PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

DESARROLLO DE UN *FRAMEWORK* PARA LA RECUPERACIÓN DE IMÁGENES A PARTIR DEL INGRESO DE DIBUJOS A MANO ALZADA

Tesis para optar el Título de **Ingeniero Informático**, que presenta el bachiller:

Willy Puenternan Fernández

ASESOR: Dr. César A. Beltrán Castañón.

Lima, Junio del 2018.

A mis padres, por sus consejos e inmensos sacrificios con la finalidad de brindarme el mejor obsequio que se le puede dar a un hijo: la educación. Asimismo a mis hermanas Katherine y Karen, por inspirarme cada día a ser una mejor persona. Agradecer también a mis abuelos que ahora me acompañan desde el cielo: a mi abuela Cipriana quien fue, es y será como una segunda madre para mí, por ser la mejor “Mamita” que Dios me haya podido dar, por brindarme siempre su amor infinito e incondicional apoyo a lo largo de toda mi formación universitaria sin importar cuán grande fuese el obstáculo a enfrentar; a mi abuelo "Chemo" por siempre tener fe en mí y en que conseguiría mis metas; y a mi abuela Susana, por enseñarme con su ejemplo de vida a ser una persona amable y sonriente a la vida.

Agradezco a mi asesor, el Dr. Cesar Beltrán Castañón, por orientarme en el desarrollo de mi proyecto de tesis en base a sus enseñanzas impartidas en los diversos cursos durante mi etapa universitaria. Agradecer también al plantel en conjunto de profesores de la universidad, cuyas enseñanzas contribuyeron a mi desarrollo profesional.

Resumen

En la actualidad las personas demandan constantemente información que les ayude a realizar todo tipo de acciones. Ante esta necesidad surgieron los buscadores web y durante un tiempo permitieron la obtención de información de forma óptima. No obstante, ante la creciente generación de contenido multimedia como imágenes y videos, estos buscadores vieron afectados en gran medida sus funcionalidades al ser incapaces de describir a través de palabras el contenido de objetos abstractos presentes en dichas imágenes.

Como alternativas de solución surgen los sistemas de recuperación de imágenes por contenido, cuyo uso se extiende inclusive a la realización de búsquedas más complejas como la recuperación de información en videos. Estos sistemas de recuperación de información visual comprenden dos conocidas áreas: similitud de imágenes y dibujos a mano alzada. En el caso de la búsqueda por similitud de imágenes se permite una mayor aproximación a las imágenes que el usuario desea obtener como resultado de su búsqueda, pero implica que el usuario disponga de una imagen previa de la que desea buscar; por lo que no tiene mucho sentido buscar una imagen si ya se cuenta con otra.

Por otro lado, el uso de dibujos hechos a mano es un medio innato de representación del conocimiento utilizado desde tiempos antiguos y que las personas emplean desde edad temprana. Entonces, ¿por qué no utilizar los dibujos a mano alzada como un parámetro de entrada del motor de búsqueda de imágenes? Es lógico pensar que, mediante el uso de trazos, muchos de los problemas presentes en los buscadores tradicionales serían resueltos. No obstante, el desarrollo de esta alternativa de solución trae consigo nuevas e interesantes dificultades a enfrentar.

En el presente proyecto de fin de carrera se desarrollará un *framework* de recuperación de imágenes mediante la especificación de dibujos a mano alzada como parámetro de entrada. Para ello se creará un algoritmo que priorice la obtención de resultados eficaces a partir del uso de técnicas de inteligencia artificial, visión computacional y sistemas de indexación de imágenes.

El presente documento se encuentra dividido en 7 capítulos, los cuales abarcan lo siguiente: el primer capítulo presenta el contexto sobre el cual actúa el proyecto de tesis, sus objetivos, los resultados y las herramientas utilizadas para la obtención de estos; el segundo capítulo define los conceptos básicos y técnicos necesarios para un mayor entendimiento durante el desarrollo del *framework*; el tercer capítulo presenta una muestra de los trabajos más importantes

aplicados hasta la fecha en el campo de recuperación de imágenes; el cuarto capítulo describe en detalle cómo se ideó la representación de las imágenes según la metodología de bolsa de características; el quinto capítulo hace hincapié en el diseño e implementación del proceso de comparación y recuperación de imágenes a partir del ingreso de trazos a mano alzada por parte del usuario; el sexto capítulo realiza un análisis de los resultados obtenidos y la validación de estos; finalmente, el séptimo capítulo presenta las conclusiones y recomendaciones obtenidas a lo largo del proyecto de tesis.

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: **DESARROLLO DE UN FRAMEWORK PARA LA RECUPERACIÓN DE IMÁGENES A PARTIR DEL INGRESO DE DIBUJOS A MANO ALZADA**

ÁREA: Ciencias de la Computación

ASESOR: Dr. César Armando BELTRÁN CASTAÑÓN

ALUMNO: Willy PUENTERNAN FERNÁNDEZ

CÓDIGO: 20084048

TEMA N°: #659

FECHA: San Miguel, 08 de Marzo de 2017

DESCRIPCIÓN

En la actualidad, gracias al auge del Internet y su uso globalizado, las personas demandan constantemente información que les ayude a realizar todo tipo de acciones. Ante esta necesidad surgieron los buscadores web y durante un tiempo permitieron la obtención de información de forma óptima. No obstante, ante la creciente generación de contenido multimedia como imágenes y videos, estos buscadores vieron afectados en gran medida sus funcionalidades al ser incapaces de describir a través de palabras el contenido de objetos abstractos presentes en dichas imágenes. Ante este escenario se hace evidente la existencia de información visual imposible de obtener mediante las herramientas ya existentes.

Como alternativas de solución surgen los sistemas de recuperación de imágenes por contenido, cuyo uso se extiende inclusive a la realización de búsquedas más complejas como la recuperación de información en videos. Estos sistemas de recuperación de información visual comprenden tres conocidas áreas: similitud de imágenes y dibujos a mano alzada. En el caso de la búsqueda por similitud de imágenes se permite una mayor aproximación a las imágenes que el usuario desea obtener como resultado de su búsqueda, pero implica que el usuario disponga de una imagen previa de la que desea buscar; por lo que no tiene mucho sentido buscar una imagen si ya se cuenta con otra. Ante tal problema, el usuario se ve forzado a realizar búsquedas basadas en palabras demasiado elaboradas o, en el peor de los casos, buscar imágenes una por una (con ayuda de la vista) dentro de un enorme conjunto de imágenes hasta encontrar la que tenía en mente.

Por otro lado, el uso de dibujos hechos a mano es un medio innato de representación del conocimiento utilizado desde tiempos antiguos y que las personas emplean desde edad temprana. Entonces, si lo que se está buscando son mejores medios por donde el usuario pueda comunicar al buscador la imagen que en realidad desea encontrar ¿Por qué no utilizar los dibujos a mano alzada como un parámetro de entrada del motor de búsqueda de imágenes? Es lógico pensar que, mediante el uso de trazos, muchos de los problemas presentes en los buscadores tradicionales serían solucionados. No obstante, el desarrollo de esta alternativa de solución trae consigo nuevas dificultades a enfrentar como la correcta obtención de bordes en las imágenes a color, inexactitud en las comparaciones entre los dibujos hechos a mano con las imágenes y el manejo masivo de imágenes dentro de una base de datos.

En el presente proyecto de fin de carrera se desarrollará un *framework* de recuperación de imágenes mediante la especificación de dibujos a mano alzada como parámetro de entrada. Para ello se creará un algoritmo que priorice la obtención de resultados eficaces a partir del uso de técnicas de inteligencia artificial, visión computacional y sistemas de indexación de imágenes. La realización de pruebas y obtención de resultados se realizarán tomando como muestra un total de diez mil imágenes previamente categorizadas, pertenecientes a bancos de imágenes gratuitos disponibles en la web.

OBJETIVO GENERAL

Desarrollar un *framework* de recuperación de imágenes por contenido basado en la especificación como entrada de un dibujo a mano alzada, mediante la representación de imágenes usando el modelo de bolsa de características.

OBJETIVOS ESPECÍFICOS

Los objetivos específicos son:

OE1. Eliminar todo ruido o imperfección visual que obstaculice la posterior identificación y obtención de características de las imágenes con las cuales el usuario desee trabajar.

OE2. Identificar los objetos o individuos contenidos en las imágenes mediante la generación de un vector de características que represente los atributos más distintivos en las siluetas.

OE3. Definir e implementar una estructura de indexación de imágenes para el manejo de éstas dentro de una base de datos.
OE4. Desarrollar un mecanismo no textual que permita al usuario hacer búsquedas de imágenes abstractas pertenecientes al repositorio local de imágenes.

ALCANCE

El presente proyecto de fin de carrera busca mejorar los medios actuales de búsqueda de información visual, para ello se elaborará un sistema de recuperación de imágenes a partir del ingreso de dibujos a mano alzada. Esta permitirá el ingreso de trazos a través del panel táctil de la laptop, una pantalla táctil o mediante el uso del mouse como si fuese un lápiz. Acto seguido, ésta procederá a buscar en la base de datos de imágenes aquellas que tengan un mayor parecido al esbozo realizado por el usuario. Los resultados serán validados a través de curvas de precisión-exhaustividad de forma que se busque demostrar que el usuario obtiene como respuesta un conjunto de imágenes de objetos con formas semejantes a la que éste dibujó en un principio.

Como modelo de caracterización de las imágenes se utilizará la técnica de bolsa de características, la cual permitirá representar las imágenes según determinadas características distintivas, constituyéndose en el módulo central del trabajo y en el funcionamiento del algoritmo de recuperación de imágenes.

Máximo: 100 páginas

INDICE DE IMAGENES

INDICE DE TABLAS

INDICE GENERAL

1. DEFINICIÓN DEL PROBLEMA

1.1 Introducción

En el presente capítulo se abarcarán aspectos introductorios en el cual se le permita al lector entender de forma clara y concisa la naturaleza del presente proyecto de fin de carrera. Primeramente se verá la problemática en el cual se darán nociones exactas sobre el contexto en el cual actúa el proyecto y los problemas que éste busca solucionar. Luego se verán aspectos como las herramientas, métodos, procedimientos y metodologías sobre las cuales se apoya el proyecto para su desarrollo; asimismo se verá el alcance que éste tendrá en el cual se describirán sus limitaciones y riesgos. Finalmente se verá la justificación del proyecto, en donde se identificarán las principales razones que motivan al desarrollo del proyecto.

1.2 Problemática

Hoy en día, gracias a Internet, las personas viven en una sociedad globalizada que continuamente solicita información que les ayude a realizar todo tipo de acciones, inclusive a tomar decisiones del día a día (Cabello, 2014). Como un medio para conseguir información en la web, se tiene a los motores de búsqueda que facilitan el rápido acceso a la información; sin embargo, la obtención de ésta se encuentra limitada por el modo en que se realizan las consultas: mediante texto. A pesar de que los algoritmos de búsqueda han mejorado bastante, llegando inclusive a realizar búsquedas personalizadas (Mysen, 2011), estos dependen mucho de la entrada textual que se tiene por parte del usuario.

Ante la creciente generación de información mediante imágenes y videos, las consultas a través de la web comienzan a requerir de cierto grado de complejidad con el fin de que el usuario consiga transmitir de forma más precisa la información que desea. Las personas ahora no solo demandan información textual, también buscan obtener imágenes específicas que difícilmente pueden ser descritas usando palabras, por lo que urge encontrar formas de recuperar imágenes de la web con características lo más parecidas posibles a la que el usuario tiene en mente. Es justamente bajo este principio que surgen los sistemas de recuperación de imágenes por contenido o CBIR (*Content-based Image Retrieval*), cuyo uso se extiende inclusive a la realización de búsquedas más complejas como la recuperación de información en videos (Veltkamp, 2013). Estos sistemas de recuperación de información visual comprenden tres conocidos paradigmas: similitud de imágenes, dibujos a mano alzada (*sketch*) y por íconos (Lew, 2013).

El paradigma de búsqueda por similitud de imágenes permite una mayor aproximación a las imágenes que el usuario desea obtener como resultado de su búsqueda, pero implica que el usuario disponga de una imagen previa de la que desea buscar (Lew, 2013); por lo que no tiene mucho sentido buscar una imagen si ya se cuenta con otra. Por otro lado, en la mayoría de ocasiones el usuario no dispone de una imagen, razón por la cual justamente está buscándola en la web. Ante tal problema, el usuario se ve forzado a realizar búsquedas basadas en palabras demasiado elaboradas o, en el peor de los casos, buscar imágenes una por una (con ayuda de la vista) dentro de un enorme conjunto de imágenes hasta encontrar la que tenía en mente (Cao, Wang, Wang, Li, Zhang, & Zhang, 2010).

El uso de dibujos hechos a mano es un medio innato de representación del conocimiento utilizado desde tiempos antiguos y que las personas usan desde edad temprana (Q. Yu, 2015). Entonces, si lo que se está buscando son mejores medios por donde el usuario pueda comunicar al buscador la imagen que en realidad desea encontrar **¿Por qué no utilizar los dibujos a mano alzada como un parámetro de entrada del motor de búsqueda de imágenes?** Es lógico pensar que, mediante el uso de trazos, muchos de los problemas presentes en los buscadores tradicionales serían solucionados.

No obstante, el desarrollo de esta alternativa de solución trae consigo nuevas dificultades a enfrentar y; lamentablemente, hasta la fecha no se ha conseguido dar con una solución completa a estas. Un ejemplo reciente, de los muchos existentes, es el sistema de recuperación de imágenes desarrollado por la compañía Microsoft: "*MindFinder*" (ver Imagen 1). Esta herramienta permite obtener imágenes de una base de datos, sin embargo, ésta al igual que otras aún no consigue desprenderse totalmente del uso de texto para la realización de búsquedas de imágenes más elaboradas.

Imagen 1: Comparativa entre los resultados de *MindFinder* y un buscador tradicional (Cao et al., 2010).

