar con soporte por parte de las herramientas de autor para poder aprovechar eficazmente esta ventaja.

La solución propuesta en este caso consiste en agregar una capa de meta aumentación de contenidos, introduciendo el concepto de template de aumentación de la realidad a herramientas de autor. Se publica en (N. Mangiarua et al. 2019; 2018; 2017; Becerra et al. 2018) la implementación de este concepto al Sistema de Catálogos Virtuales Aumentados. Dentro de este sistema, un template permite a un usuario con un nivel de conocimiento específico básico de la RA, definir una estructura base de los contenidos que serán asociados a cada objetivo de aumentación en un determinado contexto de aplicación. Permite especificar tanto la cantidad y tipos de contenidos a aumentar (texto, imagen, audio, objetos 3D, etc.), así como sus transformaciones geométricas (posición, rotación, escala), y sobre todo, un nombre y orden de visualización relativo al resto de los contenidos para un objetivo de aumentación. Además, también permite definir un término que aplica sobre el concepto mismo objetivo de aumentación, otorgándole un sentido semántico propio del área temática para el cual se crea.

Una vez creado, un template puede ser aplicado sobre un catálogo aumentado dentro de la herramienta. El template permite mantener un formato uniforme entre todos los objetivos de aumentación, así como simplificar la carga del material a aumentar por parte de otros usuarios, sin necesidad de conocimiento específico de la RA, al abstraer los conceptos a terminologías propias del tema de explotación de dicho catálogo.

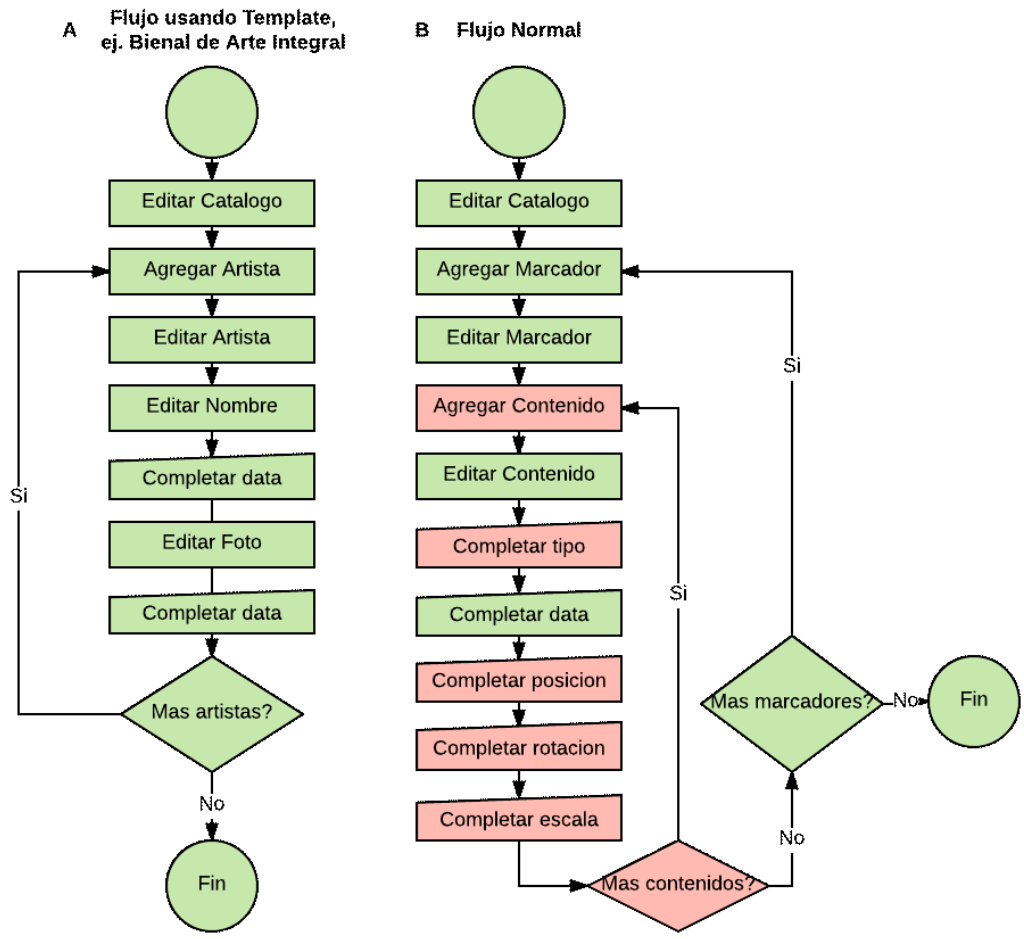

Figura 7.3. Comparación de flujo de acciones por parte del usuario final cuando se utiliza un template de RA y cuando no se lo utiliza sobre una herramienta de autor.

En la figura 7.3 se observa la diferencia entre los flujos de acciones a tomar por un usuario final para asociar contenido aumentado a un objetivo de aumentación cuando se utiliza un template. En este caso se ejemplifica utilizando un template preparado para la aumentación de obras plásticas en el marco del catálogo para una exposición o museo de arte. Cabe destacar cómo al utilizar el template los objetivos de aumentación ya no se consideran más como “marcadores”, terminología propia del contexto de la RA, sino como “artistas”. A la vez, cada contenido pasa a tener un orden y nombre descriptivos para el contexto.

La utilización de las templates de RA no sólo contribuye a disminuir tanto la barrera de conocimiento necesario para la utilización de una herramienta de autor sino que también contribuye a disminuir drásticamente la cantidad de trabajo, y en consecuencia tiempo, que requiere un usuario para cargar los objetivos de aumentación y los contenidos a

aumentar en grandes cantidades, aprovechando la escalabilidad de la solución propuesta.

7.4 Unión de la arquitectura integrada propuesta con herramientas de autor

Dada la incorporación de templates de RA al sistema de catálogos, se observa en la figura 7.4 los puntos donde la arquitectura integrada se puede unificar al sistema. En la herramienta de autor, la misma provee la funcionalidad necesaria para detectar y describir tanto rostros como POI de imágenes, a la vez que permite precomputar los indices del algoritmo HNSW para ser almacenados directamente en la base de datos central. En la aplicación visor, la arquitectura se une como pieza central del motor para la detección y aumentación de imágenes y rostros de acuerdo a los objetivos cargados en la herramienta de autor y transmitidos como catálogo o base de datos local. Ademas de la detección y reconocimiento, la arquitectura provee la matriz necesaria para dibujar elementos virtuales sobre el flujo de vídeo posicionados de forma coherente, junto con la información biométrica inferida.

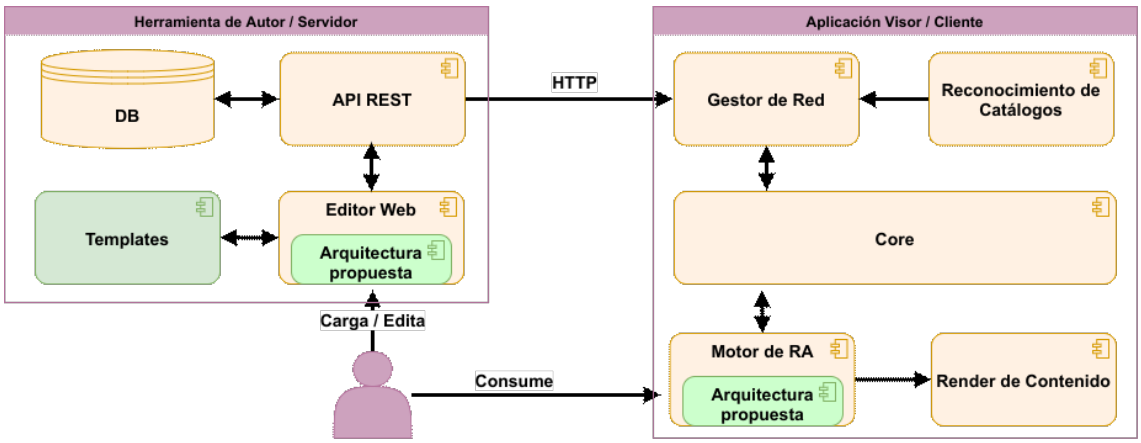

Figura 7.4. Diagrama conceptual de los módulos del sistema de catálogos con la incorporación de templates y resaltados los puntos de unión con la arquitectura integrada propuesta.

Capítulo 8

Conclusiones, aportes y trabajo futuro

En este capítulo final se presentan primero las conclusiones en la sección 1, los aportes a proyectos en la sección 2, el detalle de la producción científica en la sección 3, se finaliza en la sección 4 con la discusión de futuras líneas de investigación.

8.1 Conclusiones

Este trabajo de tesis propone una arquitectura escalable que integra la RA basada en imágenes arbitrarias con la detección y reconocimiento de rostros humanos junto con la inferencia de datos biométricos a partir de ellos.

Partiendo desde la definición de los procesos para RA basada en imágenes y rostros, en los capítulos 2 y 3 se determina cuales son los pasos necesarios para su ejecución. Se estudia la complejidad computacional teórica de cada proceso junto con la distribución de carga de procesamiento relativa. Estableciendo el algoritmo ORB o la combinación de ORB y FREAK como alternativas viables para la detección y descripción de POI en imágenes, se identifican los pasos de búsqueda de correspondencias entre descriptores, tanto de imágenes (POI) como de rostros como los cuellos de botella de cada proceso. Se selecciona para la descripción de rostros algoritmos de redes neuronales convolucionales entrenadas con el error por tripletas que producen descriptores continuos de 128 dimensiones y se establece que este paso debe ser implementado de forma asíncrona debido a su tiempo de ejecución.

Se presenta el diseño de una arquitectura integrada en el capítulo 4, compuesta por cuatro bucles en un flujo de ejecución alternante con derivación de tareas asíncronas en un esquema de ejecución paralelo para las tareas de descripción de rostros e inferencia de información biométrica. Los algoritmos previamente seleccionados permiten considerar el cuello de botella de cada proceso integrado como un mismo problema el cual se propone aliviar con el uso de algoritmos de búsqueda ANN.

Para lograr la escalabilidad de la arquitectura integrada, en el capítulo 5 se realizan una serie de experimentos para analizar comparativamente la velocidad, precisión y estabilidad de distintos algoritmos de búsqueda ANN, estableciendo un marco de evaluación y conjuntos de datos específicos para el contexto de la RA basada en imágenes y reconocimiento de rostros. Se establece y valida la superioridad del algoritmo HNSW para la tarea en este contexto particular, difiriendo de los resultados provistos por otros autores para contextos generales.

También se obtiene como aporte secundario un nuevo esquema de evaluación de algoritmos de búsqueda de vecinos más cercanos aproximados específico para el contexto de la RA. El mismo, presentado en la sección 5.3, implica la evaluación de esta familia de algoritmos utilizando sets de datos donde la variación entre los elementos query y los elementos de entrenamiento, base u originales sea reducida. Se establece un porcentaje de variación de entre el 5% y el 15% de acuerdo al tipo función de distancia utilizada pero se recomienda ajustar estos valores según el algoritmo específico que genera los elementos. Con estos sets de datos se propone un esquema donde los algoritmos de búsqueda de ANN sean comparados por su recall en los dos primeros vecinos.

Se crea un prototipo demostrador experimental que implementa la arquitectura propuesta en C++, compatible con cualquier plataforma para la que se lo compile. La misma optimiza la utilización de múltiples hilos de procesamiento y permite la posibilidad de integración con sistemas de más alto nivel para la generación y explotación de aplicaciones de RA. En la misma se implementa una interfaz abstracta que permite la incorporación dinámica de algoritmos de inferencia biométrica que serán automáticamente ejecutados de manera paralela y asíncrona.

8.2 Aportes a proyectos

Si bien se ha realizado durante etapas iniciales del trabajo de tesis una transferencia tecnológica a la industria aeroespacial, extensión al sector de producción audiovisual y aportes al campo de la educación gamificada, este aporte final permitiría mejorar significativamente las prestaciones de los productos frutos de dichos trabajos.