Hasta el momento se han venido desarrollando diversas herramientas que facilitan la recuperación de imágenes en base a su contenido, destacando entre ellas descriptores que realizan comparaciones basadas en la ubicación de siluetas presentes en imágenes ("S*hape Context")* (Hildebrand et al., 2011), metodologías encargadas de la extracción de características locales dentro de una imagen ("*key points")* (Rahman, Antani, & Thoma, 2011) y modelos de procesado de imágenes basados en visión computacional como el enfoque "*bag-of-features"* (Rosado et al., 2015). Pero, **¿qué dificultades se presentan cuando se desea implementar un motor de búsqueda basado en dibujos hechos a mano alzada?** Profundizando más en este punto, las principales barreras que se presentan cuando se desea desarrollar un proyecto de características como las que ocupa este documento se pueden agrupar en tres factores los cuales están fuertemente relacionados entre sí (Cao et al., 2010).

En primer lugar, existe una diferencia significativa entre los dibujos en blanco y negro hechos por el usuario y las imágenes a color presentes en la base de datos del motor de búsqueda: la gran cantidad de colores que pueden estar presentes en una imagen dificulta la extracción de aquellas líneas que indiquen las curvas y/o bordes representativos del objeto central en la imagen. Como segunda barrera tenemos la dificultad en hacer las validaciones respectivas entre las curvas realizadas mediante trazos por el usuario con las imágenes presentes en la base de datos del motor de búsqueda. El usuario nunca realiza dibujos con formas exactamente

iguales a la imagen original, sino que estos mantienen siempre cierto grado de inclinación, posición, adelgazamiento y/o anchura diferente a la imagen que piensa obtener; hacer posible que el programa considere estas características mientras busca coincidencias entre el dibujo y las imágenes en la base de datos es todo un reto a considerar. Finalmente como tercera barrera tenemos que, cuando se trabaja con una base de datos de aproximadamente 10'000 imágenes, su uso se traduce en problemas de escalabilidad, lo que origina que el programa sea incapaz de devolver resultados en tiempo real como ocurre con los buscadores tradicionales basados en texto; esto se podría solucionar mediante una estructura de indexación cuya elaboración tampoco es tarea fácil (Cao et al., 2010).

Ante estas limitantes, es correcto afirmar que en el ámbito relacionado a la recuperación de dibujos hechos a mano alzada aún existe mucho trabajo por hacer; sin embargo, la gama de oportunidades que se generarían por su desarrollo es inmensa y su uso no está limitado de forma exclusiva a los motores de búsqueda, sino que puede ser empleado en otras áreas como la enseñanza didáctica (p.e., aplicativos que enseñen técnicas de dibujo a los niños de los primeros grados de primaria), los video juegos (p.e., aplicaciones capaces de identificar un dibujo y competir contra el jugador (López, 2016)) y la seguridad informática (p.e., el uso de dibujos hechos por el usuario en paneles táctiles como herramienta alternativa al manejo de contraseñas). Es precisamente esta diversidad la principal fuente de inspiración del presente proyecto, es decir, fomentar las bases para el desarrollo de nuevas tecnologías a través del lenguaje de las mentes humanas: el dibujo a mano alzada (Q. Yu, 2015).

En el presente proyecto de fin de carrera se propone desarrollar un *framework* de recuperación de imágenes al cual se le puedan ingresar dibujos a mano alzada. Para ello se desarrollará un algoritmo que priorice la obtención de resultados eficaces a partir del uso de técnicas de inteligencia artificial, visión computacional y sistemas de indexación de imágenes. La realización de pruebas y obtención de resultados se realizarán tomando como muestra inicial un total de 10'000 imágenes previamente categorizadas, pertenecientes a bancos de imágenes gratuitos disponibles en la web como "*Imagenet*", "*FreeImages*", "*OpenPhoto*" y "*morgueFile*" (iPixel Estudio, 2016). Este proyecto transformará la base de datos de imágenes a un nuevo espacio esqueletizado, desarrollará una interfaz que permita la realización de dibujos a mano alzada para finalmente ser ingresados en un modelo que realice búsquedas de imágenes parecidas al dibujo ingresado. Llevar a cabo estos pasos conlleva superar diversos problemas, los cuales serán atacados utilizando el enfoque de *bag-of-features*.

1.3 Objetivo General

Desarrollar un *framework* de recuperación de imágenes por contenido basado en la especificación como entrada de un dibujo a mano alzada, mediante la representación de imágenes usando el modelo de bolsa de características.

1.4 Objetivos Específicos

Estos estarán orientados a caracterizar los objetos contenidos en las imágenes mediante un vector de características que permita identificar los atributos propios de esta.

- Objetivo Específico 1: Desarrollar un pre procesamiento que permita mejorar la nitidez de las imágenes para facilitar la posterior extracción de rasgos característicos.

- Objetivo Específico 2: Identificar los objetos o individuos contenidos en las imágenes mediante la generación de un vector de características que represente los atributos más distintivos en las siluetas.

- Objetivo Específico 3: Definir e implementar una estructura de indexación de imágenes para el manejo de éstas dentro de una base de datos.

- Objetivo Específico 4: Desarrollar un prototipo no textual que permita al usuario hacer búsquedas de imágenes abstractas pertenecientes al repositorio local de imágenes.

1.5 Resultados Esperados

- Resultado 1 para el objetivo 1: Base de datos local de imágenes previamente pre procesadas.

- Resultado 2 para el objetivo 1: Imágenes categorizadas en la base de datos de acuerdo a eventos o tipos de objetos representados en éstas.

- Resultado 3 para el objetivo 1: Base de datos de imágenes transformadas en siluetas.

- Resultado 1 para el objetivo 2: Componente de extracción de características de una imagen mediante la metodología de bolsa de características (*bag-of-features)*.
- Resultado 2 para el objetivo 2: Componente de extracción de características aplicado a los dibujos hechos a mano alzada.

- Resultado 3 para el objetivo 2: Aplicación con los componentes de extracción de características de imágenes y dibujos a mano alzada integrados.

- Resultado 1 para el objetivo 3: Estructura de indexación que facilite la eficaz comparación de imágenes en la base de datos.

- Resultado 1 para el objetivo 4: Algoritmo de recuperación de imágenes por contenido capaz de recibir como entrada un dibujo a mano alzada y que devuelva un conjunto de imágenes pertenecientes a la base de datos con siluetas similares.

- Resultado 2 para el objetivo 4: Módulo de recuperación de imágenes por contenido que integre los módulos de extracción y búsqueda de imágenes.

- Resultado 3 para el objetivo 4: Reportes y análisis de los resultados obtenidos a través del *framework* de recuperación de imágenes.

1.6 Herramientas, métodos, metodologías y procedimientos

1.6.1 Introducción

A continuación se presentarán en el documento las herramientas, métodos y metodologías necesarios para el logro de los objetivos específicos del proyecto de fin de carrera.

1.6.2 Mapeo

Tabla 1: Cuadro de herramientas a usarse.

Resultados Esperados	Herramientas a usarse
RE1: Conjunto de imágenes almacenadas en una base de datos y seleccionadas de acuerdo a un determinado número de categorías y su respectiva transformación a un nuevo espacio (siluetas).	- **ImageNet** es una base de datos de imágenes categorizada. Esta base de datos se encuentra en la web, es de uso libre y cada una de sus categorías cuenta con un rango promedio que va desde 100 a 1000 imágenes. - **OpenCV** es una librería de funciones de uso libre que da soporte a la realización de tareas relacionadas con el procesamiento de imágenes desde Python, tanto en Windows como en Linux.

	- **Algoritmo de Detección de Bordes de Canny** es un algoritmo útil para la siluetización de imágenes. Es considerado uno de los algoritmos de detección de bordes más eficaces.
RE2: Módulo extractor de características que identifique los aspectos más resaltantes de los objetos dentro de las imágenes de la base de datos, como de la imagen a mano alzada.	**- Eclipse Jee Mars** es un entorno de desarrollo integrado de uso libre utilizado para la elaboración de programas de diversos lenguajes de programación, como por ejemplo Python. - **Python 2.7.10** es un lenguaje de programación desarrollado por la *Python Software Fundation*. Este lenguaje es famoso por favorecer la tenencia de un código legible y, por lo mismo, fácil de aprender. - **OpenCV** es una librería de funciones de uso libre que da soporte a la realización de tareas relacionadas con el procesamiento de imágenes desde Python, tanto en Windows como en Linux. - **Servidor Centos 7 con GPU NVidia Grid** brindado por la universidad para el procesamiento masivo de imágenes. - **PuTTY** es una herramienta de uso libre utilizada para facilitar la conexión remota de un equipo a otro (por ejemplo, un servidor). - **WinSCP** es una aplicación de Software libre utilizada para la transferencia segura de archivos entre dos sistemas. **- Bizagi Process Modeler** es una herramienta de uso libre la cual es utilizada para la realización de diagramas de flujo de procesos.
RE3: Estructura de indexación que facilite la eficaz comparación de imágenes en la base de datos.	- **Python 2.7.10** es un lenguaje de programación desarrollado por la *Python Software Fundation*. Este lenguaje es famoso por favorecer a la tenencia de un código legible y, por lo mismo, fácil de aprender.

RE4: Módulo de recuperación de imágenes por contenido que integre los módulos de extracción y búsqueda de imágenes.	**- Eclipse Jee Mars** es un entorno de desarrollo integrado de uso libre utilizado para la elaboración de programas de diversos lenguajes de programación, como por ejemplo Python. - **Python 2.7.10** es un lenguaje de programación desarrollado por la *Python Software Fundation*. Este lenguaje es famoso por favorecer a la tenencia de un código legible y, por lo mismo, fácil de aprender. - **OpenCV** es una librería de funciones de uso libre que da soporte a la realización de tareas relacionadas con el procesamiento de imágenes desde Python, tanto en Windows como en Linux.
RE5: Reportes y análisis de los resultados obtenidos a través del *framework* de recuperación de imágenes.	**- Eclipse Jee Mars** es un entorno de desarrollo integrado de uso libre utilizado para la elaboración de programas de diversos lenguajes de programación, como por ejemplo Python. - **Python 2.7.10** es un lenguaje de programación desarrollado por la *Python Software Fundation*. Este lenguaje es famoso por favorecer a la tenencia de un código legible y, por lo mismo, fácil de aprender. - **Curva de Precisión-Exhaustividad** es un gráfico utilizado para la evaluación del rendimiento de aplicaciones dedicadas a la búsqueda y recuperación de información. A través del uso de ésta se consigue visualizar claramente que tan alto son los valores de la métrica de precisión en comparación con la de exhaustividad.

1.6.3 Herramientas

- **ImageNet:** Es una base de datos de imágenes organizada por categorías de entre 100 a 1000 imágenes cada una y que ha sido desarrollada como un proyecto en conjunto entre las universidades de Standford, Princeton, Michigan y North Carolina. Este tipo de base de datos, debido a su uso libre e interfaz intuitiva, es ampliamente utilizada

en investigaciones académicas enfocadas a la optimización del manejo de la tecnología multimedia (Imagenet, 2016). Esta herramienta será utilizada para la extracción y procesamiento de imágenes categorizadas para posteriormente desarrollar, en base a estas, pruebas experimentales y evaluar los resultados obtenidos.

- **Eclipse:** Es una ambiente de desarrollo integrado gratuito creado por un conjunto de personas que promueven el uso de programas de código abierto. Este programa hasta el momento ha beneficiado a millones de desarrolladores alrededor del mundo. Actualmente la fundación Eclipse es la encargada de dar soporte de forma gratuita a esta herramienta (Eclipse Fundation, 2016).

- **Python 2.7.10:** Es un lenguaje de programación gratuito diseñado por Guido van Rossum, compatible con prácticamente cualquier sistema operativo. Se caracteriza por ser legible y compacta: indicar en pocas líneas de código una acción a realizar por el programa. Este lenguaje, a su vez, incluye herramientas que le permite detectar muchos de los errores de programación que usualmente suelen escapar al control de los compiladores y ofrecer información al programador para que este pueda detectarlos y corregirlos.

  Python constantemente se está actualizando con nuevas versiones mejoradas, no obstante, busca siempre mantener en lo posible la compatibilidad con los programas escritos en versiones anteriores. Con el paso de los años este lenguaje ha ganado popularidad entre los programadores y empresas, siendo ampliamente utilizado en la actualidad (Marzal, 2003). Este lenguaje será utilizado para la implementación del módulo extractor de características el cual identifique los aspectos más resaltantes de los objetos dentro de las imágenes almacenadas en la base de datos, así como de las imágenes a mano alzada hechas por el usuario. Asimismo, también se utilizará en la implementación del módulo de recuperación de imágenes por contenido el cual integrará los módulos de extracción y búsqueda de imágenes.

- **OpenCV (*Open Source Computer Vision*):** Es una librería de funciones multiplataforma de uso libre la cual cuenta con más de 2500 algoritmos las cuales brindan soporte para el desarrollo de aplicaciones pertenecientes a una amplia gama de áreas de investigación informática. El uso de esta herramienta, hoy en día, es muy popular tanto en conocidas empresas como Google, Microsoft, Intel, Yahoo! y Sony; como en pequeñas compañías desarrolladoras que recién están comenzando (OpenCV Developers Team, 2016). Dado que esta herramienta soporta el uso del lenguaje de

programación Python, para el desarrollo del proyecto se utilizarán las librerías de OpenCV relacionadas al pre procesamiento de imágenes. El uso de imágenes de buena calidad es indispensable para que la herramienta de recuperación de dibujos a mano alzada obtenga información visual de forma óptima.

- **Servidor Centos 7 con GPU NVidia Grid:** Es un servidor proporcionado por la universidad para el procesamiento masivo de imágenes. Dado que cuenta con una mayor cantidad de procesadores comparado con computadoras de tipo estándar, se utilizará para la obtención de las características de las siluetas de imágenes presentes en la base de datos a través de técnicas de multiprocesamiento (The CentOS Project, 2016).

- **PuTTY:** Es un cliente SSH y telnet desarrollado inicialmente por Simon Tatham para la plataforma de Windows. Actualmente PuTTY es un programa de uso gratuito el cual pone a libre disposición su código fuente con la finalidad que los que deseen puedan experimentar con este y hacerle mejoras (putty.org, 2016). Se utilizará durante el desarrollo del proyecto para poder acceder al servidor Centos 7 con GPU NVidia Grid brindado por la universidad.

- **WinSCP:** Es una aplicación de Software Libre, es un cliente SFTP gráfico para Windows que emplea SSH. Su función principal es facilitar la transferencia segura de archivos entre dos sistemas informáticos, el local y uno remoto que ofrezca servicios SSH (Martin, 2016). En el desarrollo del proyecto se utilizará para facilitar el envío y recepción de archivos e imágenes.

- **Bizagi Process Modeler:** Es una aplicación gratuita utilizada para diagramar, documentar y simular procesos usando la notación estándar BPMN (*Business Process Modeling Notation*) (Bizagi, 2016). En el proyecto se empleará para representar de forma fácil los algoritmos desarrollados en las diversas etapas de este.

- **Curva de Precisión-Exhaustividad:** Gráfico en el cual se visualiza la relación entre los valores numéricos de las métricas de precisión y exhaustividad, las cuales son ampliamente utilizadas para evaluar los resultados obtenidos por los sistemas de recuperación de información. Mientras se tenga una mayor exhaustividad, habrá una baja precisión y viceversa; esta clase de relación es la responsable de que se generen diferentes curvas según el nivel de precisión y exhaustividad del algoritmo que está

siendo evaluado. Si se comparan dos algoritmos según sus curvas de precisión-exhaustividad, el algoritmo cuya curva abarque la mayor área posible será el mejor (Bordignon, 2008). Esta herramienta se utilizará en la fase final del proyecto a fin de evaluar los resultados obtenidos, para ello se realizarán encuestas a varios usuarios con el fin de evaluar que tan acertadas fueron las imágenes devueltas por el sistema de recuperación de imágenes en respuesta a los dibujos realizados a mano alzada.

- **Algoritmo de Detección de Bordes de Canny:**

  Es un algoritmo muy popular desarrollado por John F. Canny el cual como parte de su funcionamiento pasa por múltiples fases, las cuales son:

  - ✓ Reducción del Ruido: Debido a que la detección de bordes puede verse seriamente afectada por la presencia de ruido en una imagen, lo primero que se debe realizar es eliminar la mayor cantidad de imperfecciones en la imagen posible. Esto se consigue a través del uso de una matriz Gaussiana de cinco filas y cinco columnas la cual actúa como un filtro entre los rasgos importantes de una imagen y lo que no (doxygen, 2016).
  - ✓ Encontrar la intensidad de los gradientes de una imagen: Ya eliminado el ruido en la imagen, se procede a suavizar el tono de la imagen tanto en dirección horizontal (Gx) como vertical (Gy), esto con la finalidad de obtener la primera derivada en ambas direcciones mencionadas. A partir de esto, es factible encontrar la gradiente y dirección de cada uno de los pixeles presentes en la imagen. Cabe recalcar que la dirección de los gradientes siempre serán perpendiculares a los bordes (doxygen, 2016).
  - ✓ Supresión del no máximo: Después de haber obtenido la magnitud y dirección del gradiente, se debe hacer un escaneo completo de la imagen con el objetivo de remover cualquier tipo de pixel que no forme parte del borde en sí. Para ello, en cada pixel se verifica si este es un máximo local entre sus vecinos en la dirección del gradiente (doxygen, 2016).

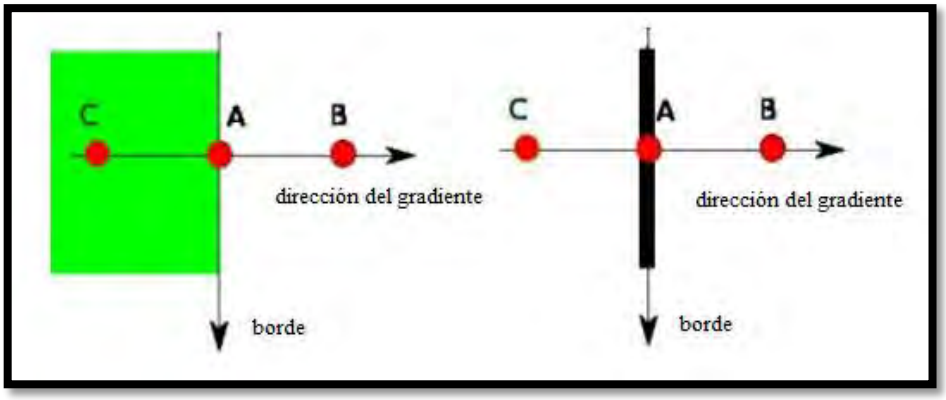


Imagen 2: Ejemplificación de la ubicación de la gradiente de un pixel respecto al borde de la imagen (imagen obtenida del URL: http://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html).

Por ejemplo, (ver imagen 2) el punto A está en el borde (en dirección vertical) y la dirección del gradiente es perpendicular a este. Los puntos B y C se encuentran en la misma dirección que el gradiente, así que el punto A es verificado con el punto B y C a fin de ver si conforman un máximo local. Si lo forman el algoritmo pasa a la siguiente etapa; caso contrario, este es suprimido. Con esto se consigue obtener una imagen binaria con bordes delgados (doxygen, 2016).

- ✓ Definición de los límites o umbrales: En esta etapa se decide cuales bordes de todos los detectados son realmente bordes útiles para el siluetizado de la imagen y cuáles no. Para ello es necesarios dos valores máximos y mínimos. Cualquier gradiente con una intensidad mayor que el valor máximo será clasificado como un borde con certeza, mientras que aquellos cuya intensidad sea menor que el valor mínimo serán descartados. Aquellos que estén entre los valores máximos y mínimos deberán se clasificados como bordes o no dependiendo de su conectividad con otros puntos. Si están conectados a pixeles previamente clasificados como bordes con certeza, estos pasaran a ser considerados como parte de los bordes; caso contrario, serán descartados. Es importante señalar que la calidad de siluetización de la imagen dependerá precisamente de que tan bien establecidos estén estos dos umbrales (doxygen, 2016).

Imagen 3: Ejemplificación de la siluetización de una imagen (imagen obtenida del URL: http://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/).

Finalmente, luego de todos estos pasos se obtiene una imagen con bordes fuertemente definidos siempre y cuando se ajusten adecuadamente los límites mínimos y máximos manejados por el algoritmo de Canny.

1.6.4 Métodos y Procedimientos

En este apartado se hace una descripción de los métodos y procedimientos que se llevarán a cabo para y durante el funcionamiento del *framework* de recuperación imágenes basado en el ingreso de dibujos a mano alzada. Además, se presenta un marco metodológico en el cual se sintetizan los pasos por los cuales atraviesa el proyecto durante su ejecución. Estos últimos han sido categorizados en cuatro fases: ingreso y salida de datos, análisis del dibujo a mano alzada, comparación entre el dibujo e imágenes y, finalmente, indexación y análisis de las imágenes en la base de datos.

En muchos de los métodos y procedimientos descritos a continuación se utilizarán diversas metodologías las cuales serán desarrolladas con mayor detalle en otro apartado dedicado especialmente a éstas, la presente sección busca únicamente dar luces sobre cada una de las etapas por los que pasa el proyecto.

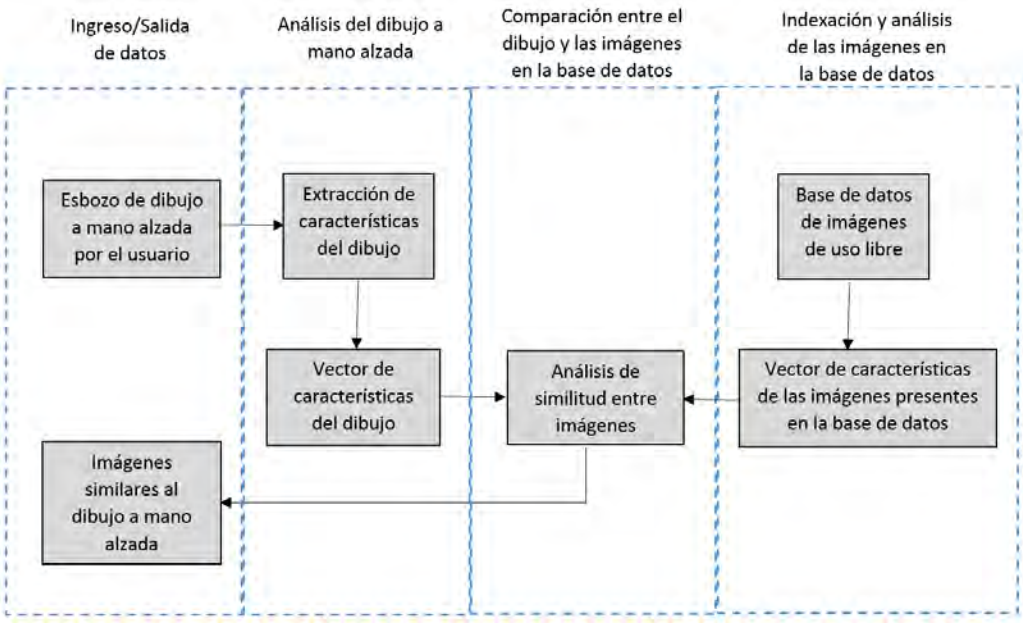


Imagen 4: Marco metodológico del proyecto de fin de carrera (imagen elaborada por el autor).

A continuación se procede a describir cada uno de los elementos del marco metodológico:

- Realización del dibujo a mano alzada:
  En esta etapa inicial el usuario realiza un dibujo a través del panel táctil de una laptop, una pantalla táctil o mediante el uso del mouse como si fuese un lápiz y lo ingresa al componente. Este último contará con un espacio a modo de lienzo sobre el cual será posible realizar dibujos utilizando cualquiera de las vías ya mencionadas (Dibujosinfantiles, 2016).

- Extracción de características del dibujo:
  Previo al inicio de esta etapa se hace un pre procesamiento de la imagen ingresada con el objetivo de trabajar siempre con dibujos de la mejor calidad posible. A continuación se procede, mediante la puesta en práctica de metodologías de recuperación de información visual que se detallarán más adelante, con la extracción de únicamente los rasgos más significativos (características) del dibujo a mano alzada hecho por el usuario (Hildebrand, Eitz, Boubekeur, & Alexa, 2011).

- Uso de un vector de características para el dibujo:
  Una vez identificados en el dibujo los rasgos más importantes para la identificación del mismo, se procede a diseñar un vector de características sobre el cual ingresar estos junto con otros datos útiles como por ejemplo la ubicación dentro del lienzo de

las características locales de la imagen y el tamaño del dibujo completo en el lienzo (Hildebrand, Eitz, Boubekeur, & Alexa, 2011).

- Procesamiento de la base de datos de imágenes:
  En esta etapa se lleva a cabo el análisis de las imágenes almacenadas dentro de la base de datos sobre la cual se apoyará la herramienta de recuperación de imágenes. Habiendo diseñado previamente una estructura de indexación para la base de datos, se analizará cada una de las imágenes dentro de esta con la finalidad de extraer sus características más importantes. La metodología que se utilizará para la obtención de estas será la misma que se utilizó en el procedimiento de extracción de características del dibujo hecho a mano alzada (Melliy al Annamalai, 2000).

- Uso de un vector de características para las imágenes en la base de datos:
  Una vez identificadas las características más importantes en cada una de las imágenes dentro de la base de datos, se procede a diseñar (de forma similar a como se hizo para los dibujos a mano alzada) para cada una de estas un vector de características sobre el cual ingresar aquellos rasgos identificativos que le permiten diferenciarse del resto de imágenes. Adicionalmente, los vectores incluirán otros datos útiles como el tamaño y la ubicación (en coordenadas) de las características locales de la imagen (Agustini Melchor & Valiente Gonzáles, 2001).

- Análisis de similitud entre imágenes:
  En esta etapa se procede a comparar las características extraídas del dibujo a mano alzada hechas por el usuario con aquellas pertenecientes a cada una de las imágenes almacenadas dentro de la base de datos. El componente debe ser capaz de reconocer lo dibujado por el usuario, para ello deberá hacerse uso de un algoritmo adecuado. El desarrollo e implementación de este último es precisamente lo que mayor tiempo y esfuerzo demandará en el presente proyecto de fin de carrera (Feijo, 2009).

- Devolución de imágenes similares al dibujo hecho por el usuario:
  Como su nombre lo indica, a partir de la evaluación del conjunto de trazos hechos por el usuario, se mostrarán aquellas imágenes que el sistema de recuperación considere que tienen una alta probabilidad de ser las que el usuario estuvo pensando al momento de realizar su dibujo. Es importante indicar que primero se visualizarán aquellas imágenes que tengan un alto nivel de parecido con el dibujo ingresado al componente y luego se irán mostrando de forma secuencial aquellas imágenes menos similares; no

obstante, solo se mostrarán aquellas cuyo nivel de similitud estén por encima de un determinado valor previamente establecido (Cao, y otros, 2010).