En este marco se realizó en el año 2018 una actividad de extensión temprana del prototipo de la tecnología de templates sobre el sistema de catálogos virtuales aumentados previamente citado para implementar una aplicación de aumentación de obras plásticas para la exposición bienal de arte realizada en la Universidad Nacional de La Matanza. Esta actividad no sólo permitió refinar y confirmar el aporte de usabilidad del sistema de templates sobre una herramienta de autor, sino que se exalta la necesidad de un incremento de la escalabilidad que permita aumentar un mayor número de obras. A su vez refuerza la necesidad de contar con la capacidad de reconocimiento facial, la

cual permitiría la aumentación directa de los artistas, tanto en vivo como en posters y fotografías.

Por otro lado, bajo el marco del programa de Financiación de Fase Cero de la Fundación Sadosky, con aprobación del Ministerio de Ciencia y Tecnología de la Nación se desarrolla en conjunto a un emprendimiento privado parte de la unidad de desarrollo en el marco del plan estratégico de desarrollo satelital de la CONAE un demostrador tecnológico combinando la RA con el posicionamiento GPS e información geoespacial publicado en (J. Ierache, Urien, y Mangiarua 2018). El mismo buscó demostrar cómo la RA basada en la posición geográfica en conjunto con la información de sensores de los dispositivos modernos permiten guiar el desplazamiento de los usuarios y presentarles información geolocalizada sin interrumpir su actividad laboral o recreativa. Si bien este demostrador hace uso de una forma de RA basada en la posición, la misma se beneficiaria con el agregado de las capacidades de aumentación basada en imágenes con el fin de reconocer fachadas de edificaciones a medida que el usuario se desplaza. La gran escalabilidad de la solución propuesta permitiría no sólo la aumentación de mobiliarios que resulten de explícito interés para el usuario, sino también el reconocimiento de edificaciones en posiciones clave que permitan contribuir al posicionamiento geográfico del usuario en entornos donde las capacidades de GPS o servicios similares se vean momentáneamente impedidas. Finalmente, una alta escalabilidad permitiría también la captura de un mismo edificio en distintos ángulos y condiciones de iluminación, como de día y de noche, solventando las dificultades inherentes de esta tecnología a dichos cambios cuando sólo cuenta con una captura o ejemplo.

También se publica en (J. Ierache et al. 2018) un framework basado en catálogos virtuales aumentados que facilita la creación de juegos didácticos para entornos educativos. Este framework se vería particularmente beneficiado por la integración de reconocimiento facial que permita identificar a los distintos jugadores mientras que se infiere información de los mismos. Por ejemplo, se puede estimar el grado de excitación o interés que los usuarios, en este caso estudiantes, presentan por la actividad, resultando en información de gran valor para el personal docente.

8.3 Producción científica

8.3.1 Revistas, series internacionales y capítulos de libro

- Mangiarua, Nahuel A., Jorge S. Ierache, and María J. Abásolo. “Scalable Integration of Image and Face Based Augmented Reality.” In *Augmented Reality, Virtual Reality, and Computer Graphics*, edited by Lucio Tommaso De Paolis and Patrick Bourdot, 232–42. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020. https://doi.org/10.1007/978-3-030-58465-8_18.

  En esta publicación se comunican los principales elementos de diseño y la arquitectura integrada objeto de este trabajo de tesis.

- Mangiarua, Nahuel, Jorge Ierache, and María José Abasolo. “Implementation of an Open Source Based Augmented Reality Engine for Cloud Authoring Frameworks.” *Journal of Computer Science and Technology* 19, no. 2 (October 2019): e16. https://doi.org/10.24215/16666038.19.e16.

  En esta publicación se comunica el diseño del proceso basado en imágenes y su análisis particular así como consideraciones para el aprovechamiento de capacidades de la nube para herramientas de autor.

- Mangiarua, Nahuel, Jorge Ierache, Martin Becerra, Hernán Maurice, Santiago Igarza, and Osvaldo Spositto. “Templates Framework for the Augmented Catalog System.” In *Computer Science – CACIC 2018*, edited by Patricia Pesado and Claudio Aciti, 267–276. Cham Springer International Publishing: Springer International Publishing, 2019. http://sedici.unlp.edu.ar/handle/10915/73478.

  En esta publicación se comunican los principales elementos de diseño y la implementación del concepto de template para sistemas de autor que facilita el trabajo a usuarios finales frente a un incremento de escala en el sistema.

- Ierache, Jorge, Nahuel Mangiarua, Martín Ezequiel Becerra, and Santiago Igarza. “Framework for the Development of Augmented Reality Applications Applied to Education Games.” In *Proceedings of the 5th International Conference on Augmented Reality, Virtual Reality, and Computer Graphics*, edited by Lucio Tommaso De Paolis and Patrick Bourdot, 340–350. Cham Springer International Publishing: Springer International Publishing, 2018. https://doi.org/10.1007/978-3-319-95270-3_28.

  Esta publicación complementaria comunica un framework especializado desarrollado sobre el Sistema de Catálogos Virtuales Aumentados, el cual constituye un caso de aplicación potencial para la implementación de la arquitectura propuesta.

8.3.2 Congresos internacionales

- Montalvo, C., F. Petrolo, D. Sanz, Nahuel Mangiarua, N. Verdicchio, S. Igarza, and J. Ierache. “Knowledge Based Augmented Card System for Medical Assistance Over Mobile Devices.” In *Selected Papers of the XXI Argentine Congress of Computer Science*, 257–65, 2017.
- Ierache, Jorge, Nicolás Nazareno Verdicchio, N. Duarte, and Nahuel Mangiarua. “Augmented Reality Card System for Emergency Medical Services.” In *Proceedings of the International Work-Conference on Bioinformatics and Biomedical Engineering*, 487–94. Granada, Spain, 2016.
- Ierache, Jorge, Nahuel Mangiarua, N. Verdicchio, and D. Sanz. “Augmented Card System Based on Knowledge for Medical Emergency Assistance.” In *2016 IEEE Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI)*, 1–3, 2016. https://doi.org/10.1109/CACIDI.2016.7785979.
- Ierache, Jorge, Nahuel Mangiarua, S. Bevacqua, N. Verdicchio, and M. Becerra. “Development of a Catalogs System for Augmented Reality Applications.” In *World Academy of Science, Engineering and Technology, International Science Index*, 9:1–7, 2015. http://waset.org/Publications/development-of-a-catalogs-system-for-augmentedreality- applications/10000077.

8.3.3 Congresos nacionales y workshops

- Mangiarua, Nahuel, Jorge Ierache, and María José Abásolo Guerrero. “Avances en Línea de Investigación Doctoral: Integración Escalable de Realidad

Aumentada Basada en Images y Rostros.” In *Libro de actas del XXII Workshop de Investigadores en Ciencias de la Computación*. Santa Cruz, Argentina, 2020.

- Mangiarua, Nahuel, Jorge Ierache, Martín Ezequiel Becerra, Hernán Maurice, Santiago Igarza, and Osvaldo Mario Spositto. “Framework para la generación de templates en sistema de catálogos de RA.” In *Libro de actas del XXIV Congreso Argentino de Ciencias de la Computación*. La Plata, Argentina, 2018. http://sedici.unlp.edu.ar/handle/10915/73478.
- Ierache, Jorge, Javier Urien, Nahuel Mangiarua, Martín Ezequiel Becerra, Paola Pezoimburg, Emidio Bueno, Hernán Maurice, and Eric S. Auchterberge. “Desarrollo de un prototipo de RA de tipo geoespacial.” In *Libro de actas del XII Jornadas de Vinculación Universidad-Industria*, 5–8, 2018. http://sedici.unlp.edu.ar/handle/10915/71289.
- Becerra, Martín Ezequiel, Nahuel Mangiarua, Santiago Igarza, Jorge Ierache, and María José Abásolo Guerrero. “Líneas de investigación del grupo de RA aplicada: templates de catálogos aumentados integración escalable de RA basada en imágenes y rostros aumentación de sistemas SCADA en el contexto de la industria 4.0.” In *Libro de actas del XX Workshop de Investigadores en Ciencias de la Computación*, 2018. http://sedici.unlp.edu.ar/handle/10915/67455.
- Mangiarua, Nahuel, Cristian Montalvo, Facundo Petrolo, Diego Rubén Sanz, Nicolás Nazareno Verdicchio, E. Lobatto, A. Rosenthal, Martín Ezequiel Becerra, Santiago Igarza, and Jorge Ierache. “Framework para la generación de templates en sistemas de catálogos de RA.” In *Libro de actas del XIX Workshop de Investigadores en Ciencias de la Computación*, 2017. http://sedici.unlp.edu.ar/handle/10915/61824.
- Verdicchio, Nicolás Nazareno, Diego Rubén Sanz, Cristian Montalvo, Facundo Petrolo, Nahuel Mangiarua, Santiago Igarza, and Jorge Ierache. “Sistema de catálogo virtual aumentado: integración de framework especializado orientado a juegos didácticos.” In *Libro de actas del XI Congreso de Tecnología en Educación y Educación en Tecnología*, 2016. http://hdl.handle.net/10915/54651.

8.3.4 Trabajos previos

- Bevacqua, Sebastián Ariel, Santiago Igarza, Nahuel Mangiarua, Martín Ezequiel Becerra, Nicolás Nazareno Verdicchio, Fernando Martín Ortiz, Diego Rubén Sanz, Nicolás Daniel Duarte, Matías Ezequiel Sena, and Jorge Ierache. “Líneas de investigación del grupo de Realidad Aumentada Aplicada de UNLaM.” In *Libro de actas del XVII Workshop de Investigadores en Ciencias de la Computación*. Salta, Argentina, 2015. http://sedici.unlp.edu.ar/handle/10915/45657.

- Mangiarua, Nahuel, Jorge Ierache, and Martin Becerra. “Herramienta de Realidad Aumentada Para La Explotación de Material Didáctico Tradicional.” In *Libro de Actas Del IX Congreso de Tecnología En Educación En Tecnología*, 240–54. Chilecito, Argentina, 2014.
- Ierache, Jorge, Santiago Igarza, Nahuel Mangiarua, Martín Ezequiel Becerra, Sebastián Ariel Bevacqua, Nicolás Nazareno Verdicchio, Fernando Martín Ortiz, Diego Rubén Sanz, Nicolás Daniel Duarte, and Esteban de la Llave. “Realidad Aumentada (RA) en el contexto de usuarios finales.” In *Libro de actas del XVI Workshop de Investigadores en Ciencias de la Computación*, 2014. http://sedici.unlp.edu.ar/handle/10915/41253.
- Ierache, Jorge, Nahuel Mangiarua, and Sebastián Ariel Bevacqua. “Sistema de catálogo para la asistencia a la creación, publicación, gestión y explotación de contenidos multimedia y aplicaciones de RA.” In *Libro de actas del XX Congreso Argentino de Ciencias de la Computación*. Buenos Aires, Argentina, 2014. http://sedici.unlp.edu.ar/handle/10915/42339.
- Ierache, Jorge, Nahuel Mangiarua, and Martin Becerra. “Herramienta de Realidad Aumentada Para Facilitar La Enseñanza En Contextos Educativos Mediante El Uso de Las TICs,” Revista Latinoamericana de Ingeniería de Software, 1 (2014): 1–3.

8.4 Futuras líneas de investigación

Si bien el marco de evaluación de algoritmos de ANN para el contexto de la RA propuesto predice con mayor precisión el desempeño de los algoritmos, se requiere continuar sobre esta línea para establecer condiciones aún más específicas y con el fin de desarrollar una métrica de comparación concreta más precisa que la observación directa del recall para los dos primeros vecinos más cercanos.

Entre las futuras líneas de trabajo que se plantea abordar se encuentra la compilación cruzada del prototipo de implementación a plataformas ARM, abriendo la puerta a su aplicación para la explotación en una mayor gama de dispositivos móviles.