1.6.5 Metodologías

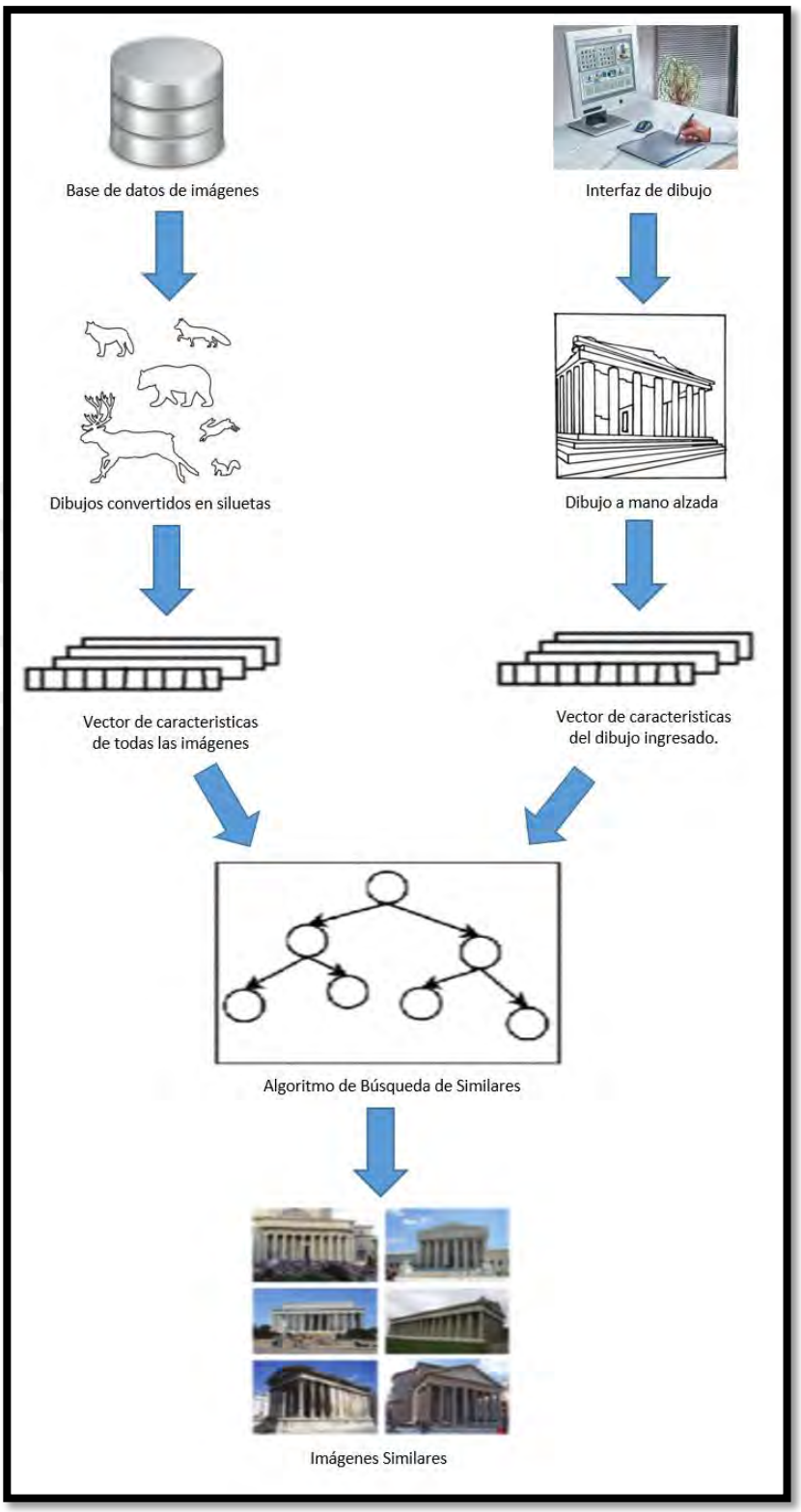


Imagen 5: Esquema de la metodología empleada (imagen elaborada por el autor).

En esta sección se indicarán las metodologías bajo las cuales se desarrollarán los métodos y procedimientos mencionados líneas arriba.

- Modelo Bolsa de Características (*Bag-of-Features*)

  Es un método mayormente conocido por su nombre en inglés y ampliamente utilizado en la categorización y recuperación de información. Actúa de forma similar al modelo *Bag-of-Words* con la diferencia de que en vez de trabajar con palabras para el procesamiento de documentos, se enfoca en la representación de imágenes para su posterior clasificación (Feijo, 2009). Es decir, si a través del modelo *Bag-of-Words* cada documento es representado según la frecuencia de las palabras contenidas en este y sin tomar en cuenta la representación sintáctica existente entre dichas palabras; el modelo *Bag-of-Features* evalúa de forma análoga las imágenes como un conjunto de regiones que describen su apariencia pero ignoran su estructura espacial (Rosado et al., 2015).

  Para la recuperación y clasificación de información textual los documentos usualmente son separados en palabras atómicas y sin ninguna clase de significado semántico. Sin embargo, este enfoque no es aplicable del todo cuándo se trabaja con imágenes puesto que un pixel por sí solo no aporta suficiente información. En lugar de eso, se consideran recuadros de la imagen más grandes llamados descriptores o características locales las cuales si bien no tienen un límite de tamaño y número, necesitan que sus ubicaciones dentro de la imagen estén debidamente muestreadas. En el presente proyecto se seleccionará de forma aleatoria la ubicación de los descriptores locales dentro de una imagen con el objetivo de asegurar una equitativa selección de contenido tanto relevante como irrelevante (Hildebrand et al., 2011).

  El uso exitoso de esta metodología para la representación de imágenes en proyectos afines ha demostrado ser una de las mejores técnicas a emplearse debido a que tienen impresionantes niveles de rendimiento en tareas de categorización de imágenes (Rosado et al., 2015), razón por la cual será utilizado en el proyecto.

- Generación de vocabulario (*codebooks)*

  Esta metodología es aplicada una vez obtenidas las características locales de las imágenes, ya que lo que se busca es generar un vocabulario visual (*codebook*) capaz de representarlas. Para ello se emplean algoritmos de *clustering* como el k-*means* cuya función consiste en organizar las características locales según su contenido, formando palabras visuales (*codewords*) que serán parte del mencionado vocabulario (Feijo,

2009). La generación del *codebook* es una consecuencia directa del entrenamiento al que se somete el algoritmo de clasificación, especialmente si es que este se encuentra basado en redes neuronales. Por lo tanto, el grado de eficiencia del mencionado algoritmo dependerá de que tan bien entrenado este se encuentre y; por ende, que tan extenso es el *codebook* (Hildebrand et al., 2011). Esta metodología será utilizada en el proyecto puesto que permite reducir el volumen de los datos con los que se trabaja al asignar cada descriptor a un único *codeword* (Feijo, 2009).

- Descriptor de siluetas *(Shape Context)*

  Es un descriptor de características utilizado ampliamente para comparar objetos de una imagen a partir de la forma de sus contornos. Consiste en establecer una determinada cantidad de puntos sobre los bordes de una imagen para acto seguido trazar vectores que conecten cada uno de estos entre sí, con ello se consigue tener una descripción detallada del contorno de una imagen (Hildebrand et al., 2011).

  Esta metodología resulta de gran utilidad cuando se trabaja con imágenes de siluetas o dibujos hechos a mano alzada, puesto que permite obtener información detallada de cada una de las características locales extraídas de esta; obteniéndose vectores de características con mejor información. Para ello basta con hacer algunas adaptaciones válidas; es decir, por cada característica local se ubicarán puntos de manera aleatoria siempre y cuando estos coincidan con algún trazo del dibujo en dicha región. Ubicados dichos puntos, se procede a trazar vectores dirigidos hacia el eje central de esta región y finalmente se almacena la ubicación de estos puntos respecto a dicho eje (Hildebrand et al., 2011).

- Descriptor de siluetas (*Spark Feature)*

  Es una extensión del descriptor *Shape Context* especializado en tareas afines a la recuperación de imágenes basados en trazos y por lo mismo de gran utilidad para el presente proyecto. Primero se generan puntos aleatorios a lo largo de todo el dibujo, cuidando de que estos no coincidan con trazo alguno. Luego, por cada uno de estos puntos se trazan líneas en direcciones aleatorias hasta que una de estas coincida con un trazo del dibujo. Finalmente, cuando se dé este caso, se procede a almacenar toda la información posible relacionada a dicho punto (Hildebrand et al., 2011).

- Histograma de gradientes orientados (HoG)

  Es un algoritmo usualmente utilizado en investigaciones relacionadas al procesamiento de imágenes para representar características. Esta técnica parte bajo la

premisa de que toda imagen puede ser descrita utilizando la distribución de las gradientes y, mediante ellas, obtener las características más relevantes. Un aspecto importante a señalar sobre el funcionamiento del histograma de gradientes orientados es que extrae las características considerando siempre los bordes presentes en la imagen. Para efectos del desarrollo del proyecto es posible optimizar esta metodología almacenando solamente en el histograma las características más importantes que se consigan encontrar al momento de procesar los trazos hechos por el usuario (Hildebrand et al., 2011).

1.7 Alcance

El presente proyecto de fin de carrera busca mejorar los medios actuales de búsqueda de información visual, para ello se propone la elaboración de un *framework* el cual permita recuperar imágenes a partir del ingreso de dibujos a mano alzada. Esta permitirá el ingreso de trazos a través del panel táctil de la laptop, una pantalla táctil o mediante el uso del mouse como si fuese un lápiz.

Acto seguido, ésta procederá a buscar en la base de datos de imágenes aquellas que tengan el mayor parecido al esbozo realizado por el usuario. Los resultados serán validados a través de curvas de precisión-exhaustividad de forma que se busque demostrar que el usuario obtiene como respuesta un conjunto de imágenes de objetos con formas semejantes a la que éste dibujó en un principio.

Para lograr este objetivo se utilizará el modelo de procesamiento de imágenes *Bag-of-Features*, el cual permitirá categorizar las imágenes según determinados características distintivas las cuales jugaran un rol importante en el funcionamiento del algoritmo de recuperación de imágenes de acuerdo al conjunto de trazos ingresados por el usuario. Por ello, debe considerarse adicionalmente los siguientes aspectos:

- Los resultados de las búsquedas llevadas a cabo por el componente serán bastante precisos, por lo que no serán inmediatos; es decir, no serán devueltos de forma inmediata.

- La herramienta de recuperación de imágenes trabajará únicamente con dibujos hechos en blanco y negro, simulando la realización de bosquejos o trazos a mano alzada sobre papel.

- Solo se trabajará con dibujos a mano alzada de siluetas; es decir, si el trazo es de color negro, el componente no tendrá soporte para hacer búsquedas de imágenes con base en dibujos a mano alzada con fondos de color negro y viceversa.

- Las búsquedas hechas se limitarán a ser ejecutadas sobre una base de datos local de imágenes debidamente clasificada e indexada.

1.7.1 Limitaciones

- Debido a la escasez de bases de datos masivas de imágenes a color transformadas en siluetas que sean de uso libre en la web, se trabajará en su mayoría con una base de datos categorizada de dibujos hechos a mano alzada los cuales simulen la silueta de una imagen a color transformada en silueta mediante el algoritmo de Canny.
- Para el procesamiento de imágenes masivas se empleará un servidor proporcionado por la universidad. Si bien esta herramienta facilitará el desarrollo del *framework*, su uso está sujeto a determinadas normas por parte de la universidad; imposibilitándose algunas acciones (a nivel de administrador) en el servidor.

1.7.2 Riesgos

A continuación se presenta un cuadro en el que se identifican los riesgos del proyecto, el impacto que estos representan para su desarrollo y las medidas correctivas a realizar en caso de que ocurriesen a fin de mitigarlos a tiempo.

Tabla 2: Cuadro de Riesgos.

Riesgo identificado	Impacto en el proyecto	Medidas correctivas para mitigar
Pérdida inesperada de las imágenes en la base de datos.	Resulta imposible llevar a cabo pruebas de rendimiento al proyecto. El proyecto se vería estancado, ocasionando retraso en su desarrollo. Impacto Alto.	Realizar con frecuencia constantes respaldos de información del contenido en la base de datos y almacenarlas en la nube, dispositivos físicos como USB y discos duros.

Elaboración defectuosa del algoritmo de recuperación de imágenes.	Mala obtención de resultados por parte del componente. Impacto Alto.	Se debe considerar todos los escenarios posibles al momento de elaborar el algoritmo. Antes de dar por correcto el funcionamiento de este se deben realizar pruebas rigurosas que demuestren que no existen errores.
Pérdida del avance de lo programado en el proyecto hasta el momento.	Se tendría que iniciar nuevamente la programación del proyecto desde el comienzo. Se duplicaría el tiempo estimado para la implementación del proyecto. Impacto Alto.	Se debe tener más de un back up en el que se vaya registrando el avance en el proceso. Por ejemplo, repositorios en la nube, medios de almacenamiento físicos como unidades USB y discos duros.
Pérdida de la documentación del proyecto.	Se tendría que hacer nuevamente la documentación del proyecto puesto que sin una documentación adecuada se corre el riesgo de encaminar de forma errónea el desarrollo del proyecto. Volver a realizar la documentación ocasionará una pérdida de tiempo considerable la cual podría emplearse en otras actividades asociadas al proyecto. Impacto Medio.	Se debe tener más de un respaldo de información en el que se vaya registrando el avance obtenido a lo largo del proyecto como almacenamiento en la nube, discos duros y/o unidades USB.
Inoperatividad repentina de las librerías sobre las cuales trabaja la herramienta de recuperación de imágenes.	La funcionalidad del componente se verá afectada, volviéndose obsoleta en el peor caso. Impacto Medio.	Se debe utilizar librerías de confianza. Disponer de más de una librería con las mismas herramientas en lo medida de lo posible.

Falta de disponibilidad por parte de los usuarios al momento en que se desee evaluar la percepción de este respecto al funcionamiento del componente.	Imposibilidad de elaborar las curvas de precisión-exhaustividad con las cuales se consigue evaluar la eficiencia del algoritmo desarrollado. Impacto Medio.	Se debe coordinar con tiempo las entrevistas con los usuarios.
Dificultad para implementar ciertas funcionalidades del componente a través del lenguaje de programación seleccionado por falta de herramientas propias de esta.	Incapacidad para continuar con el desarrollo del proyecto. Impacto Medio.	Asegurarse desde un inicio de que el lenguaje de programación seleccionado para llevar a cabo el proyecto cuenta con todas las herramientas necesarias.
Mal funcionamiento del manejador de base de datos elegido.	Retraso del proyecto por problemas de incompatibilidad. Impacto bajo.	Disponer de más de un sistema de administración de base de datos.

1.8 Justificación

1.8.1 Justificativa del proyecto de tesis

Como se mencionó en la problemática, en la actualidad las personas no solo demandan la obtención acertada de información de tipo textual sino que, gracias a la acelerada evolución del contenido multimedia y su rápida inclusión en las herramientas tecnológicas más utilizadas por la sociedad, se hace indispensable el desarrollo de herramientas que permitan obtener información visual (imágenes) de manera ágil y sobre todo precisa. La obtención de imágenes a partir del uso de herramientas de búsqueda de información textual en muchas ocasiones no devuelve los resultados esperados debido a que existen casos en los que las imágenes presentan rasgos abstractos imposibles de describir a través de palabras.

El proyecto de fin de carrera propone el desarrollo de un *framework* de recuperación de imágenes por contenido basado en la especificación como entrada de un dibujo a mano alzada.

Las personas desde la antigüedad han empleado el uso de dibujos como un medio para expresar y por lo mismo transmitir conocimientos e información, por lo que se buscará adecuar el uso de esta técnica como una herramienta para la obtención acertada de imágenes. Si bien el proyecto, dada su naturaleza, tiene un impacto global sobre la sociedad y su continua demanda de información, la investigación implícita en el desarrollo de éste abre la posibilidad de que su aplicación se extienda a áreas como la educación, desarrollo de video juegos y el manejo de la seguridad en informática.

Debido a la inexactitud en las imágenes obtenidas por otros sistemas de recuperación pertenecientes al estado del arte, para el presente proyecto se priorizará la optimización de los resultados obtenidos sobre el tiempo que éste pueda requerir para ser ejecutado; para ello se desarrollara un algoritmo basado en la metodología de bolsa de características (*Bag-of-features*) que sea capaz de realizar las comparaciones respectivas entre el dibujo realizado por el usuario y las imágenes almacenadas en la base de datos.

2. MARCO CONCEPTUAL

2.1 Introducción

En la presente sección se desarrollará una explicación concisa de determinados conceptos que el autor considera pertinentes para esclarecer cualquier duda a nivel conceptual que pudiese existir en el planteamiento de la problemática expuesta en el capítulo anterior.

2.1.1 Objetivo del marco conceptual

El objetivo principal de marco conceptual es explicar al lector de forma detallada aquellos conceptos necesarios para el desarrollo del proyecto de fin de carrera que puedan escapar del entendimiento del lector. Los conceptos mostrados en esta sección están divididos en conceptos básicos y conceptos técnicos. En la primera parte se explican los términos relacionados con la elaboración de dibujos e imágenes digitales, mientras que en la segunda parte, se definen aquellos conceptos asociados con las herramientas que servirán de soporte para el proyecto. Es necesario tener bien entendidos los conceptos presentados en ambas partes para poder entender todo el proceso de desarrollo.

2.1.2 Conceptos Básicos:

- *Dibujo a mano alzada:*

  Es aquella técnica de dibujo que se realiza sin hacer uso de herramientas auxiliares, sino que se lleva a cabo únicamente con la mano y el lápiz u otro instrumento similar; por ejemplo un rotulador, carbón, crayón o incluso un mouse (siempre y cuando solo sea usado para hacer trazos). Si bien estos tipos de dibujos no se realizan a escala, es posible emplear técnicas con el lápiz como el sombreado, claroscuro y texturizado. Para realizar este tipo de dibujo es necesario tener libertad de movimiento en la muñeca para trazar líneas, aunque en ocasiones se emplea un truco que consiste en apoyar la mano sobre el papel o lienzo y deslizarla a lo largo de esta, afín de conseguir líneas más rectas, evitando que la mano tiemble (Dibujosinfantiles, 2016).

  Existen distintos tipos de dibujo a mano alzada, estos son:

  - ✓ Dibujo artístico: Es aquel tipo de dibujo utilizado para expresar ideas, sentimientos y emociones. El artista dibuja las cosas no como son, sino como éste las percibe por lo que es necesario de un talento innato (Dibujosinfantiles, 2016).
  - ✓ Dibujo del natural: Es aquel dibujo que copia los objetos directamente tal y como son observados por el artista y el resto de personas. Para este tipo de

dibujo se trabaja con el modelo en frente y no se hace de memoria, se busca copiar el objeto lo más preciso posible (Dibujosinfantiles, 2016).

- ✓ Croquis: Es un dibujo rápido y eficaz el cual solo captura los detalles más importantes de una imagen sin detenerse en detalles pequeños que aportan pocos datos útiles (Dibujosinfantiles, 2016).
- ✓ Caricatura: Es un dibujo a modo de retrato el cual exagera o distorsiona la forma del modelo a dibujar. Su técnica se basa en exagerar el rasgo más marcado y/o menos atractivo que el modelo presente. Su uso es mayormente empleado con fines humorísticos (Dibujosinfantiles, 2016).

2.1.3 Conceptos Técnicos:

- *Base de datos:*

  Es el conjunto de datos pertenecientes a un mismo contexto y que son almacenados sistemáticamente para su posterior uso. En la actualidad, debido al desarrollo tecnológico de la informática y la electrónica, la mayoría de bases de datos están en formato digital logrando dar soluciones al problema de almacenamiento de datos. Esto les permite además ser recolectados y explorados por sistemas de información. Existen dos tipos de base de datos clasificados según la variabilidad de los datos que contienen: base de datos estáticas, los cuales son de solo lectura y utilizadas mayormente para almacenar datos históricos; y base de datos dinámicas, en el cual la información se va modificando con el tiempo con lo cual es posible operaciones de actualización, borrado y adición de datos, además de las operaciones de consulta (Cáceres, 2011).

- *Motor de búsqueda:*

  También conocido como buscador, es un programa *software* que busca sitios web basándose en palabras clave (*keywords*) designadas como términos de búsqueda. Los motores de búsqueda crean listados de sitios web utilizando arañas (*spiders*) que rastrean (*crawl*) las páginas web, indexan su información y siguen los enlaces desde ellas hacia otras páginas. Las arañas regresan con frecuencia a los sitios ya rastreados para comprobar actualizaciones o cambios, todo lo que encuentran queda reflejado en la base de datos del motor de búsqueda (Curiel, 2014).

  Existen distintos tipos de motores de búsqueda, se tienen los que obtienen los resultados de otros múltiples motores de búsqueda llamados “metabuscadores”, estos permiten a sus usuarios ingresar criterios de búsqueda una sola vez, y acceder a

múltiples buscadores de forma simultánea. Por otra parte, existen los llamados "buscadores especializados", que se encargan de restringir la búsqueda en la web a aquellos recursos que cumplen una serie de requisitos como tipo de documento (fotos, blogs, videos, libros, artículos, etc.), materia (ciencia, humanidades, etc.) o nivel de la información (documentación de carácter científico y académico) (Curiel, 2014).

- *CBIR (Content-based Image Retrieval):*

  La Recuperación de Imágenes por Contenido (mayormente llamado CBIR por sus siglas en inglés) es un sistema de búsqueda encargado de recuperar imágenes mediante las características presentes en esta. Al momento de representar una imagen dentro de este sistema se consideran dos procesos: la extracción de características y la construcción de descriptores visuales para el almacenamiento y recuperación de la imagen (Nora La Serna Palomino, 2010).

  El primer proceso consiste en la extracción de características clasificadas como de bajo y alto nivel. Las características de bajo nivel son consideradas las más básicas en una imagen y proporcionan información visual como el color, la textura y la forma; estas se presentan como de bajo nivel en contraste con las características de alto nivel que definen conceptos como montaña, cielo, células o una persona en particular. Además, existen dos métodos de extracción de características: global (realizadas sobre imágenes enteras) y local (realizadas sobre un grupo pequeño de pixeles dentro de una imagen) (Nora La Serna Palomino, 2010).

  Respecto al segundo proceso, un descriptor permite representar una imagen según las características que este tenga y usualmente es una formulación matemática que se realiza tanto para características de tipo locales como globales (Nora La Serna Palomino, 2010). Es posible distinguir tres tipos de descriptores que permitan realizar un análisis de imágenes a tres niveles con diferente capacidad de abstracción (Agustini Melchor & Valiente Gonzáles, 2001), estas son:

  - ✓ Nivel 1: Primitivas básicas

    En este nivel es posible establecer relaciones entre las imágenes en base al color, textura y contornos (Agustini Melchor & Valiente Gonzáles, 2001).
  - ✓ Nivel 2: Sintáctico

    Aquí se lleva a cabo un reconocimiento de patrones en la imagen basado en conocimientos previos asociados al tipo de imagen que se está analizando (Agustini Melchor & Valiente Gonzáles, 2001).

- ✓ Nivel 3: Semántico

  En este nivel la complejidad es mayor ya que es posible que existan imprecisiones entre la definición del objeto de la realidad a buscar y sus posibles transformaciones al ser plasmados en una imagen de dos dimensiones: percepción del objeto (Agustini Melchor & Valiente Gonzáles, 2001).

Los sistemas de consultas en bases de datos de imágenes inicialmente hacen un pre proceso de la imagen con la cual obtienen un conjunto de descriptores útiles al momento de hacer las búsquedas. Este pre procesamiento evita el costo computacional de hacer, en cada consulta, un análisis de contenido de los elementos que componen la base de datos de imágenes y limita las búsquedas únicamente a combinaciones entre los descriptores. Este tipo de análisis es efectivo para conjuntos de imágenes de gran tamaño, fácilmente distinguibles del fondo y cuyos valores de color, tamaño o posición como caracteres, caras o representantes de categorías (p.e., gatos, peces o sillas) se pueden normalizar. Sin embargo, no es suficiente para analizar imágenes que presenten una gran cantidad de elementos representativos en su contenido (Agustini Melchor & Valiente Gonzáles, 2001).

En los últimos años se han realizado diversas propuestas abordando diferentes aspectos de la información contenida en las imágenes como las texturas, similitud de formas y relaciones semánticas entre los objetos de una imagen. Por décadas se han venido desarrollando diversos estudios relacionados a este tipo de consulta con regular éxito, por lo que en la actualidad representa todo un desafío (Agustini Melchor & Valiente Gonzáles, 2001).

- *Recuperación de imágenes por similitud:*

  Tipo de búsqueda de imágenes basado en CBIR donde el usuario selecciona una imagen en particular para que acto seguido el sistema le responda mostrándole una lista de imágenes con características similares a la inicialmente seleccionada por este (Lew, 2013).

- *Recuperación de imágenes por Sketch:*

  Tipo de búsqueda de imágenes basado en CBIR en el cual un dibujo hecho a mano por el usuario es ingresado al sistema de búsqueda para que este encuentre imágenes con características parecidas (p.e., el contorno de la imagen) al dibujo ingresado (Lew, 2013).

- *Recuperación de imágenes por iconos :*

  Tipo de búsqueda de imágenes basado en CBIR en el cual el usuario ingresa iconos simbólicos en lugares específicos de la imagen donde este cree que debe haber una mayor cantidad de características de imágenes (Lew, 2013).

- *Sistemas de indexación de imágenes:*

  Es un proceso complejo que consiste en varias etapas y tiene como objetivo encontrar el mayor número de características que mejor representen la información contenida en cada imagen. Un sistema de indexación de imágenes, a diferencia de otras indexaciones, no es una simple estructura de datos sino una colección de objetos de base de datos (Melliy al Annamalai, 2000) cuyos componentes principales son:

  - ✓ Tabla de características: Cada imagen tiene una fila en esta tabla y cada una de estas filas contiene un conjunto de números los cuales son representaciones cuantitativas de rasgos distintivos de una imagen en particular (Melliy al Annamalai, 2000).
  - ✓ Índices de mapas de bits: Es un conjunto de índices localizados a modo de columnas en la parte superior de la tabla de características (Melliy al Annamalai, 2000).

- *Modelo bolsa de palabras (Bag-of-Words):*

  Es un método muy utilizado en la categorización de información textual en la cual se consigue que, en efecto, cada documento a analizar parezca una bolsa conteniendo palabras. Bajo esta metodología cada uno de los documentos son tratados como un histograma en el cual se cuenta la frecuencia de cada una de las palabras que la conforman independientemente del orden en que estas aparecen (Piñar, 2012).

- *Visión Computacional:*

  Rama de la inteligencia artificial la cual busca modelar, a través del uso de cálculos y análisis matemáticos, la percepción visual propia de los seres vivos y con ello desarrollar programas de computadora que simulen estas capacidades (Crespín Luis & Julián García, 2014). Esta se divide en seis etapas:

  - ✓ Sensado: Proceso mediante el cual se logra obtener una imagen (Crespín Luis & Julián García, 2014).
  - ✓ Pre procesamiento: Proceso en el cual se hace una eliminación del ruido presente en una imagen a fin de obtener imágenes con un buen nivel de detalle (Crespín Luis & Julián García, 2014).

- ✓ Segmentación: Etapa en la cual se realizan un conjunto de actividades con el fin de extraer uno o varios objetos de interés dentro de una imagen (Crespín Luis & Julián García, 2014).
- ✓ Descripción: Proceso relacionado con el computo de características útiles para diferenciar un tipo de objeto de otro (Crespín Luis & Julián García, 2014).
- ✓ Reconocimiento: Proceso en el cual se lleva a cabo la identificación de objetos (Crespín Luis & Julián García, 2014).
- ✓ Interpretación: Etapa en la cual se otorga un significado a un grupo de objetos reconocidos (Crespín Luis & Julián García, 2014).

Pese a que es una rama de estudio relativamente nueva, ésta ha avanzado a pasos agigantados, utilizándose en la actualidad en diversos campos tecnológicos como la robótica, la industria y la medicina. Si bien esta tecnología es relativamente cara, sus precios han venido reduciéndose conforme su popularidad y demanda han ido aumentando (Crespín Luis & Julián García, 2014).

2.1.4 Conclusión

Expuestos los conceptos mostrados líneas arriba, le resultará más comprensible al lector el contexto propio de la problemática del presente proyecto de fin de carrera, asimismo se hace más entendible las razones que motivan el desarrollo de esta. Una vez ya familiarizado el lector con los conceptos, será capaz de hacer un mejor seguimiento a las siguientes etapas del proyecto y las herramientas tecnológicas empleadas para el desarrollo de éste.

3. ESTADO DEL ARTE

3.1 Introducción

En la actualidad existen diversos estudios enfocados en el reconocimiento de dibujos a mano alzada; sin embargo; dada la complejidad que implica el desarrollo de estas, algunas son de carácter experimental o de prueba. Para el estado del arte del presente proyecto únicamente se consideraron aquellos cuyos aportes son significativos para el desarrollo de tecnologías de recuperación de imágenes basadas en el reconocimiento de trazos hechos por el usuario.

3.2 Objetivos de la revisión del estado del arte

El objetivo principal del estado del arte expuesto a continuación es hacer de conocimiento del lector los trabajos de investigación más importantes en el área de reconocimiento de trazos a mano alzada hasta la fecha. Estos programas en la mayoría de casos no solo abarcan la detección de esbozos; sin embargo, es pertinente hacer una investigación sobre las limitantes a las que se tuvieron que enfrentar para el desarrollo de estos con el fin de determinar correctamente el alcance que tendrá el proyecto de fin de carrera.