Adicionalmente, se considera la integración con sistemas de alto nivel, específicamente el motor de gráficos Unity3D, mediante un wrapper que exponga la funcionalidad existente para su invocación desde el entorno del motor, simplificando la generación de aplicaciones para usuarios finales.

También se plantea la integración continua de algoritmos de inferencia de datos biométricos a medida que sean desarrollados y publicados por sus respectivos autores, haciendo uso de las facilidades de integración ya incluidas en esa sección de la arquitectura a través de una interfaz común.

Finalmente se planea, continuando con la línea de investigación postulada en (J. Ierache, Mangiarua, et al. 2016; J. Ierache, Verdicchio, et al. 2016) y bajo el marco del proyecto PROINCE C-231 Comandos de Voz y Reconocimiento Facial para Aplicaciones de Realidad Aumentada, el desarrollo de un prototipo demostrador en el contexto de la emergentología. Esta contempla como elemento clave la utilización dual y escalable de credenciales (imágenes) y rostros humanos con reconocimiento facial como objetivos de aumentación para la explotación de información relevante a la atención médica de emergencia, como lo pueden ser un listado de alergias o historial de patologías. En el Apéndice B se presenta la propuesta inicial que se desarrollara durante el periodo 2020-2021.

Reconocimientos

Esta tesis se radicó en el Grupo de Realidad Aumentada Aplicada de la Universidad Nacional de La Matanza articulado en el marco de los proyectos:

PROINCE C-168 “Realidad Aumentada en el Contexto de Usuarios Finales Geoposicionados”, 2015-2016, DIIT, UNLAM.

PROINCE C-185 “Realidad Aumentada Basada en Conocimiento en el Contexto de la Medicina Tarjeta de Emergencia Personal Aumentada (TEPA)”, 2016-2017, DIIT, UNLAM

PROINCE C202 “Framework para la Generación de Templates en Sistemas de Catálogos de Realidad Aumentada”, 2017-2018, DIIT, UNLAM

PROINCE C-213 “Framework para la generación de video juegos educativos basado en sistema de catálogos de RA”, 2018-2019, DIIT, UNLAM

Convocatoria Fase Cero (AVT) Fundación Sadosky “Desarrollo de un prototipo de Realidad Aumentada de tipo Geoespacial”, 2017

Adicionalmente, se contó con una beca Formando UNLaM edición 2018 de dicha casa de altos estudios.

Apéndice A
Repositorio digital de código del proyecto

Se encuentra disponible el código fuente del prototipo de implementación de la arquitectura, incluyendo el código particular para los experimentos y documentación al respecto en formato doxygen en las siguientes direcciones de internet:

Repositorio de código (https://gitlab.com/frozensun/Engar) (para quien solicite el acceso con fines académicos, hasta la determinación de una licencia libre adecuada y su subsiguiente publicación).

Documentación del código (http://frozen-sun.net/engar/)

Para comprender el código aquí incluido es necesario poseer conocimientos de programación en el lenguaje C++ y otros aspectos técnicos propios del desarrollo de software.

A continuación se resume la lista de archivos disponibles en el repositorio digital previamente citado:

```
.
├── cmake
│   ├── cotire.cmake
│   ├── FindFLANN.cmake
│   ├── FindIconv.cmake
│   ├── FindOpenCV.cmake.disabled
│   ├── FindTBB.cmake
│   ├── Makefile
│   └── OpenCVConfig.cmake
├── CMakeLists.txt
├── dependencies.sh
├── Doxyfile
├── experiments
│   ├── age
│   │   ├── age.cpp
│   │   └── CMakeLists.txt
│   ├── bench
│   │   ├── bench.cpp
│   │   └── CMakeLists.txt
│   ├── camera_calibration.cpp
│   ├── facemark_tests.cpp
│   ├── features
```

```
│   │   ├── BRISKMatching.cpp
│   │   ├── BRISKMatching.h
│   │   ├── CMakeLists.txt
│   │   ├── FASTMatching.cpp
│   │   ├── FASTMatching.h
│   │   ├── GFTTMatching.cpp
│   │   ├── GFTTMatching.h
│   │   ├── ORBMatching.cpp
│   │   └── ORBMatching.h
│   ├── filter-homography-decomp.cpp
│   ├── flanntest
│   │   ├── CMakeLists.txt
│   │   └── flannTest.cpp
│   ├── gender
│   │   ├── CMakeLists.txt
│   │   └── gender.cpp
│   ├── hammingtest
│   │   ├── CMakeLists.txt
│   │   └── hammingtest.cpp
│   ├── LiveBench
│   │   ├── CMakeLists.txt
│   │   └── LiveBench.cpp
│   ├── ngt
│   │   ├── CMakeLists.txt
│   │   └── ngt.cpp
│   ├── nmslib
│   │   ├── CMakeLists.txt
│   │   └── nmslib.cpp
│   ├── ocltest
│   │   ├── CMakeLists.txt
│   │   └── ocltest.cpp
│   └── SaEngar
│       ├── CMakeLists.txt
│       └── SaEngar.cpp
├── img
│   ├── chips.jpg
│   ├── faces
│   │   ├── jorge.jpg
│   │   ├── martin.jpg
│   │   ├── nahuel.jpg
│   │   └── patricia.jpg
│   ├── IMG_20141023_142816.jpg
│   ├── IMG_20141117_175253.jpg
│   ├── IMG_20141118_163718.jpg
│   ├── IMG_20141118_163731.jpg
│   ├── IMG_20141118_163740.jpg
│   ├── IMG_20141118_163751.jpg
│   ├── IMG_20141118_163805.jpg
│   ├── obj.png
│   ├── paper.jpg
│   ├── paper.png
│   ├── qr
│   │   ├── QRBase.png
│   │   ├── QRExample0.png
│   │   └── qrScene.png
│   ├── scene.png
│   ├── scenep.png
│   ├── scenepr45.png
```

```
│   ├── scener45.png
│   ├── scener45su.png
│   ├── scener90.png
│   ├── stones.jpg
│   ├── targets
│   │   ├── target_1.jpg
│   │   ├── target_2.jpg
│   │   ├── target_3.jpg
│   │   ├── target_4.jpg
│   │   ├── target_5.jpg
│   │   ├── target_6.jpg
│   │   └── target_7.jpg
│   ├── test.png
│   ├── wall.jpg
│   └── wall.png
├── libs
│   ├── libngt.so
│   └── libNonMetricSpaceLib.a
├── main
│   ├── headers
│   │   ├── AkazeDetector.h
│   │   ├── akazeutils.h
│   │   ├── ArbitraryImageEngine.h
│   │   ├── ArDetectionResult.h
│   │   ├── ArDetector.h
│   │   ├── BasicColors.h
│   │   ├── BinBoostDetector.h
│   │   ├── CaptureVideo.h
│   │   ├── CascadeFaceDetector.h
│   │   ├── CvNetFaceDetector.h
│   │   ├── DlibFaceDetector.h
│   │   ├── DlibFaceRecognizer.h
│   │   ├── FaceDetector.h
│   │   ├── FaceEngine.hpp
│   │   ├── FaceRecognizer.h
│   │   ├── FastDetector.h
│   │   ├── FlannBasedMatcherHack.hpp
│   │   ├── HomographyHelper.h
│   │   ├── Identity.h
│   │   ├── ImageBasedEngine.h
│   │   ├── inferrers
│   │   │   ├── AgeInferrer.h
│   │   │   ├── CaffeNetAbstractInferrer.h
│   │   │   ├── FERInferrer.h
│   │   │   ├── GenderInferrer.h
│   │   │   └── Inferrer.h
│   │   ├── IntegratedAREngine.h
│   │   ├── KeypointMatches.h
│   │   ├── LKTracker.h
│   │   ├── matchers
│   │   │   ├── AnnoyMatcher.h
│   │   │   ├── FLANNMatcher.h
│   │   │   ├── KGraphMatcher.h
│   │   │   ├── MRPTMatcher.h
│   │   │   ├── NGTMatcher.h
│   │   │   └── NMSMatcher.h
│   │   ├── OpenCvFrameProvider.h
│   │   ├── OrbDetector.h
```

```
│   │   ├── PointAndIndex.h
│   │   ├── PoolFrameProvider.hpp
│   │   ├── RandomFrameProvider.hpp
│   │   ├── Range.hpp
│   │   ├── StaticFrameProvider.hpp
│   │   ├── ThreadSafeQueue.h
│   │   ├── TrackedFace.h
│   │   ├── Unity3DFrameProvider.h
│   │   ├── Utils.h
│   │   └── VideoFrameProvider.h
│   └── source
│       ├── akazedemo.cpp
│       ├── AkazeDetector.cpp
│       ├── ArbitraryImageEngine.cpp
│       ├── ArDetectionResult.cpp
│       ├── BasicColors.cpp
│       ├── BinBoostDetector.cpp
│       ├── CaptureVideo.cpp
│       ├── CascadeFaceDetector.cpp
│       ├── CvNetFaceDetector.cpp
│       ├── DisplayImage.cpp
│       ├── DlibFaceDetector.cpp
│       ├── DlibFaceRecognizer.cpp
│       ├── FastDetector.cpp
│       ├── HomographyHelper.cpp
│       ├── ImageBasedEngine.cpp
│       ├── inferrers
│       │   └── CaffeNetAbstractInferrer.cpp
│       ├── IntegratedAREngine.cpp
│       ├── KeypointMatches.cpp
│       ├── LKTracker.cpp
│       ├── matchers
│       │   ├── AnnoyMatcher.cpp
│       │   ├── FLANNMatcher.cpp
│       │   ├── KGraphMatcher.cpp
│       │   ├── MRPTMatcher.cpp
│       │   ├── NGTMatcher.cpp
│       │   └── NMSMatcher.cpp
│       ├── OpenCvFrameProvider.cpp
│       ├── OrbDetector.cpp
│       ├── PointAndIndex.cpp
│       ├── TrackedFace.cpp
│       ├── Unity3DFrameProvider.cpp
│       ├── Unity3DPlugin.cpp
│       └── Utils.cpp
├── notes.txt
├── plugins
│   ├── headers
│   │   ├── annoylib.hpp
│   │   ├── binboost
│   │   │   ├── BoostDesc.h
│   │   │   └── Utils.h
│   │   ├── ezh5.hpp
│   │   ├── GLEW
│   │   │   ├── glew.c
│   │   │   ├── glew.h
│   │   │   ├── glxew.h
│   │   │   └── wglew.h
```

```
│   │   ├── kissrandom.h
│   │   ├── minicsv.h
│   │   ├── Mrpt.hpp
│   │   ├── PlatformBase.h
│   │   ├── rapidjson
│   │   │   ├── allocators.h
│   │   │   ├── cursorstreamwrapper.h
│   │   │   ├── document.h
│   │   │   ├── encodedstream.h
│   │   │   ├── encodings.h
│   │   │   ├── error
│   │   │   │   ├── en.h
│   │   │   │   └── error.h
│   │   │   ├── filereadstream.h
│   │   │   ├── filewritestream.h
│   │   │   ├── fwd.h
│   │   │   ├── internal
│   │   │   │   ├── biginteger.h
│   │   │   │   ├── diyfp.h
│   │   │   │   ├── dtoa.h
│   │   │   │   ├── ieee754.h
│   │   │   │   ├── itoa.h
│   │   │   │   ├── meta.h
│   │   │   │   ├── pow10.h
│   │   │   │   ├── regex.h
│   │   │   │   ├── stack.h
│   │   │   │   ├── strfunc.h
│   │   │   │   ├── strtod.h
│   │   │   │   └── swap.h
│   │   │   ├── istreamwrapper.h
│   │   │   ├── memorybuffer.h
│   │   │   ├── memorystream.h
│   │   │   ├── msinttypes
│   │   │   │   ├── inttypes.h
│   │   │   │   └── stdint.h
│   │   │   ├── ostreamwrapper.h
│   │   │   ├── pointer.h
│   │   │   ├── prettywriter.h
│   │   │   ├── rapidjson.h
│   │   │   ├── reader.h
│   │   │   ├── schema.h
│   │   │   ├── stream.h
│   │   │   ├── stringbuffer.h
│   │   │   └── writer.h
│   │   ├── RenderAPI.h
│   │   └── Unity
│   │       ├── IUnityGraphicsD3D11.h
│   │       ├── IUnityGraphicsD3D12.h
│   │       ├── IUnityGraphicsD3D9.h
│   │       ├── IUnityGraphics.h
│   │       ├── IUnityGraphicsMetal.h
│   │       └── IUnityInterface.h
│   └── source
│       ├── bigint
│       │   ├── BigIntegerAlgorithms.cc
│       │   ├── BigIntegerAlgorithms.hh
│       │   ├── BigInteger.cc
│       │   ├── BigInteger.hh
```

```
│       │   ├── BigIntegerLibrary.hh
│       │   ├── BigIntegerUtils.cc
│       │   ├── BigIntegerUtils.hh
│       │   ├── BigUnsigned.cc
│       │   ├── BigUnsigned.hh
│       │   ├── BigUnsignedInABase.cc
│       │   ├── BigUnsignedInABase.hh
│       │   ├── ChangeLog
│       │   ├── Makefile
│       │   ├── NumberlikeArray.hh
│       │   └── README
│       └── binboost
│           ├── BoostDesc.cpp
│           ├── main.cpp
│           └── Utils.cpp
├── results.txt
├── r.txt
├── runbench.sh
└── structure.txt
```

Apéndice B
Código de generación de los sets de datos de entrenamiento y de consultas de algoritmos ANN

El presente código se encarga de leer los archivos de los sets de datos SIFT1M y SIFT10M para luego aplicarles una distorsión del 5% o del 15% de acuerdo a lo presentado en el capítulo 5. Si bien este código es parte del repositorio digital, se lo incluye para ahorrarle al lector la necesidad de buscarlo.