El proceso de investigación del estado del arte por parte del autor se realizó mediante una revisión de la literatura no sistemática presente en la web. Las búsquedas fueron realizadas durante los meses de Marzo y Abril del año 2016 utilizando el navegador de Google Chrome así como la plataforma de Google Scholar para la obtención de documentos académicos (*papers)* en los que se documente con mayor detalle los productos expuestos más adelante.

Concretamente, las búsquedas de los productos experimentales (no comerciales) fueron hechas en *Google Scholar* para la adquisición y lectura de artículos académicos (*papers*) donde se describiesen los procesos de desarrollo de prototipos de recuperación de imágenes similares al proyecto de fin de carrera. La información de aquellos productos comerciales, por su parte, fueron obtenidos de las páginas webs pertenecientes a estos.

3.3 Aplicaciones comerciales/disponibles:

3.3.1 *MindFinder*:

Es un motor de búsqueda de imágenes basado en dibujos a mano alzada desarrollado en el año 2010 por el equipo de investigación de Microsoft con sede en Asia. Esta aplicación permite la búsqueda de imágenes a través de un componente de reconocimiento de trazos basado en curvas, pero se apoya en el uso de pequeños cuadros de texto (a modo de etiquetas) que le

sirven como herramientas auxiliares en caso de que se tenga que realizar búsquedas demasiado complejas. Dado que el color es una característica infaltable en la mayoría de las imágenes, este motor de búsqueda incluye también un pequeño icono circular el cual permite indicar al usuario los colores más dominantes (máximo tres) de la imagen que se está buscando. Si bien el algoritmo de identificación de trazos utilizado por *MindFinder* permite obtener imágenes con resultados medianamente aceptables (Imagen 4), su mayor logro radica en su capacidad para realizar comparaciones entre un dibujo y millones de imágenes de forma rápida (Cao et al., 2010).

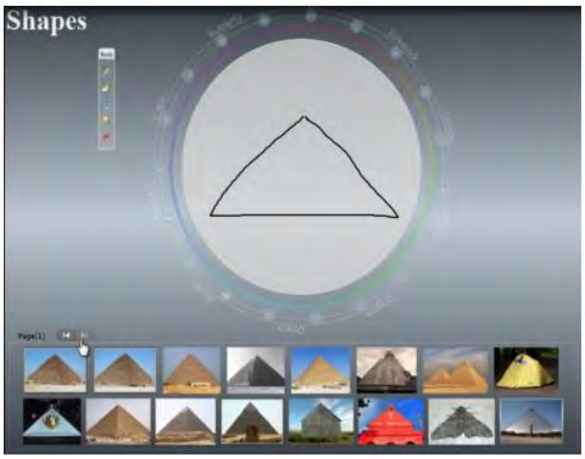


Imagen 6: Interfaz de *MindFinder* (imagen recuperada del URL: http://research.microsoft.com/en-us/projects/mindfinder/).

3.3.2 *Sketch2Photo: Internet Image Montage*

Es un sistema (desarrollado en conjunto por la *National Basic Research Project of China* y la *National High Technology Research and Development Program of China* en el año 2009) capaz de componer imágenes de tono realista utilizando como herramientas los dibujos hechos a mano alzada y etiquetas de texto ingresadas por el usuario (Imagen 5). El sistema realiza las búsquedas de las imágenes en Internet valiéndose de las etiquetas de texto y, mediante un mecanismo de filtrado, selecciona las imágenes que mejor se ajusten a lo dibujado por el usuario. La imagen final devuelta por el sistema es generada a través de la unión de distintas fotografías de individuos u objetos que presenten una silueta parecida a los esbozos realizados (T. Chen, 2016).

Con la finalidad de reducir los rastros de que las imágenes han sido transpuestas, Sketch2Photo hace uso de un novedoso algoritmo de su autoría que permite eliminar los bordes de estas; consiguiendo dar la apariencia de que la imagen es verídica. No obstante, entre sus principales defectos está su alto índice de fallos cuando se decide realizar montajes tomando como escenario (*background*) imágenes de ambientes cerrados y/o cuando se dibujan demasiados

elementos a colocar en el escenario, dificultades para trabajar con imágenes con muchos individuos u objetos en ella, incapacidad para tratar a una misma escala de tamaño los elementos ingresados en un montaje, incapacidad de la aplicación para recuperarse de forma automática ante un fallo y finalmente su mayor desventaja radica en su tiempo de ejecución, el cual es de 15n + 5 minutos, donde n es el número de siluetas o contornos dibujados por el usuario (Tao Chen, 2009).

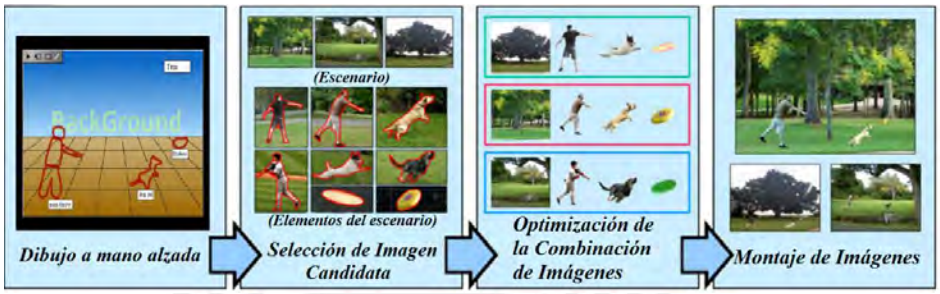


Imagen 7: Representación del funcionamiento de *Sketch2Photo* (imagen adaptada del URL: http://cg.cs.tsinghua.edu.cn/montage/main.htm)

3.3.3 *Retrievr*:

Es una página web de tipo experimental lanzada en el año 2005 que permite realizar búsquedas de imágenes mediante el ingreso, de forma local, de una imagen similar a la que se desea obtener (búsqueda por imagen) o de un dibujo cromático hecho a mano alzada por el usuario. Esta aplicación, desarrollada completamente en lenguaje Python, está basado en un trabajo de investigación hecho en 1995 dentro de la Universidad de Washington (Langreiter, 2016). A diferencia de otros productos, *Retrievr* es incapaz de devolver como resultado de búsqueda una imagen con características exactas al dibujo realizado por el usuario ya que no apoya su funcionamiento en el reconocimiento de objetos, rostros o texto; sino más bien, hace comparaciones demasiado simples entre las imágenes almacenadas en la página web de Flickr.com y los dibujos hechos por el usuario. Estos últimos son creados empleando una reducida paleta de colores junto a un pequeño lienzo rectangular en la parte izquierda de la aplicación para el ingreso de trazos de la forma y color que el usuario desee, resultando al final en un dibujo con apariencia de haber sido hecho con acuarelas (Imagen 6). Debido a la poca cantidad de colores disponibles y al solo uso de las formas más pronunciadas de los trazos en el lienzo, sus resultados (mostrados a la derecha del cuadro para dibujar y la paleta de colores) suelen ser inexactos; por lo que *Retrievr* no es considerado un buscador de imágenes serio (LK, 2016).

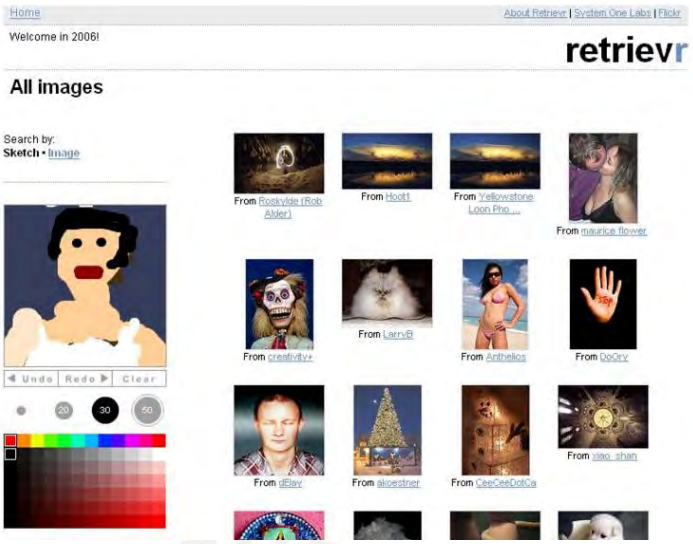


Imagen 8: Funcionamiento de la herramienta *Retrievr* (imagen sacada del URL: https://techjourney.net/search-online-photos-and-images-from-flickr-by-drawing-and-sketching-with-retrievr/)

3.3.4 *Sketch-a-Net:*

Programa informático capaz de identificar dibujos a mano alzada en tiempo real de forma más rápida que un humano. Según científicos responsables de este proyecto, esta aplicación es capaz de identificar (en tiempo real) un dibujo correctamente el 74,9% de las veces a diferencia de los humanos cuyo porcentaje de éxito es del 73,1% (López, 2016).

Esta aplicación parte bajo la premisa de que un dibujo es un conjunto de trazos hechos por el hombre en un determinado orden; es decir, si bien existe una diversidad de estrategias para dibujar un determinado objeto en términos de qué parte de este será dibujado primero, suele ser común que se comience dibujando el contorno principal del este seguido por sus detalles. Sketch-a-Net propone un modelamiento simple pero efectivo apoyado en el uso de redes neuronales artificiales el cual consiste en separar los trazos en tres grupos conforme se va haciendo el dibujo y tratarlos como canales independientes. Mediante la creación de estas subdivisiones (Imagen 7) es posible luego realizar una gran cantidad de combinaciones entre estos distintos grupos de trazos y con ello realizar mejores predicciones sobre lo que se está dibujando en ese momento (Q. Yu, 2015).

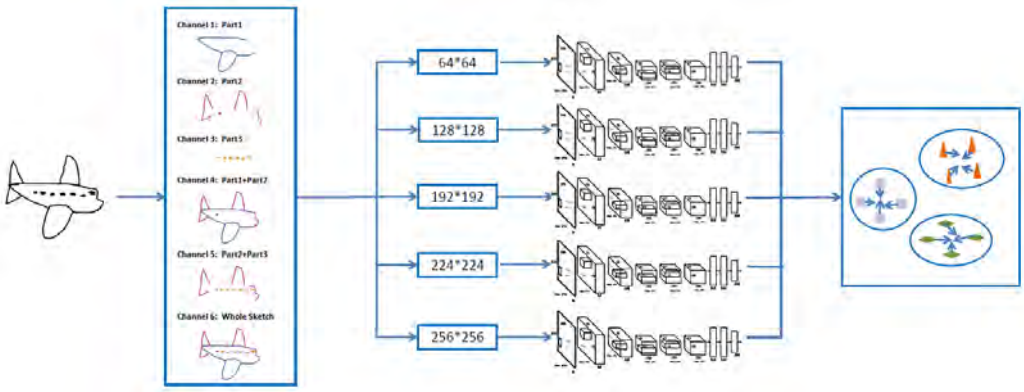


Imagen 9: Esquema del funcionamiento de la división por canales por *Sketch-a-Net* (Q. Yu, 2015).

3.3.5 *GazoPa: Similar Image Search*

Es un programa informático capaz de realizar búsquedas mediante un algoritmo que permite obtener todo tipo de información y deducciones a partir de una imagen como color de cabello, de piel, formas de la imagen, entre otros. Una vez obtenidos estos datos, pasan a ser procesados y comparados con una base de datos de más de 50 millones de imágenes. Además, *GazoPa* permite analizar imágenes congeladas de un video y buscar en la web otros videos que guarden relación con este. *GazoPa* dispone también de una herramienta de dibujo propia, basada en Flash, para realizar bocetos rápidos y pasarlos por el buscador con la finalidad de obtener resultados que se asemejen a lo ingresado (Imagen 8). Finalmente, esta aplicación cuenta con una herramienta para el ingreso de texto a modo de ayuda para cuando se tenga que realizar búsquedas demasiado complejas. Lamentablemente, desde el 2011 *GazoPa* ya no se encuentra disponible en la web debido a que decidió orientar su servicio exclusivamente al mercado Chino B2B (*Business to business*) (GazoPa Team, 2016).

Imagen 10: Interfaz de la aplicación *GazoPa* (imagen sacada del URL: http://www.gazopa.com/).

3.4 Cuadro Comparativo:

Tabla 3: Cuadro Comparativo entre los distintos productos de reconocimiento de trazos a mano alzada.

Características	*Retrievr*	*MindFinder*	*Sketch2Photo*	*Sketch-a-Net*	*GazoPa*
Uso de texto para búsquedas	Sí	Sí	Sí	No	Sí
Tiempo de ejecución demasiado largo	No	No	Sí	No	No
Realiza búsquedas en la web	Sí, aunque solo en Flickr.com	No, emplea una base de datos de imágenes	Sí	No	No, emplea una base de datos de imágenes
Reconocimiento de patrones en las curvas	No	Sí, aunque moderado	Sí	Sí	Sí
Detección de bocetos cromáticos	Sí, aunque moderadamente	Sí	No, solo busca siluetas	No	Sí
Tiene problemas de escalabilidad en la Base de Datos	No	No	Sí	No	No
Detección de Color	Sí	Sí	No, solo trabaja con siluetas	No	Sí
Uso de base de datos de imágenes	No	Sí	No	No	Sí

3.5 Conclusiones sobre el Estado del Arte

Los productos mostrados en el estado del arte son una muestra de la diversidad de herramientas asociadas a la recuperación de imágenes por contenido existente, y permiten evidenciar la existencia de ciertos problemas como el manejo ineficiente de cantidades masivas de imágenes; razón por la cual muchos de los productos optan por manejar bases de datos de imágenes en lugar de interactuar de forma directa con la infinidad de imágenes disponibles en la web.

Además, se observan ciertas dificultades relacionadas con el alto tiempo de ejecución (costo computacional) que conlleva el procesamiento de recuperación de imágenes por contenido y la persistente dependencia del uso de texto para la realización de búsquedas complejas, lo cual denota una deficiencia en los componentes de detección de dibujos a mano alzada en la actualidad. Estas limitantes son solo algunas de las encontradas en cada uno de los productos, razón por la cual es necesario el desarrollo de una mejor herramienta de recuperación de imágenes que sea capaz de sobreponerse a las mencionadas falencias mientras abre el camino hacia la creación de nuevas utilidades a esta tecnología.

4. MODELO DE REPRESENTACIÓN DE IMÁGENES POR BOLSA DE CARACTERÍSTICAS

4.1 Introducción

La presente sección del documento tiene como finalidad ir describiendo de forma ordenada como se fue cumpliendo cada objetivo mediante la puesta en marcha de un conjunto de procedimientos para desarrollar un *framework* de recuperación de imágenes a partir del ingreso de dibujos a mano alzada. Para un mayor entendimiento de los pasos ejecutados se irá describiendo de forma secuencial las actividades realizadas, haciendo hincapié en cada uno de los objetivos específicos conforme éstos se vayan alcanzando.

4.2 Transformación de imágenes en esbozos

Esta etapa consiste en hacer un adecuado acotamiento del conjunto de imágenes con las que se trabajará, para luego pasar a procesar cada una de éstas mediante el uso de técnicas y herramientas descritas a continuación.

- **Selección, pre procesamiento y transformación de imágenes en siluetas**

  Dado que el proyecto a desarrollar trabajará con imágenes, es primordial que las imágenes con las que se trabajen sean de la mejor calidad posible; es decir, que presenten la mínima distorsión y/o ruido visual. En caso contrario, se dificultará la posterior identificación de trazos por parte del *framework* de recuperación de imágenes. Por tal motivo, lo primero que se realizó fue consolidar una base de datos de imágenes extraídas de un banco de imágenes a color disponible en la web de forma gratuita como Imagenet.com. A través de este sitio web se realizó una descarga masiva de imágenes para luego hacer una selección manual de aquellas que contuviesen la mayor nitidez y tamaño posible (ver imagen 11).

Imagen 11: Comparativa imagen con y sin ruido digital (imagen obtenida del url: http://www.digitalfotored.com/imagendigital/ruidodigital.html)

La selección de las mencionadas imágenes fue hecha de forma no automatizada y con la finalidad de que respondan a un conjunto de categorías previamente establecidas. Se buscó obtener la mayor cantidad posible de imágenes, siendo ésta de más de 6500. Las categorías sobre las cuales se agruparon las imágenes fueron las siguientes:

Tabla 4: Listado de las categorías de imágenes a color empleadas en el proyecto.

Categoría	Nro. De imágenes
Diseños de banderas de países	196
Modelos de autos	663
Diseño de collares	1315
Tipos de flores silvestres	242
Frutas comestibles	744
Modelos de jarrones	749
Fotos de paraguas	759
Fotografías de patos	826
Diseños de pelotas	518
Rostros de personas	594

Para el desarrollo del proyecto se creó localmente una carpeta llamada "*BDimagenesOriginales*" la cual actuase como una base de datos categorizada de imágenes a color que a su vez contenga dentro de sí 10 carpetas: una por cada categoría de imagen a color mencionadas en la Tabla 4. Finalmente, ya dentro de cada una de éstas carpetas se almacenaron las imágenes a color según la categoría a la que pertenece. Una vez consolidado el repositorio local de imágenes con las que se trabajará, se prosiguió a procesar cada una de éstas mediante el uso de la herramienta *cv2.Canny (image, lower, upper)* de OpenCV. Éste algoritmo se basa en la premisa de que las siluetas presentes en una imagen suelen tener mayor intensidad en cada uno de los pixeles que la conforman y éstos están enlazados uno a continuación de otro; por ello, la herramienta de Canny recibe como parámetros la imagen que se desea transformar a silueta (*image*) junto con los valores mínimos (*lower)* y máximos (*upper*) entre los cuales deberán estar las intensidades de los gradientes de cada uno de los pixeles de la imagen para poder ser considerada como un borde o silueta válido.

Imagen 12: Imagen original y silueta de ésta mediante el algoritmo de Canny (imagen de autoría propia).

Cabe señalar que ésta herramienta de OpenCV realiza además un pre procesamiento de la imagen, con lo cual elimina cualquier tipo de ruido visual que pudiese existir en la imagen. Este paso previo resulta útil para el desarrollo del proyecto puesto que realiza un filtro adicional al realizado inicialmente de forma manual y, por consiguiente, disminuye considerablemente la probabilidad de trabajar con imágenes defectuosas.

Debido a que se pensaba trabajar con una gran cantidad de imágenes y el rango de los valores mínimos y máximos a usarse en el algoritmo de Canny variaba de acuerdo al grado de complejidad que tenía cada imagen en sus colores, se optó por automatizar la obtención de los mencionados parámetros. Por este motivo inicialmente se decidió hacer uso de un artificio, el cual consiste en transformar los colores de la imagen original a escala de grises y luego calcular la intensidad promedio de los gradientes de cada uno de los pixeles pertenecientes a esta. Dicha técnica aplicada a las imágenes en su totalidad permitió delimitar un rango bajo el cual aquellos pixeles que se encontraban dentro de éste eran considerados como siluetas dentro de una imagen. De esta forma, mediante la automatización del algoritmo de detección de bordes de Canny, se procedió a transformar en silueta (de forma iterativa) cada una de las imágenes presentes en el repositorio local de modo que diesen la apariencia de haber sido dibujadas usando trazos a mano alzada (ver imagen 12). En el caso del proyecto, con la finalidad de mantener un orden en el manejo de las imágenes, se creó otra carpeta llamada "*BDimagenesSiluetizadas*", de estructura similar a la carpeta "BDimagenesOriginales" la cual contuviese a todas las imágenes debidamente categorizadas y transformadas en siluetas mediante el algoritmo de Canny (ver imagen 13).

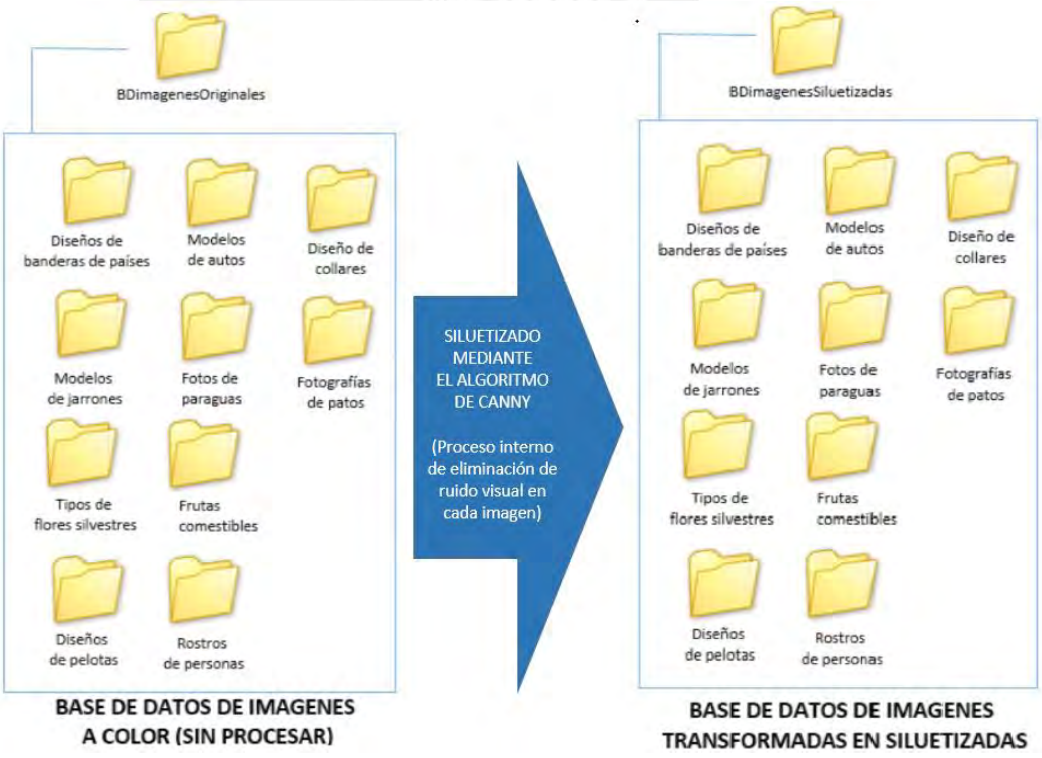


Imagen 13: Estructura de almacenamiento de las imágenes originales y sus respectivas transformaciones en siluetas a través del algoritmo de Canny (imagen de autoría propia).

Dado el gran volumen de imágenes a color en la base de datos de imágenes, se decidió uniformizar a cada una de éstas nombrándolas con un valor numérico que va desde el número 1 hasta la cantidad total de imágenes presentes en la base de datos: 6605. Así mismo, se ordenaron las carpetas de forma alfabética según el nombre de su categoría y se estableció la dimensión 1111x1111 como estándar para cada una de las imágenes en la base de datos de imágenes transformadas en siluetas. De esta forma resultó factible realizar un algoritmo que leyera cada una de las imágenes y las fuese procesando para posteriormente trabajar con éstas. Siendo el diagrama de flujo del algoritmo mencionado el siguiente:

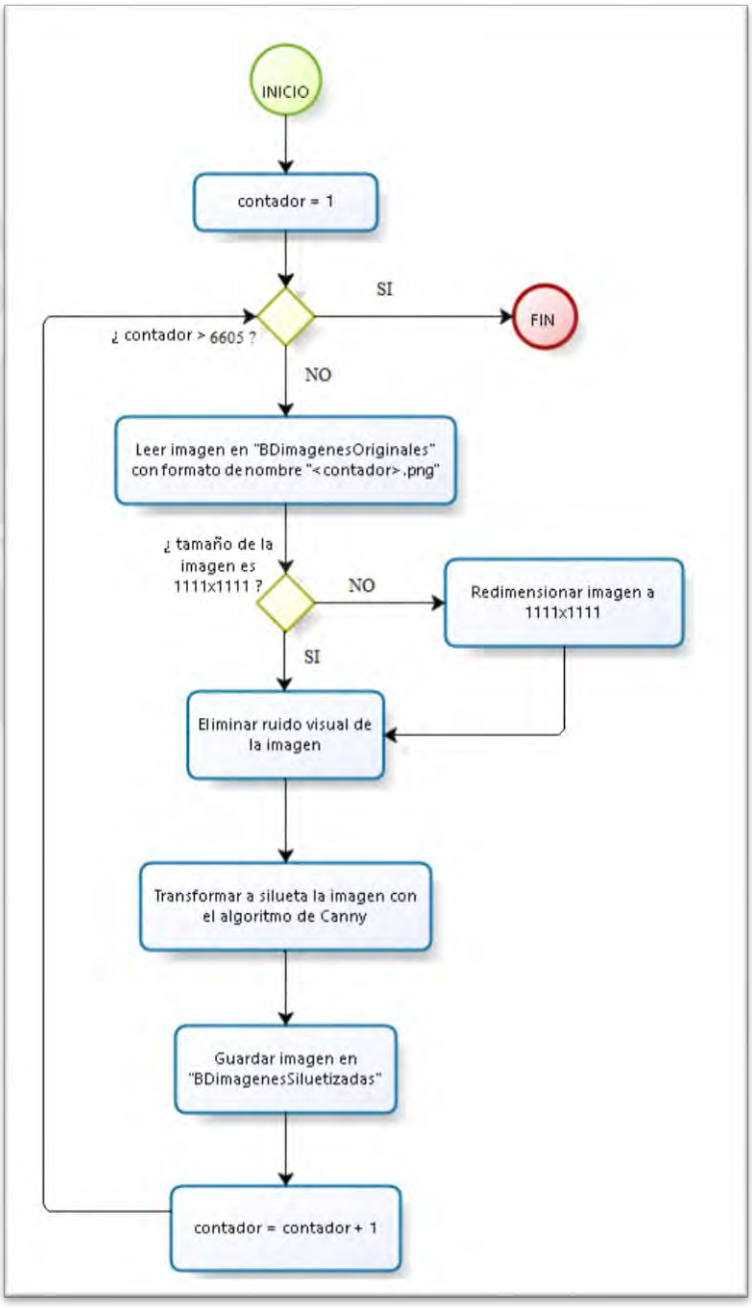


Imagen 14: Flujo del proceso de lectura, transformación a silueta y guardado de las imágenes a color procesadas (imagen de autoría propia).

Lamentablemente, el uso de este artificio no resultó del todo eficiente puesto que en la mayoría de imágenes (sobre todo en aquellas con una amplia variedad de tonalidades de colores) el algoritmo de Canny no conseguía transformar a silueta únicamente los objetos presentes en la imagen, sino que también incluía parte del fondo y para revertir esto era necesario procesar de forma individual cada imagen, ajustando manualmente los valores mínimos y máximos bajo el cual Canny considera cada pixel como silueta o no (ver imagen 15).

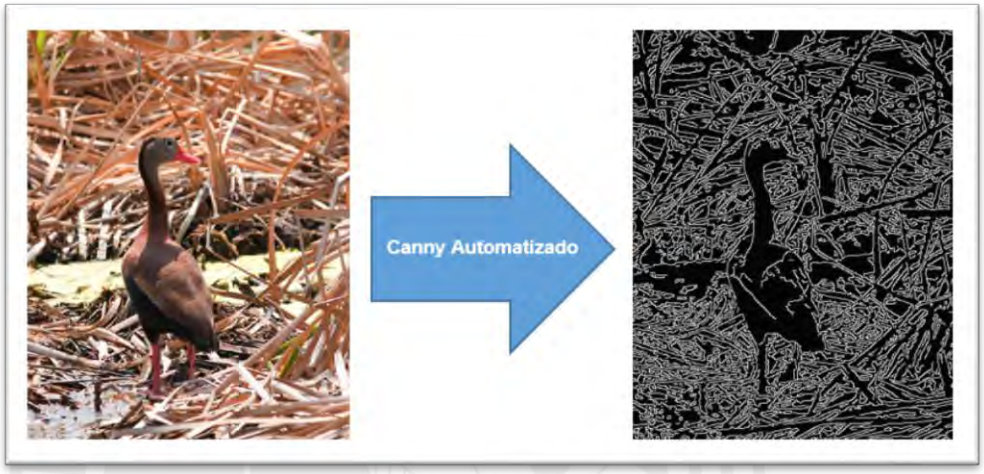


Imagen 15: Ejemplificación del problema de transformación masiva a silueta de imágenes mediante el algoritmo de Canny. (Imagen extraída del url: http://www.image-net.org/).