Adicionalmente se incluye las direcciones de acceso a los principales juegos de datos base mencionados que se encuentran disponibles bajo sus respectivos autores:

SIFT1M (http://corpus-texmex.irisa.fr/) (Jégou, Douze, y Schmid 2011)

SIFT10M (https://archive.ics.uci.edu/ml/datasets/SIFT10M) (Dua y Graff 2017)

```
template<typename T>
vector<vector<T>> read_record(string fileName, unsigned long maxRows,
unsigned long offset) {

    // File pointer
    fstream fin;

    // Open an existing file
    fin.open(fileName, ios::in);

    // Read the Data from the file
    // as String Vector
    string line, word;

    vector<vector<T>> result;
    unsigned long rowCount = 0;
    unsigned long lastRowLength = 0;
    unsigned long offCount = 0;
    while (getline(fin, line) && rowCount < maxRows) {
        offCount++;
        if (offCount < offset) continue;
        rowCount++;

        vector<T> &row = result.emplace_back();
        row.reserve(lastRowLength);
```

```
        lastRowLength = 0;

        // used for breaking words
        stringstream s(line);

        while (!s.eof()) {
            T temp;
            s >> temp;
            row.emplace_back(temp);
            s.ignore();
            lastRowLength++;
        }
    }

    fin.close();

    return result;
}

template<>
vector<vector<unsigned char>> read_record(string fileName, unsigned
long maxRows, unsigned long offset) {

    // File pointer
    fstream fin;

    // Open an existing file
    fin.open(fileName, ios::in);

    // Read the Data from the file
    // as String Vector
    string line, word;
    unsigned short elemCount;
    vector<vector<unsigned char>> result;
    unsigned long rowCount = 0;
    unsigned long lastRowLength = 0;
    unsigned long offCount = 0;
    while (getline(fin, line) && rowCount < maxRows) {
        offCount++;
        if (offCount < offset) continue;
        rowCount++;

        vector<unsigned char> &row = result.emplace_back();
        row.reserve(lastRowLength);
        lastRowLength = 0;

        // used for breaking words
        stringstream s(line);

        elemCount = 0;
        while (!s.eof() && elemCount < BYTE_COUNT) {
            elemCount++;
            float temp;
            s >> temp;
            unsigned char val = static_cast<unsigned char>(temp);
            row.emplace_back(val);
            s.ignore();
```

```
            lastRowLength++;
        }
    }

    fin.close();

    return result;
}

    typedef unsigned char dType;
    #define BYTE_COUNT 32

int main(int argc, char **argv) {
    int samplesAmount = 5000;
    int sampleSize = 200;
    int searchSize = 50;
    int k = 2;
    float change = (is_same<float, dType>::value) ? 0.05 : 0.15;
    int changeType = 0;

    // ISO 8601: %Y-%m-%d %H:%M:%S, e.g. 2017-07-31 00:42:00+0200.
    auto time = std::time(nullptr);
    std::cout << std::put_time(std::gmtime(&time), "%F %T%z") << endl;

    cout << "Samples Amount: " << samplesAmount << endl;
    cout << "Sample Size: " << sampleSize << endl;
    cout << "Search Size: " << searchSize << endl;
    cout << "k: " << k << endl;
    cout << "change: " << change << endl;
    cout << "change type " << changeType << endl;
    std::string dist = (is_same<float, dType>::value) ? "L2" :
"Hamming";
    cout << "distance: " << dist << endl;


    vector<cv::Mat> trainSamples;
    // SIFT10M_extract_
    vector<vector<dType>> train =
read_record<dType>("../../../../../data/train.csv",
samplesAmount * sampleSize, 0);
    toMatVector(samplesAmount, sampleSize, train, trainSamples);

    std::random_device dev;
    std::mt19937 rng(dev());
    std::uniform_real_distribution<> random(0, 1); // distribution in
range [1, 6]
    std::uniform_int_distribution<> sign(0, 1); // distribution in
range [1, 6]

    float c = change * 255;
    vector<cv::Mat> testSamples;
    for (int q = 0; q < searchSize; q++) {
        auto &s = trainSamples[q];
        cv::Mat sample;
        s.copyTo(sample);
        for (int i = 0; i < sample.rows; i++) {
```

```
            if (is_same<float, dType>::value) {
                auto row = sample.ptr<float>(i);
                for (int j = 0; j < sample.cols; j++) {

                    if (changeType == 0) {
                        // All dimensions change a bit
                        float m = (sign(rng) ? 1 : -1) * c;
                        row[j] = max(min(row[j] + m, (float) 255.0),
(float) 0.0);
                    } else {
                        // Only "change %" dimension change randomly
                        if (random(rng) < change) {
                            row[j] = random(rng) * 255;
                        }
                    }
                }
            } else {
                auto row = sample.ptr<unsigned char>(i);
                for (int j = 0; j < sample.cols; j++) {
                    std::bitset<8> bits(row[j]);

                    for (int b = 0; b < 8; b++) {
                        if (random(rng) < change) {
                            bits.flip(b);
                        }
                    }

                    row[j] = static_cast<unsigned
char>(bits.to_ulong());
                }
            }
        }
        testSamples.emplace_back(sample);
    }

// Set de datos listo para pruebas

}
```

Código 1. Distorsión de elementos de data set de descriptores SIFT.

Apéndice C
Código de generación de flujo de vídeo simulado para prueba de detección

El siguiente código corresponde a la clase PoolVideoFrameProvider encargada de generar el flujo de vídeo simulado utilizado para la prueba de detección en la secciones 5.6. Nuevamente aunque este código es parte del repositorio digital, se lo incluye para ahorrarle al lector la necesidad de buscarlo.

Adicionalmente se incluye la dirección de acceso al juego de imágenes base utilizados que se encuentran disponibles bajo sus respectivos autores:

Obras de arte (https://www.kaggle.com/c/ painter-by-numbers/overview) («Kaggle» 2020)

```
#ifndef ENGAR_POOLFRAMEPROVIDER_H
#define ENGAR_POOLFRAMEPROVIDER_H

#include <opencv2/videoio.hpp>
#include "VideoFrameProvider.h"

#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
#include <random>
#include <opencv2/imgproc.hpp>

namespace engar {

    class PoolFrameProvider : public VideoFrameProvider {

    private:
        int maxTargetId = 0;
        int framesPerTarget = 10;

        int start;
        int end;

        const std::vector<std::string> &targetNames;

        cv::Size size;
```

```
        int angleCount = 0;
        int targetId = 0;

        int rotationSpeed;

        cv::Mat affineRotation;

        cv::Mat wall;
        cv::Mat paper;

    public:

        int currentTargetId = -1;

        void Reset() override {
            targetId = start - 1;
            LoadNextTarget();
            currentTargetId = targetId;
        }

        explicit PoolFrameProvider(const std::vector<std::string>
&targetNames, int framesPerTarget, int startIdx, int endIdx,
                                   const cv::Size &size =
cv::Size(640, 480),
                                   bool rotate = false, int
rotationSpeed = 1, bool blackInterpolation = false) :
targetNames(targetNames) {
            this->size = size;
            maxTargetId = targetNames.size();
            start = startIdx;
            end = endIdx;
            this->rotationSpeed = rotationSpeed;
            this->framesPerTarget = framesPerTarget;


            this->rotate = rotate;
            this->blackInterpolation = blackInterpolation;

            wall = imread("./img/wall.jpg", cv::IMREAD_COLOR);
            paper = imread("./img/paper.jpg", cv::IMREAD_COLOR);

            targetId = startIdx - 1;
            LoadNextTarget();
            currentTargetId = targetId;

            cv::Point2f pts1[] = {cv::Point2f(56, 65),
cv::Point2f(368, 52), cv::Point2f(28, 387), cv::Point2f(389, 390)};
            cv::Point2f pts2[] = {cv::Point2f(0, 0), cv::Point2f(300,
0), cv::Point2f(0, 300), cv::Point2f(300, 300)};

            affineRotation = getPerspectiveTransform(pts1, pts2);
        }

        ~PoolFrameProvider() {
            this->frame.release();
            this->uFrame.release();
```

```
        }

        void LoadNextTarget() {
            targetId++;
            currentTargetId = targetId;

            paper.copyTo(this->frame);

            if (currentTargetId > end)
                return;

            cv::Mat temp = imread(targetNames[targetId],
cv::IMREAD_COLOR);
            temp.copyTo(frame(cv::Rect(frame.cols / 2 - temp.cols / 2,
frame.rows / 2 - temp.rows / 2, temp.cols, temp.rows)));

            this->uFrame = frame.getUMat(cv::ACCESS_READ,
cv::USAGE_ALLOCATE_HOST_MEMORY);
            angleCount = 0;
        }

        void GrabFrame(cv::UMat &out) override {
            if (frameCount % framesPerTarget == 0) {
                LoadNextTarget();
            }

            currentTargetId = targetId;

            if (blackInterpolation && (frameCount % 100 == 0 ||
frameCount % 99 == 0 || frameCount % 98 == 0 || frameCount % 97 == 0))
{
                out = cv::UMat(this->uFrame.size(), this-
>uFrame.type());
                cv::randn(out, mean, sigma);
                currentTargetId = -1;
            } else {
                if (rotate) {
                    ulong angle = angleCount % 360;
                    cv::Mat matRotation =
getRotationMatrix2D(cv::Point(frame.cols / 2, frame.rows / 2), angle,
1);
                    cv::Mat temp;
                    warpAffine(frame, temp, matRotation,
frame.size());
                    warpPerspective(temp, temp, affineRotation,
frame.size());

                    cv::Mat mask(frame.size(), CV_8U, cv::Scalar(1));
                    warpAffine(mask, mask, matRotation, mask.size());
                    warpPerspective(mask, mask, affineRotation,
mask.size());

                    wall.copyTo(out);
                    temp.copyTo(out, mask);

                    GaussianBlur(out, out, cv::Size(5, 5), 0.1, 0.1);
```

```
                } else {
                    uFrame.copyTo(out);
                }
            }

            frameCount++;
            angleCount += rotationSpeed;
        }

        void GrabFrame(cv::Mat &out) override {
            if (frameCount % framesPerTarget == 0) {
                LoadNextTarget();
            }

            currentTargetId = targetId;

            if (blackInterpolation && (frameCount % 50 == 0 ||
frameCount % 49 == 0 || frameCount % 51 == 0)) {
                out = cv::Mat(this->uFrame.size(), this-
>uFrame.type());
                cv::randn(out, mean, sigma);
                currentTargetId = -1;
            } else {
                if (rotate) {
                    ulong angle = angleCount % 360;
                    cv::Mat matRotation =
getRotationMatrix2D(cv::Point(frame.cols / 2, frame.rows / 2), angle,
1);

                    cv::Mat temp;
                    warpAffine(uFrame, temp, matRotation,
uFrame.size());
                    warpPerspective(temp, temp, affineRotation,
temp.size());

                    cv::Mat mask(frame.size(), CV_8U, cv::Scalar(1));
                    warpAffine(mask, mask, matRotation, mask.size());
                    warpPerspective(mask, mask, affineRotation,
mask.size());

                    wall.copyTo(out);
                    temp.copyTo(out, mask);

                    GaussianBlur(out, out, cv::Size(5, 5), 0.1, 0.1);
                } else {
                    uFrame.copyTo(out);
                }
            }
            angleCount += rotationSpeed;
            frameCount++;
        }

    private:
        cv::UMat uFrame;
        cv::Mat frame;
        ulong frameCount = 45;
        bool rotate;
```

```
        bool blackInterpolation;

        cv::Mat mean = cv::Mat::zeros(1, 1, CV_64FC1);
        cv::Mat sigma = cv::Mat::ones(1, 1, CV_64FC1);
    };


}
#endif //ENGAR_POOLFRAMEPROVIDER_H
```

Código 2. Generación de flujo de vídeo simulado.

Apéndice D
Código para medición de desempeño de búsqueda de correspondencias con distintos algoritmos de ANN

El siguiente código corresponde al archivo bench.cpp del repositorio digital utilizado para medir la desempeño y calidad de distintos algoritmos de ANN bajo los distintos tipos de set de datos de prueba generados.

```
class BenchResults {
public:
    BenchResults(string name, int searchSize, int sampleSize, int k) {
        this->k = k;
        this->name = name;
        this->searchSize = searchSize;
        this->sampleSize = sampleSize;
        this->recallCounts.resize(k);
        this->errorRelativeDistanceSums.resize(k);
        this->errorDistanceSums.resize(k);
        this->distanceSums.resize(k);
    }

    string name;
    chrono::microseconds trainTime =
std::chrono::microseconds::zero();
    chrono::microseconds knnTime = std::chrono::microseconds::zero();
    int searchSize = 0;
    int sampleSize = 0;
    vector<int> recallCounts;
    int recall2Counts = 0;
    vector<float> errorRelativeDistanceSums;
    vector<float> errorDistanceSums;
    vector<float> distanceSums;
    int errorOffsetSum = 0;
    int offsetSum = 0;
    int k;

    void ResetTestResults() {
        this->knnTime = std::chrono::microseconds::zero();

        this->recallCounts.clear();
        this->recallCounts.resize(k);
```

```
        this->errorRelativeDistanceSums.clear();
        this->errorRelativeDistanceSums.resize(k);

        this->errorDistanceSums.clear();
        this->errorDistanceSums.resize(k);

        this->distanceSums.clear();
        this->distanceSums.resize(k);

        this->errorOffsetSum = 0;
        this->offsetSum = 0;
        this->recall2Counts = 0;
    }
};

ostream &operator<<(ostream &os, const BenchResults &dt) {
    os << endl;
    os << fixed << std::setprecision(3);
    os << dt.name << ": " << endl;
    os << "  Total Time: " <<
chrono::duration_cast<chrono::seconds>(dt.trainTime +
dt.knnTime).count() << "s" << endl;
    os << "  Total knn Time: " <<
chrono::duration_cast<chrono::milliseconds>(dt.knnTime).count() <<
"ms" << endl;
    os << "  Train Time: " <<
chrono::duration_cast<chrono::seconds>(dt.trainTime).count() << "s" <<
endl;
    os << "  Avg Time per Sample: " <<
chrono::duration_cast<chrono::milliseconds>(dt.knnTime).count() /
(float) dt.searchSize << "ms" << endl;
    os << "  Avg Time per Query: " <<
chrono::duration_cast<chrono::milliseconds>(dt.knnTime).count() /
(float) (dt.searchSize * dt.sampleSize) << "ms" << endl;

    os << "          Recall\t";
    for (int i = 0; i < dt.recallCounts.size(); i++) {
        auto const &rec = dt.recallCounts[i];
        os << rec / (float) (dt.searchSize * dt.sampleSize) << "\t";
    }
    os << endl;
    os << "          Recall 2NN\t";
    os << dt.recall2Counts / (float) (dt.searchSize * dt.sampleSize)
<< "\t";
    os << endl;
    os << "  Avg distance \t";
    for (int i = 0; i < dt.distanceSums.size(); i++) {
        auto const &rec = dt.distanceSums[i];
        os << rec / (float) (dt.searchSize * dt.sampleSize) << "\t";
    }
    os << endl;
    os << "  Error distance %\t";
    for (int i = 0; i < dt.errorRelativeDistanceSums.size(); i++) {
        auto const &rec = dt.errorRelativeDistanceSums[i];
        os << rec / (float) ((dt.searchSize * dt.sampleSize) -
dt.recallCounts[i]) << "\t";
    }
```

```
    os << endl;
    os << "  Error distance \t";
    for (int i = 0; i < dt.errorDistanceSums.size(); i++) {
        auto const &rec = dt.errorDistanceSums[i];
        os << rec / (float) ((dt.searchSize * dt.sampleSize) -
dt.recallCounts[i]) << "\t";
    }
    os << endl;
    os << "  Avg Error Index Offset: " << dt.errorOffsetSum / (float)
((dt.searchSize * dt.sampleSize) - dt.recallCounts[1]);
    os << endl;
    os << "  Avg Index Offset: " << dt.offsetSum / (float)
(dt.searchSize * dt.sampleSize);
    os << endl << endl;
    return os;
}

void testMatcher(vector<cv::Mat> &testSamples, int searchSize,
cv::DescriptorMatcher *matcher, BenchResults &r, int k,
vector<vector<int>> &golds,
                 vector<vector<float>> &dists) {
    omp_set_num_threads(1);
    for (int i = 0; i < searchSize; i++) {
        vector<vector<cv::DMatch> > matches;
        int sampleSize = 200;// testSamples[i].rows;
        matches.resize(sampleSize);

        auto t2 = std::chrono::high_resolution_clock::now();
        matcher->knnMatch(testSamples[i], matches, k);
        r.knnTime +=
chrono::duration_cast<chrono::microseconds>(std::chrono::high_resoluti
on_clock::now() - t2);

        for (const auto &match : matches) {
            bool st = false;
            bool nd = false;
            for (int nn = 0; nn < match.size(); nn++) {
                auto &m = match[nn];

                // Get the correct nn
                int absoluteQueryIdx = m.queryIdx + i * sampleSize;
                auto &gold = golds[absoluteQueryIdx];

                // Compute the found nn in terms on absolute indexing
(trainIdx is in a per img indexing)
                int found = m.trainIdx + m.imgIdx * sampleSize;


                if (gold[nn] == found) {
                    // If nn is correct, increase the recall count
                    r.recallCounts[nn] += 1;

                    if (nn == 0) st = true;
                    if (nn == 1) nd = true;
                } else {
                    // If we made a mistake, compute some statistics
                    if (nn <= 1) {
```

```
                        // Compute index distance to the correct nn
                        for (int j = nn + 1; j < gold.size(); j++) {
                            if (gold[j] == found) break;
                            r.errorOffsetSum++;
                        }
                    }
                    float d = dists[absoluteQueryIdx][nn];
                    r.errorRelativeDistanceSums[nn] += abs((m.distance
- d) / d);
                    r.errorDistanceSums[nn] += m.distance;

                    if (nn <= 1) {
                        // Compute index distance to the correct nn
                        for (int j = nn + 1; j < gold.size(); j++) {
                            r.offsetSum++;
                            if (gold[j] == found) break;
                        }
                    }
                }
                r.distanceSums[nn] += m.distance;


            }
            r.recall2Counts += (st && nd) ? 1 : 0;
        }
    }

    omp_set_num_threads(12);
}
```

Código 3. Medición de desempeño de algoritmos ANN.

El presente código proporciona una salida en forma de texto como la que se ejemplifica a continuación.

```
SamplesAmount SampleSize SearchSize k
Samples Amount: 5000
Sample Size: 200
Search Size: 50
k: 10
change: 0.05
change type 0
Golds computed in: 33s
      Avg query time: 3.351ms
```

```
      Avg distances:    126.145     199.598     224.286     233.228
      237.829     240.632     242.686     244.356     245.739
      246.942     247.995     248.954     249.816     250.602
      251.335     252.016     252.654     253.249     253.808
      254.342
FLANN:
  Total Time: 339s
  Total knn Time: 963ms
  Train Time: 338s
  Avg Time per Sample: 19.260ms
  Avg Time per Query: 0.096ms
          Recall  0.891 0.534 0.287 0.157 0.089 0.051 0.031 0.021
      0.014 0.010
  Avg distance    136.675     212.892     236.172     244.677
      249.442     252.810     255.483     257.720     259.731
      261.510
  Error distance %      0.759 0.152 0.077 0.059 0.053 0.053 0.054
      0.056 0.057 0.059
  Error distance  222.189     241.775     246.637     250.066
      252.790     254.951     256.980     258.804     260.487
      262.118
  Avg Error Index Offset: 5.030
  Avg Index Offset: 2.345



FLANN:
  Total Time: 338s
  Total knn Time: 563ms
  Train Time: 338s
  Avg Time per Sample: 11.260ms
  Avg Time per Query: 0.056ms
          Recall  0.834 0.424 0.188 0.084 0.038 0.019 0.011 0.006
      0.004 0.002
  Avg distance    142.401     218.729     241.130     249.824
      254.846     258.638     261.688     264.290     266.687
      268.872
  Error distance %      0.773 0.178 0.096 0.079 0.075 0.076 0.079
      0.082 0.086 0.089
  Error distance  224.116     243.608     248.811     253.029
      256.440     259.506     262.283     264.641     266.899
      269.004
  Avg Error Index Offset: 7.203
  Avg Index Offset: 4.146


hnsw:
  Total Time: 793s
  Total knn Time: 1345ms
  Train Time: 792s
  Avg Time per Sample: 26.900ms
  Avg Time per Query: 0.134ms
          Recall  0.996 0.987 0.968 0.941 0.910 0.875 0.840 0.801
      0.759 0.714
  Avg distance    126.703     199.819     224.457     233.423
      238.059     240.906     243.006     244.692     246.132
      247.368
```

```
  Error distance %      1.050 0.085 0.021 0.013 0.010 0.008 0.008
      0.006 0.006 0.006
  Error distance  265.168     268.021     267.423     267.494
      267.640     267.286     267.344     267.420     267.610
      267.201
  Avg Error Index Offset: 1.797
  Avg Index Offset: 0.023



hnsw:
  Total Time: 793s
  Total knn Time: 809ms
  Train Time: 792s
  Avg Time per Sample: 16.180ms
  Avg Time per Query: 0.081ms
          Recall  0.983 0.963 0.927 0.877 0.819 0.761 0.703 0.641
      0.586 0.525
  Avg distance    128.510     200.452     224.821     233.773
      238.436     241.324     243.471     245.203     246.699
      247.980
  Error distance %      1.045 0.121 0.031 0.017 0.013 0.011 0.010
      0.009 0.009 0.008
  Error distance  264.356     263.403     262.825     262.809
      262.480     262.109     262.076     261.511     261.811
      261.559
  Avg Error Index Offset: 2.992
  Avg Index Offset: 0.109


Processed 100000 objects. time= 1.67316 (sec)
Processed 200000 objects. time= 2.00848 (sec)
Processed 300000 objects. time= 2.00606 (sec)
Processed 400000 objects. time= 2.20206 (sec)
Processed 500000 objects. time= 2.23169 (sec)
Processed 600000 objects. time= 2.32772 (sec)
Processed 700000 objects. time= 2.31173 (sec)
Processed 800000 objects. time= 2.41011 (sec)
Processed 900000 objects. time= 2.41304 (sec)
Processed 1000000 objects. time= 2.49291 (sec)

ANNG:
  Total Time: 26s
  Total knn Time: 1897ms
  Train Time: 24s
  Avg Time per Sample: 37.940ms
  Avg Time per Query: 0.190ms
          Recall  0.884 0.821 0.715 0.607 0.505 0.420 0.352 0.294
      0.252 0.211
  Avg distance    140.180     204.594     227.282     235.921
      240.501     243.395     245.588     247.360     248.871
      250.198
  Error distance %      0.951 0.153 0.047 0.028 0.021 0.019 0.017
      0.016 0.016 0.016
  Error distance  248.188     255.410     257.437     258.218
      258.578     258.785     259.052     259.400     259.548
      259.902
```

```
  Avg Error Index Offset: 2.744
  Avg Index Offset: 0.493


ANNG:
  Total Time: 26s
  Total knn Time: 1378ms
  Train Time: 24s
  Avg Time per Sample: 27.560ms
  Avg Time per Query: 0.138ms
          Recall  0.828 0.754 0.633 0.516 0.412 0.330 0.269 0.217
      0.179 0.148
  Avg distance    147.093     207.475     228.932     237.282
      241.778     244.710     246.918     248.726     250.279
      251.633
  Error distance %      0.952 0.175 0.056 0.034 0.027 0.024 0.022
      0.021 0.021 0.021
  Error distance  248.842     255.190     256.757     257.325
      257.570     257.833     258.145     258.494     258.766
      259.204
  Avg Error Index Offset: 3.711
  Avg Index Offset: 0.913


MRPT:
  Total Time: 2154s
  Total knn Time: 57629ms
  Train Time: 2096s
  Avg Time per Sample: 1152.580ms
  Avg Time per Query: 5.763ms
          Recall  0.891 0.328
  Avg distance    135.452     225.065
  Error distance %      0.670 0.194
  Error distance  210.165     249.762
  Avg Error Index Offset: 14.624
  Avg Index Offset: 9.826


MRPT:
  Total Time: 2153s
  Total knn Time: 57542ms
  Train Time: 2096s
  Avg Time per Sample: 1150.840ms
  Avg Time per Query: 5.754ms
          Recall  0.891 0.328
  Avg distance    135.452     225.065
  Error distance %      0.670 0.194
  Error distance  210.165     249.762
  Avg Error Index Offset: 14.623
  Avg Index Offset: 9.825


Generating control...
knn Search Avg time: 0.000 ms
Initializing...
iteration: 1 recall: 0 accuracy: 1.1687 cost: 0.00038 M: 10 delta: 1
time: 18.0653 one-recall: 0 one-ratio: 1.97849
```

```
iteration: 2 recall: 0 accuracy: 0.909061 cost: 0.000637268 M: 10
delta: 0.885527 time: 26.1363 one-recall: 0 one-ratio: 1.74749
iteration: 3 recall: 0.04 accuracy: 0.717122 cost: 0.000986742 M: 10
delta: 0.838639 time: 36.8087 one-recall: 0.04 one-ratio: 1.57247
iteration: 4 recall: 0.2 accuracy: 0.455791 cost: 0.00137568 M: 10
delta: 0.788213 time: 48.5366 one-recall: 0.2 one-ratio: 1.3831
iteration: 5 recall: 0.55 accuracy: 0.192105 cost: 0.00176293 M: 10
delta: 0.653629 time: 60.2169 one-recall: 0.55 one-ratio: 1.16801
iteration: 6 recall: 0.76 accuracy: 0.0799293 cost: 0.00207797 M: 10
delta: 0.379211 time: 69.892 one-recall: 0.76 one-ratio: 1.08417
iteration: 7 recall: 0.87 accuracy: 0.0402454 cost: 0.00226166 M: 10
delta: 0.153157 time: 75.9166 one-recall: 0.87 one-ratio: 1.04751
Graph completion with reverse edges...

0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
**************************************************
Reranking edges...

0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
**************************************************

KGraph:
  Total Time: 82s
  Total knn Time: 4601ms
  Train Time: 78s
  Avg Time per Sample: 92.020ms
  Avg Time per Query: 0.460ms
          Recall  0.993 0.924 0.850 0.779 0.720 0.664 0.608 0.560
      0.513 0.474
  Avg distance    127.137     200.950     225.619     234.344
      238.912     241.763     243.898     245.634     247.091
      248.365
  Error distance %      1.034 0.083 0.037 0.019 0.015 0.013 0.012
      0.011 0.010 0.010
  Error distance  263.929     262.683     264.905     265.658
      265.747     266.241     266.143     266.469     266.568
      266.424
  Avg Error Index Offset: 1.934
  Avg Index Offset: 0.146


KGraph:
  Total Time: 81s
  Total knn Time: 3256ms
  Train Time: 78s
  Avg Time per Sample: 65.120ms
  Avg Time per Query: 0.326ms
          Recall  0.977 0.892 0.801 0.720 0.649 0.581 0.520 0.466
      0.419 0.374
  Avg distance    129.671     202.410     226.749     235.406
      239.970     242.848     244.980     246.740     248.232
      249.551
  Error distance %      1.186 0.125 0.053 0.032 0.024 0.021 0.019
      0.017 0.017 0.016
```

```
  Error distance  280.886     267.081     266.218     265.744
      265.294     265.067     264.454     264.343     264.370
      264.210
  Avg Error Index Offset: 3.268
  Avg Index Offset: 0.353


ANNOY:
  Total Time: 375s
  Total knn Time: 6024ms
  Train Time: 369s
  Avg Time per Sample: 120.480ms
  Avg Time per Query: 0.602ms
          Recall  0.974 0.523 0.268 0.146 0.082 0.048 0.032 0.022
      0.016 0.013
  Avg distance    128.387     212.917     237.494     246.209
      251.042     254.409     257.082     259.302     261.256
      262.991
  Error distance %      0.666 0.142 0.081 0.065 0.060 0.059 0.060
      0.061 0.063 0.065
  Error distance  211.495     244.382     249.098     252.297
      254.616     256.740     258.794     260.564     262.251
      263.778
  Avg Error Index Offset: 5.656
  Avg Index Offset: 2.699


ANNOY:
  Total Time: 374s
  Total knn Time: 4974ms
  Train Time: 369s
  Avg Time per Sample: 99.480ms
  Avg Time per Query: 0.497ms
          Recall  0.921 0.390 0.164 0.075 0.037 0.019 0.012 0.007
      0.005 0.004
  Avg distance    133.825     220.922     244.207     252.694
      257.691     261.265     264.162     266.664     268.881
      270.918
  Error distance %      0.760 0.183 0.108 0.090 0.086 0.086 0.089
      0.091 0.094 0.096
  Error distance  223.452     246.382     252.568     256.560
      259.667     262.441     264.942     267.125     269.243
      271.181
  Avg Error Index Offset: 8.318
  Avg Index Offset: 5.071


Process finished with exit code 0
```

Apéndice E
Vídeos demostrativos

Se encuentra disponible una serie de breves elementos multimedia generados a partir del prototipo de implementación de la arquitectura integrada que demuestran las capacidades de aumentación simultanea de imágenes y rostros así como la incorporación de información obtenida mediante la inferencia biométrica a partir del rostro.

Vídeo 1: http://frozen-sun.net/engar/video1.html

Breve ejemplo del flujo de video simulado construida en en apéndice C.

Vídeo 2: http://frozen-sun.net/engar/video2.html

Breve demostración de la capacidad de detección y reconocimiento facial múltiple con inferencia biométrica.

Vídeo 3: http://frozen-sun.net/engar/video3.html

Breve demostración de la capacidad simultanea de aumentación de imágenes y rostros.

Apéndice F
Caso de aplicación en el contexto de la atención médica de emergencia

La propuesta de la línea de trabajo se orienta al desarrollo de un demostrador tecnológico que integre la utilización de comandos de voz con la arquitectura escalable producto de la presente tesis sobre el Sistema de Catálogos Virtuales Aumentados (J. Ierache et al. 2015), base del sistema de Tarjetas de Emergencia Personal Aumentadas (TEPA) (J. Ierache, Mangiarua, et al. 2016), desarrollado en el marco del proyecto PROINCE C-185. En la figura F.1 se observa la estructura general del demostrador a nivel conceptual.

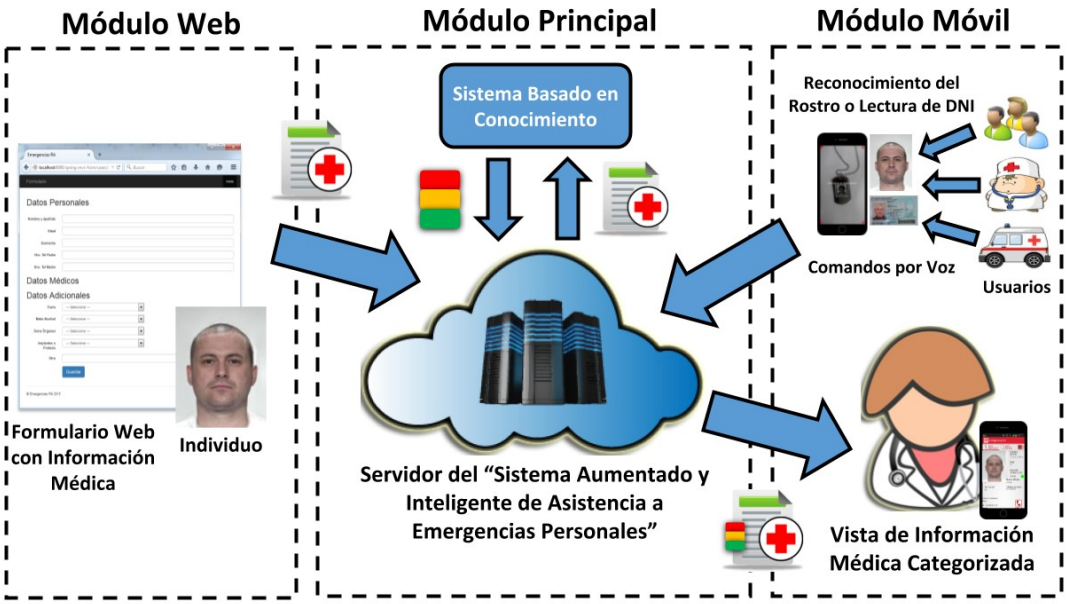

Figura F.1. Diagrama conceptual del caso de aplicación de la arquitectura integradora y un framework de comandos por voz sobre el sistema TEPA.

Con el fin de demostrar el incremento en el abanico de posibilidades de operación y la simplificación de uso que la integración de estas tecnologías otorga se propone la incorporación de dicha arquitectura y demostrador como instancia de prueba sobre el sistema TEPA según se observa en la figura F.2.

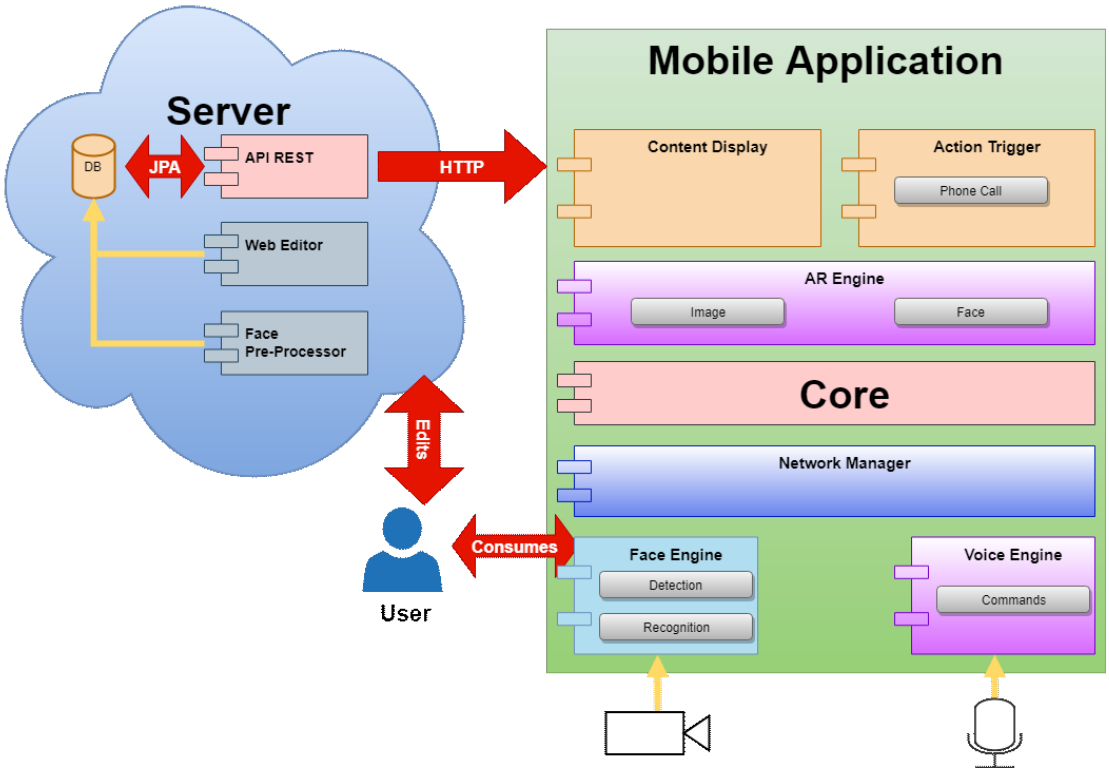

Figura F.2. Diagrama conceptual de la arquitectura propuesta haciendo uso de la arquitectura integrada producto de la presente tesis con la incorporación de comandos de voz.

A su vez, una interfaz de comandos por voz permite al usuario interactuar con el sistema sin la necesidad de ocupar sus manos, liberándolas para la práctica de otras actividades. El reconocimiento de rostros otorga una nueva capa de robustez a los sistemas permitiéndoles potencialmente reconocer usuarios autorizados, inferir información emocional, étnica o etaria, etc., permitiéndoles reaccionar y adecuarse en consecuencia.

Considerando los objetivos primarios detallados en la tabla B.1 de reconocer credenciales (imágenes), reconocer rostros con humanos en el loop y aceptar seis comandos de voz básicos; la arquitectura de integración escalable basada en imágenes arbitrarias y rostros con reconocimiento facial e inferencia biométrica permitiría cumplimentar con los dos primeros. A la vez, la misma asegura un potencial de escalabilidad suficiente para permitir la explotación en entornos específicos como instituciones educativas y comerciales o incluso pequeñas comunidades.

Reconocimiento de credenciales:	
	• Detectar y reconocer una credencial (imagen)
Reconocimiento de rostros con humanos en el loop:	
	• Detectar rostros.
	• Reconocer a los 4 titulares de una TEPA más parecidos al individuo enfocado con la cámara
	• Ofrecer datos disponibles de los 4 titulares para seleccionar manualmente por voz al correcto
Comandos básicos por voz:	
	• Factor: Comando para mostrar el factor sanguíneo del titular de la TEPA.
	• Contacto: Comando para mostrar los contactos del titular de la TEPA.
	• Medicamentos: Comando para mostrar los medicamentos que toma el titular de la TEPA.
	• Alergias: Comando para mostrar las alergias del titular de la TEPA.
	• Historial: Comando para mostrar el historial clínico del titular de la TEPA.
	• Seleccionar (1,2,3,4): Comando para seleccionar al individuo correcto de entre los 4 más parecidos

Tabla F.1. Detalle de objetivos propuestos para la línea de trabajo.

Referencias bibliográficas

Abásolo Guerrero, María José, Cristina Manresa Yee, Ramón Más Sansó, y Marcelo Vénere. 2011. *Realidad virtual y realidad aumentada*. Editorial de la Universidad Nacional de La Plata (EDULP). http://sedici.unlp.edu.ar/handle/10915/18399.

«Affectiva». 2020. Affectiva. 2020. https://www.affectiva.com/.

Agrawal, M., K. Konolige, y M. Rufus Blas. 2008. «Censure: Center surround extremas for real-time feature detection and matching». En *ECCV (4), volume 5305 of Lecture Notes in Computer Science*, editado por David A. Forsyth, Philip H. S. Torr, y Andrew Zisserman, 102-15. Springer.

«Annoy». 2020. ANNOY. 2020. https://github.com/spotify/annoy.

«ARCore». 2020. ARCore. 2020. https://developers.google.com/ar/develop/java/augmented-images/guide.

«ARKit». 2020. ARKit. 2020. https://developer.apple.com/documentation/arkit/.

«ARToolKit». 2020. AR Tool Kit X. 2020. https://github.com/artoolkitx/artoolkitx/wiki.

Athitsos, V., J. Alon, S. Sclaroff, y G. Kollios. 2004. «BoostMap: A Method for Efficient Approximate Similarity Rankings». En *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2004.1315173.

«Augment». 2020. Augment. 2020. https://www.augment.com.

«Aumentaty». 2020. Aumentaty.com. 2020. http://www.aumentaty.com.

Aumüller, Martin, Erik Bernhardsson, y Alexander John Faithfull. 2018. «ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms». *CoRR* abs/1807.05614. https://doi.org/10.1016/j.is.2019.02.006.

«Aurasma». 2020. Aurasma.com. 2020. https://www.aurasma.com.

Azuma, R. T. 1997. «A survey of augmented reality». *Presence: Teleoperators and Virtual Environments* 6 (4): 355-85.

Bay, H., A. Ess, T. Tuytelaars, y L. V. Gool. 2008. «Speeded-up robust features (surf)». *Computer Vision and Image Understanding* 110 (3): 346-59.

Becerra, Martín Ezequiel, Nahuel Mangiarua, Santiago Igarza, Jorge Ierache, y María José Abásolo Guerrero. 2018. «Líneas de investigación del grupo de realidad aumentada aplicada: templates de catálogos aumentados integración escalable de realidad aumentada basada en imágenes y rostros aumentación de sistemas SCADA en el contexto de la industria 4.0». En *Libro de actas del XX Workshop*

de Investigadores en Ciencias de la Computación. http://sedici.unlp.edu.ar/handle/10915/67455.

Boytsov, Leonid, y Bilegsaikhan Naidan. 2013. «Engineering Efficient and Effective Non-metric Space Library». En *Similarity Search and Applications - 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings*, 280–293. https://doi.org/10.1007/978-3-642-41062-8_28.

Chum, Ondrej, James Philbin, y Andrew Zisserman. 2008. «Near Duplicate Image Detection: min-Hash and tf-idf Weighting». En *BMVC 2008 - Proceedings of the British Machine Vision Conference 2008*. https://doi.org/10.5244/C.22.50.

Corneanu, C.A., M.O. Simón, J.F. Cohn, y S.E. Guerrero. 2016. «Survey on RGB, 3D, Thermal, and Multimodal Approaches for Facial Expression Recognition: History, Trends, and Affect-Related Applications». *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38. https://doi.org/10.1109/TPAMI.2016.2515606.

Dantone, M., J. Gall, G. Fanelli, y L.Van Gool. 2012. «Real-Time Facial Feature Detection Using Conditional Regression Forests». En *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2578–85. https://doi.org/10.1109/CVPR.2012.6247976.

«dlib». 2020. dlib. 2020. http://dlib.net/.

Dua, Dheeru, y Casey Graff. 2017. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml.

EmoVu. s. f. «EmoVu Emotion Recognition Software». *EmoVu*. http://emovu.com/e/.

«Facial expression Recognition ResNet». 2020. Facial expression Recognition ResNet. 2020. https://github.com/ybch14/Facial-Expression-Recognition-ResNet.

Freund, Yoav, y Robert E. Schapire. 1995. «A Desicion-Theoretic Generalization of on-Line Learning and an Application to Boosting». En *Computational Learning Theory*, 23–37. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-59119-2_166.

Girshick, Ross B. 2015. «Fast R-CNN». *CoRR Abs/1504.08083*. http://arxiv.org/abs/1504.08083.

Grauman, K., y T. Darrell. 2007. «Pyramid Match Hashing: Sub-Linear Time Indexing Over Partial Correspondences». En *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1-8. https://doi.org/10.1109/CVPR.2007.383225.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, y Jian Sun. 2015. «Deep Residual Learning for Image Recognition». *arXiv:1512.03385 [cs]*, diciembre. http://arxiv.org/abs/1512.03385.

Ierache, Jorge, Nahuel Adiel Mangiarua, Martín Ezequiel Becerra, y Santiago Igarza. 2018. «Framework for the Development of Augmented Reality Applications Applied to Education Games». En *Augmented Reality, Virtual Reality, and*

Computer Graphics, editado por Lucio Tommaso De Paolis y Patrick Bourdot, 340–350. Cham: Springer International Publishing.

Ierache, Jorge, Nahuel Mangiarua, y Martin Becerra. 2014. «Herramienta de Realidad Aumentada para facilitar la enseñanza en contextos educativos mediante el uso de las TICs», Revista Latinoamericana de Ingeniería de Software, 1: 1-3.

Ierache, Jorge, Nahuel Mangiarua, S. Bevacqua, N. Verdicchio, y M. Becerra. 2015. «Development of a Catalogs System for Augmented Reality Applications». En *World Academy of Science, Engineering and Technology, International Science Index*, 9:1-7. http://waset.org/Publications/development-of-a-catalogs-system-for-augmentedreality- applications/10000077.

Ierache, Jorge, Nahuel Mangiarua, N. Verdicchio, y D. Sanz. 2016. «Augmented Card System Based on Knowledge for Medical Emergency Assistance». En *2016 IEEE Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI)*, 1–3. https://doi.org/10.1109/CACIDI.2016.7785979.

Ierache, Jorge Salvador, Nahuel Mangiarua, y Sebastián Ariel Bevacqua. 2014. «Sistema de catálogo para la asistencia a la creación, publicación, gestión y explotación de contenidos multimedia y aplicaciones de realidad aumentada». En *Libro de actas del XX Congreso Argentino de Ciencias de la Computación*. Buenos Aires, Argentina. http://sedici.unlp.edu.ar/handle/10915/42339.

Ierache, Jorge, Javier Urien, y Nahuel Mangiarua. 2018. «Desarrollo de un prototipo de realidad aumentada de tipo geoespacial». En *Libro de actas del XII Jornadas de Vinculación Universidad-Industria*, 5-8. http://sedici.unlp.edu.ar/handle/10915/71289.

Ierache, Jorge, Nicolás Nazareno Verdicchio, Nicolas Duarte, y Nahuel Mangiarua. 2016. «Augmented Reality Card System for Emergency Medical Services». En *Proceedings of the International Work-Conference on Bioinformatics and Biomedical Engineering*, 487-94. Granada, Spain.

Iwasaki, M. 2010. «Proximity search in metric spaces using approximate k nearest neighbor graph (in Japanese)». *IPSJ Trans. on Database* 3 (1): 18-28.

Iwasaki, Masajiro, y Daisuke Miyazaki. 2018. «Optimization of Indexing Based on k-Nearest Neighbor Graph for Proximity Search in High-dimensional Data». *arXiv:1810.07355 [cs]*, octubre. http://arxiv.org/abs/1810.07355.

Jegou, Herve, Matthijs Douze, y Cordelia Schmid. 2008. «Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search». En *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision*, editado por David Forsyth, Philip Torr, y Andrew Zisserman, 304–317. Marseille, France: Springer. https://doi.org/10.1007/978-3-540-88682-2_24.

Jégou, Hervé, Matthijs Douze, y Cordelia Schmid. 2011. «Product Quantization for Nearest Neighbor Search». *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (1): 117-28. https://doi.org/10.1109/TPAMI.2010.57.

Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, y Trevor Darrell. 2014. «Caffe: Convolutional Architecture for Fast Feature Embedding». *arXiv preprint arXiv:1408.5093*.

Jones, David, y Shirley Gregor. 2007. «The Anatomy of a Design Theory». *Journal of the Association for Information Systems* 8 (5): 1.

«Kaggle». 2020. Kaggle. 2020. https://www.kaggle.com/c/painter-by-numbers/overview.

«Kairos». 2020. Kairos. 2020. https://www.kairos.com/.

«KGraph». 2020. KGraph. 2020. https://github.com/aaalgo/kgraph.

Krevelen, Van, Rick, y Ronald Poelman. 2010. «A Survey of Augmented Reality Technologies, Applications and Limitations». *International Journal of Virtual Reality*. https://doi.org/10.20870/IJVR.2010.9.2.2767.

Lapuschkin, Sebastian, Alexander Binder, Klaus-Robert Müller, y Wojciech Samek. 2017. «Understanding and Comparing Deep Neural Networks for Age and Gender Classification». En *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 1629-38. https://doi.org/10.1109/ICCVW.2017.191.

Leutenegger, S., M. Chli, y R. Y. Siegwart. 2011. «Brisk: Binary robust invariant scalable key-points». En *ICCV*, 2548-55. Washington, DC, USA: Proceedings of the 2011 International Conference on Computer Vision.

Li, H., Z. Lin, X. Shen, J. Brandt, y G. Hua. 2015. «A Convolutional Neural Network Cascade for Face Detection». En *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, 5325–34. https://doi.org/10.1109/CVPR.2015.7299170.

Lowe, David G. 2004. «Distinctive Image Features from Scale-Invariant Keypoints». *International Journal of Computer Vision* 60 (2): 91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.

Lucas, B. D., y T. Kanade. 1981. «An iterative image registration technique with an application to stereo vision». En *Proceedings of the 7th International Joint Conference on Artificial Intelligence Volume 2*, 674-79. San Francisco, CA, USA.

Malkov, Yury A., y D. A. Yashunin. 2016. «Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs». *CoRR* abs/1603.09320. http://arxiv.org/abs/1603.09320.

Mangiarua, Nahuel Adiel, Jorge Ierache, y María José Abasolo. 2020. «Scalable integration of image and face based augmented reality». En *Augmented Reality, Virtual Reality, and Computer Graphics*, editado por Lucio Tommaso De Paolis y Patrick Bourdot. Cham: Springer International Publishing.

Mangiarua, Nahuel, Jorge Ierache, y María José Abásolo Guerrero. 2019. «Implementation of an Open Source Based Augmented Reality Engine for Cloud

Authoring Frameworks». *Journal of Computer Science & Technology* 19, n.º 2 (octubre). https://doi.org/10.24215/16666038.19.e16.

Mangiarua, Nahuel, Jorge Ierache, Martín Ezequiel Becerra, Hernán Maurice, Santiago Igarza, y Osvaldo Mario Spositto. 2018. «Framework para la generación de templates en sistema de catálogos de realidad aumentada». En *Libro de actas del XXIV Congreso Argentino de Ciencias de la Computación*. La Plata, Argentina. http://sedici.unlp.edu.ar/handle/10915/73478.

Mangiarua, Nahuel, Jorge Ierache, Martin Becerra, Hernán Maurice, Santiago Igarza, y Osvaldo Spositto. 2019. «Templates Framework for the Augmented Catalog System». En *Computer Science – CACIC 2018*, editado por Patricia Pesado y Claudio Aciti, 267–276. Cham Springer International Publishing: Springer International Publishing. http://sedici.unlp.edu.ar/handle/10915/73478.

Mangiarua, Nahuel, Jorge Ierache, Santiago Igarza, Martín Ezequiel Becerra, Sebastián Ariel Bevacqua, Nicolás Nazareno Verdicchio, Fernando Martín Ortiz, Diego Rubén Sanz, Nicolás Daniel Duarte, y Esteban de la Llave. 2014. «Herramienta de realidad aumentada para la explotación de material didáctico tradicional». En *Libro de actas del IX Congreso sobre Tecnología en Educación & Educación en Tecnología*. La Rioja, Argentina. http://sedici.unlp.edu.ar/handle/10915/38625.

Mangiarua, Nahuel, Cristian Montalvo, Facundo Petrolo, Diego Rubén Sanz, Nicolás Nazareno Verdicchio, E. Lobatto, A. Rosenthal, Martín Ezequiel Becerra, Santiago Igarza, y Jorge Ierache. 2017. «Framework para la generación de templates en sistemas de catálogos de realidad aumentada». En *Libro de actas del XIX Workshop de Investigadores en Ciencias de la Computación*. http://sedici.unlp.edu.ar/handle/10915/61824.

Mikolajczyk, Krystian, y Jiri Matas. 2007. «Improving Descriptors for Fast Tree Matching by Optimal Linear Projection». En *IEEE 11th International Conference on Computer Vision, Volume IEEE 11th*, 1-8. https://doi.org/10.1109/ICCV.2007.4408871.

Milgram, Paul, y Fumio Kishino. 1994. «A Taxonomy of Mixed Reality Visual Displays». *IEICE Trans. Information Systems* E77-D, n.º 12: 1321-29.

Montalvo, C., F. Petrolo, D. Sanz, Nahuel Mangiarua, y Jorge Ierache. 2017. «Knowledge Based Augmented Card System for Medical Assistance Over Mobile Devices». En *Selected Papers of the XXI Argentine Congress of Computer Science*, 257-65.

Muja, M., y D. G. Lowe. 2009. «Fast approximate nearest neighbors with automatic algorithm configuration». En *International Conference on Computer Vision Theory and Application*, editado por VISSAPP'09, 331-40. INSTICC Press.

———. 2014. «Scalable Nearest Neighbor Algorithms for High Dimensional Data». *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (11): 2227-40. https://doi.org/10.1109/TPAMI.2014.2321376.

«OpenCV: 2D Features Algorithms». 2020. 2020. https://docs.opencv.org/4.3.0/d5/d0d/features2d_8hpp.html.

«OpenCV: Experimental 2D Features Algorithms». 2020. 2020. https://docs.opencv.org/4.3.0/d7/d7a/group__xfeatures2d__experiment.html.

Parkhi, Omkar M., Andrea Vedaldi, y Andrew Zisserman. 2015. «Deep Face Recognition». En *Procedings of the British Machine Vision Conference 2015*, 41.1-41.12. Swansea: British Machine Vision Association. https://doi.org/10.5244/C.29.41.

«Repositorio». s. f. Repositorio. Accedido 30 de abril de 2020. https://gitlab.com/frozensun/Engar.

Rosten, Edward, y Tom Drummond. 2006. «Machine Learning for High-Speed Corner Detection». En *Computer Vision – ECCV 2006*, editado por Aleš Leonardis, Horst Bischof, y Axel Pinz, 430-43. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. https://doi.org/10.1007/11744023_34.

Rublee, E., V. Rabaud, K. Konolige, y G. Bradski. 2011. «Orb: An efficient alternative to sift or surf». En *Proceedings of the 2011 International Conference on Computer Vision*, 2564-71. Washington, DC, USA: Proceedings of the 2011 International Conference on Computer Vision.

Salakhutdinov, Ruslan, y Geoffrey Hinton. 2009. «Semantic Hashing». *Int. J. Approx. Reasoning* 50 (7). https://doi.org/10.1016/j.ijar.2008.11.006.

Schroff, Florian, Dmitry Kalenichenko, y James Philbin. 2015. «FaceNet: A Unified Embedding for Face Recognition and Clustering». *CoRR* abs/1503.03832. http://arxiv.org/abs/1503.03832.

Shakhnarovich, G., P. Viola, y T. Darrell. 2003. «Fast Pose Estimation with Parameter-Sensitive Hashing». En *Proceedings Ninth IEEE International Conference on Computer Vision*, 2:750–57. https://doi.org/10.1109/ICCV.2003.1238424.

Shakhnarovich, Gregory, Trevor Darrell, y Piotr Indyk. 2006. «Nearest-Neighbor Methods in Learning and Vision:Theory and Practice». *MIT Press*. http://ieeexplore.ieee.org/servlet/opac?bknumber=6267334.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, y Andrew Rabinovich. 2014. «Going Deeper with Convolutions». *arXiv:1409.4842 [cs]*, septiembre. http://arxiv.org/abs/1409.4842.

Taigman, Yaniv, Ming Yang, Marc'Aurelio Ranzato, y Lior Wolf. 2014. «DeepFace: Closing the Gap to Human-Level Performance in Face Verification». En *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, 1701–1708. IEEE Computer Society. https://doi.org/10.1109/CVPR.2014.220.

Takacs, Gabriel, Vijay Chandrasekhar, Natasha Gelfand, Yingen Xiong, Wei-Chao Chen, Thanos Bismpigiannis, Radek Grzeszczuk, Kari Pulli, y Bernd Girod. 2008. «Outdoors Augmented Reality on Mobile Phone Using Loxel-Based Visual Feature Organization». En *Proceedings of the 1st ACM International*

Conference on Multimedia Information Retrieval, 427–434. MIR '08. New York, NY, USA: ACM. https://doi.org/10.1145/1460096.1460165.

Torralba, A., R. Fergus, y Y. Weiss. 2008. «Small codes and large image databases for recognition». En *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1-8. https://doi.org/10.1109/CVPR.2008.4587633.

Valstar, Michel F., Enrique Sánchez-Lozano, Jeffrey F. Cohn, László A. Jeni, Jeffrey M. Girard, Zheng Zhang, Lijun Yin, y Maja Pantic. 2017. «FERA 2017 - Addressing Head Pose in the Third Facial Expression Recognition and Analysis Challenge». *CoRR Abs/1702.04174*. http://arxiv.org/abs/1702.04174.

Vandergheynst, P., R. Ortiz, y A. Alahi. 2012. «Freak: Fast retina keypoint». En *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 510-17.

Verdicchio, Nicolás Nazareno, Diego Rubén Sanz, Cristian Montalvo, Facundo Petrolo, Nahuel Mangiarua, Santiago Igarza, y Jorge Ierache. 2016. «Sistema de catálogo virtual aumentado: integración de framework especializado orientado a juegos didácticos». En *Libro de actas del XI Congreso de Tecnología en Educación y Educación en Tecnología*. http://hdl.handle.net/10915/54651.

Viola, P., y M. Jones. 2001. «Rapid Object Detection Using a Boosted Cascade of Simple Features». En *, 1:I-511-I-518*. IEEE Comput. Soc. https://doi.org/10.1109/CVPR.2001.990517.

«Vuforia». 2020. Vuforia. Last accessed de 2020. www.vuforia.com/.

Wang, X., T.X. Han, y S. Yan. 2009. «An HOG-LBP Human Detector with Partial Occlusion Handling». En *2009 IEEE 12th International Conference on Computer Vision*, 32–39. https://doi.org/10.1109/ICCV.2009.5459207.

Waring, C.A., y Xiuwen Liu. 2005. «Face Detection Using Spectral Histograms and SVMs». *IEEE Transactions on Systems*. https://doi.org/10.1109/TSMCB.2005.846655.

Weiss, Yair, Antonio Torralba, y Rob Fergus. 2009. «Spectral Hashing». En *Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference*, 1753-60. https://nyuscholars.nyu.edu/en/publications/spectral-hashing.

«Wikiart». 2020. Wikiart. 2020. https://www.wikiart.org/.

Wu, Bo, y Ram Nevatia. 2007. «Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet Based Part Detectors». *International Journal of Computer Vision* 75 (2). https://doi.org/10.1007/s11263-006-0027-7.

«Zappar». 2020. Zappar. 2020. https://www.zappar.com.

Zhang, Cha, y Zhengyou Zhang. 2010. «A Survey of Recent Advances in Face detection». Technical Report MSR-TR-2010-66. One Microsoft Way Redmond, WA 98052: Microsoft Corporation.

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/facedetsurvey.pdf.

Índice alfabético