Esto último fue rechazado como alternativa de solución, puesto que se pensaba manejar una gran cantidad de imágenes y procesarlas una por una resultaba inviable. Por tal motivo, se decidió agregar una base de datos de dibujos a mano alzada que consiguiesen simular imágenes transformadas en siluetas en lugar de aquellas imágenes a color difíciles de transformar a siluetas.

- **Adición al *framework* de un banco de imágenes de dibujos hechos a mano alzada**
  Ya con el banco inicial de imágenes procesado, se cuenta con un repositorio de siluetas de imágenes aparte del banco de imágenes a color. No obstante, para efectos del proyecto se deseaba que el *framework* a desarrollar esté familiarizado con las características de los dibujos hechos por los usuarios (*sketches*), por lo que resultaba necesario que éste también incluyese dibujos hechos por usuarios dentro del repositorio de imágenes a modo de entrenamiento y, si el usuario final lo desea, como imagen final a recuperar: que se permita también la recuperación de un dibujo a mano alzada mediante el ingreso de otro dibujo del mismo tipo.

Un aspecto destacable que tiene este conjunto de imágenes es que al ser completamente trazos no necesitan ser convertidos a siluetas, puesto que de por sí ya lo son. Para obtener estos dibujos se accedió a una base de datos gratuita de dibujos a mano alzada disponible en la web (ver imagen 16), la cual maneja imágenes de tamaños considerablemente grandes y libres de ruidos visuales (Hildebrand, Eitz, Boubekeur, & Alexa, 2011).

Este conjunto de imágenes presenta bocetos de diversos tipos obtenidos como resultado de un trabajo de investigación previo en el que se solicitó a numerosas personas de forma aleatoria que realizasen dibujos asociados a varias categorías de imágenes (ver categorías en el ANEXO). Esta base de datos de dibujos consta de 250 categorías, cada una conformada por 80 dibujos; obteniéndose aproximadamente 20'000 imágenes de dibujos a mano alzada útiles tanto para el entrenamiento del *framework* de recuperación de imágenes como para el incremento de la cantidad de imágenes a manejar por parte de éste (Eitz, Hays, & Alexa, 2016).

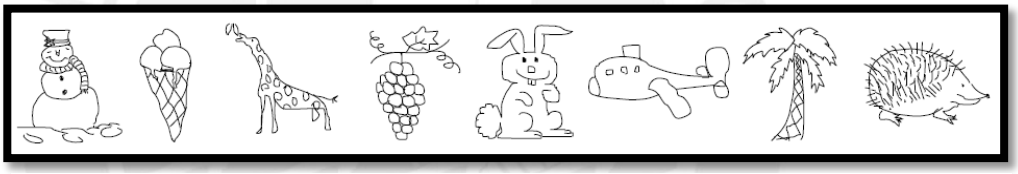


Imagen 16: Ejemplos de las imágenes pertenecientes al repositorio local de dibujos hechos a mano alzada (imagen extraída del url: http://cybertron.cg.tu-berlin.de/eitz/projects/classifysketch/).

Para efectos prácticos del proyecto se creó una carpeta llamada "*Sketches*" a modo de repositorio local de estructura similar a las carpetas "BDimagenesOriginales" y "BDimagenesSiluetizadas" en donde se almacene cada uno de los dibujos en carpetas según su categoría. Estas imágenes también fueron uniformizadas de forma similar a las otras bases de datos de imágenes previamente mencionadas, nombrando cada dibujo con un número de forma secuencial en cada categoría; teniéndose por ejemplo las imágenes con nombres que van del "1" al "80" en la categoría "*airplane"*, del "81" al "160" en la categoría "*alarm clock"*, del "161" al "240" en la categoría "*angel"* y así sucesivamente según el orden alfabético de las 250 categorías de estos dibujos (ver nombres de las categorías en el ANEXO).

- **Pre Procesamiento de los dibujos a mano alzada ingresados por el usuario final**

  No solamente se realiza un pre procesamiento de las imágenes presentes en el banco de imágenes del *framework*, los dibujos que el usuario final ingrese también pasarán por un proceso similar de eliminación del ruido visual. Con esto se busca prevenir que se generen situaciones de error inducidas por el usuario final voluntaria o involuntariamente, como el pintado o coloreado de siluetas. Para ello se someterá siempre a las imágenes ingresadas por el usuario a un proceso de adelgazamiento o esqueletización de sus trazos. De esta forma la aplicación se asegura de que únicamente se trabajen con trazos de un pixel de grosor (ver imagen 17).

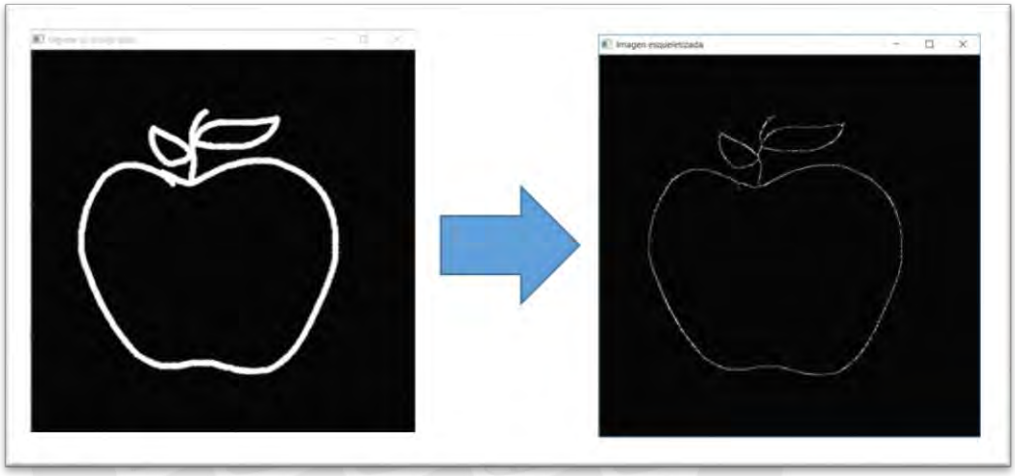


Imagen 17: Siluetizado del dibujo ingresado por el usuario final, obteniéndose una imagen con trazos más delgados (imagen de autoría propia).

Para llevar a cabo la esqueletización del dibujo ingresado por el usuario, se tomó como base el uso de las herramientas incluidas en la librería de OpenCV, creándose inicialmente para ello una imagen en escala de grises completamente de color negro y de dimensiones 800x800 la cual luego fue colocada dentro de una ventana generada por otra función de OpenCV. Ya creada la ventana, a través de la función *cv2.setMouseCallback* de OpenCV, se pasó a permitir el ingreso de trazos en el mencionado fondo negro a través del mouse. Lo que hace esta función es dibujar una figura geométrica básica por cada *click* que se realice con el mouse y/o dibujar figuras geométricas idénticas de forma consecutiva si se arrastra el mouse.

Para realizar tal acción la función *cv2.setMouseCallback* recibe como argumento la llamada a una función que, también apoyada en otras herramientas de OpenCV, dibuja una figura geométrica de determinado tamaño el cual para efectos del presente proyecto se optó por que sea de color blanco y del menor tamaño posible. Dado que

un trazo o (siendo más exactos) una recta es básicamente la concatenación de varios puntos, la figura geométrica elegida a dibujar con cada *click* que haga el usuario fue la de una circunferencia con el menor radio posible. Líneas arriba se indicó que se trabajaría con unas dimensiones estándar para las imágenes, siendo éstas de 1111x1111. No obstante la ventana habilitada para que el usuario ingrese su dibujo a mano alzada fue de 800x800, esta medida se debió a que el uso de las dimensiones estándar mencionadas resultaban demasiado grandes para poderse visualizar a través de la pantalla del computador bajo el cual se desarrolló el proyecto de fin de carrera; razón por la cual se optó por redimensionar el dibujo ingresado a la dimensión de 1111x1111 una vez el usuario hubiese terminado de dibujarlo para posteriormente ser esqueletizado mediante el uso de la librería de OpenCV (ver imagen 14). Además, cabe señalar que la ventana en la cual el usuario ingresa su dibujo fue programada de manera que una vez terminado el ingreso de los trazos blancos respectivos en el lienzo color negro, al presionar la tecla ESC (Escape) comience el proceso redimensionado, esqueletizado, inversión de colores y demás procesos asociados a la recuperación de imágenes similares a la dibujada (ver imagen 18), los cuales se describirán detalladamente a lo largo del presente documento.

Ya definidos los pasos a seguir para el pre procesamiento del dibujo ingresado por el usuario, se decidió por comodidad hacer el procesamiento de imágenes pensando en que estas tienen fondo blanco y trazos/siluetas de color negro y únicamente utilizar el formato inicial de imagen con fondo negro y trazos blancos al momento de solicitar al usuario que ingrese su dibujo a mano alzada. Esto también resultaba útil para indicar en todo momento al usuario que se encuentra en la etapa de ingreso del dibujo a mano alzada dentro del *framework.* Para realizar esta inversión en los colores del dibujo ingresado se realizó un paso adicional el cual consiste en que, luego de haber sido esqueletizado el dibujo ingresado por el usuario, se procede a invertir los colores de éste nuevamente a través del uso de herramientas de OpenCV.

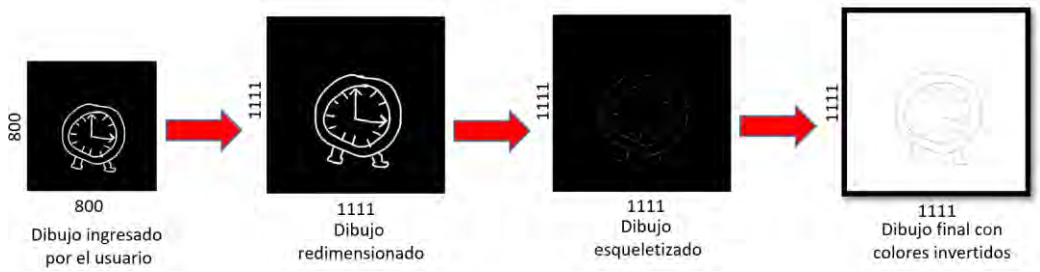


Imagen 18: Flujo del pre procesamiento de los dibujos a mano alzada ingresados por el usuario (imagen de autoría propia).

Invertir los colores en una imagen no es demasiado complejo y menos aún si se hace uso de las herramientas de OpenCV dedicadas al procesamiento de imágenes. Basta con que se aplique una malla de negación a cada uno de los pixeles presentes en la imagen binaria, con esto un pixel cambiará su valor de uno a cero y viceversa. Con los pasos descritos y debidamente realizados hasta el momento, es correcto afirmar que se logra cumplir con el primer objetivo específico del proyecto, el cual es **"Suprimir toda falta de nitidez presente en las imágenes que dificulte la posterior identificación y obtención de rasgos característicos mediante el procesamiento de éstas"**. Lo que corresponde ahora es ver cómo se extraerán y representarán las características de cada una de las imágenes.

4.3 Representación y extracción de características

- **Extracción de características locales**

Ya consolidada la base de datos de imágenes se procedió con la extracción de características locales en las imágenes, estos son trozos o regiones medianamente pequeñas comparadas con la imagen total cuyas ubicaciones son elegidas de forma aleatoria. A continuación se mencionarán las pautas seguidas para el desarrollo del *framework* relacionadas a la extracción de características locales en la base de datos de imágenes.

Puesto que ya se tenía estandarizadas las dimensiones de todas las imágenes con las que el *framework* trabajará, es correcto afirmar que tanto el ancho como el alto de las características locales tuvieron tamaños uniformes. Se tomaron al azar 10 características locales con tamaño equivalente al 25 por ciento de la diagonal de la imagen de origen a fin de que se trabajen con trozos de imágenes lo suficientemente grandes como para abarcar en su contenido formas de siluetas que se presten a ser posteriormente clasificadas (ver imagen 19). Cabe señalar que estas medidas fueron elegidas en base a revisiones hechas a trabajos relacionados con SBIR, en los cuales dichas dimensiones permitieron la obtención de resultados adecuados (Hildebrand, Eitz, Boubekeur, & Alexa, 2011).

Si bien en la bibliografía revisada se recomendaba utilizar el 25 por ciento de la diagonal de la imagen original como medida de ancho y alto de las características locales con la finalidad de obtener resultados óptimos, en éstas también se enfatizaba en utilizar un total de 500 características locales las cuales durante el desarrollo del

proyecto se descubrió que tenían un fuerte impacto negativo en el tiempo de ejecución del *framework* a menos que se contase con una computadora de alta capacidad de procesamiento, lo cual no aplicaba para el presente proyecto de fin de carrera.

Ante este escenario se buscó mejorar esto a través de un análisis del funcionamiento y utilidad de las 500 características locales por cada imagen. Encontrándose que si cada característica local tiene como ancho y alto el valor equivalente al 25 por ciento del tamaño de la diagonal de la imagen completa de dimensiones 1111x1111, cada característica local tiene una dimensión aproximada de 393x393 pixeles; necesitándose 8 características locales (en el mejor caso en que ninguna de estas se traslapen entre sí) para cubrir la imagen completa de la cual provienen. No obstante, cuando se representó lo descrito de forma gráfica (ver imagen 19) resultó no ser tan sencillo como se imaginaba puesto que, dadas las dimensiones de las características locales, estas tienen una alta probabilidad de traslaparse una sobre otra y/o que una abarque determinada cantidad de área cubierta por otra, pudiéndose inclusive no cubrir áreas de la imagen las cuales podrían contener características valiosas para su identificación. En la imagen 19 se puede apreciar el caso más favorable y a la vez improbable posible, en este las características locales 1, 2, 4 y 5 no se traslapan entre sí y aportan información válida sobre diferentes regiones de la imagen. Sin embargo, esto no ocurre con las características locales 3, 6, 7 y 8; las cuales obligatoriamente van a tener que traslaparse con otras características locales puesto que exceden las dimensiones de la imagen, pudiendo ubicarse sobre áreas ya registradas por otras características locales.

Según la experimentación realizada en el proyecto se pudo constatar que, dado que las características locales se ubican dentro de la imagen de forma aleatoria, en muchas ocasiones estas se traslapaban una sobre otra dejando de lado regiones con trazos útiles para la identificación de la imagen de la cual provienen. Ante estos resultados obtenidos se decidió aumentar de 8 a 10 el número de características locales, de modo que se consiguiese una mejora considerable en la obtención de características de las imágenes.

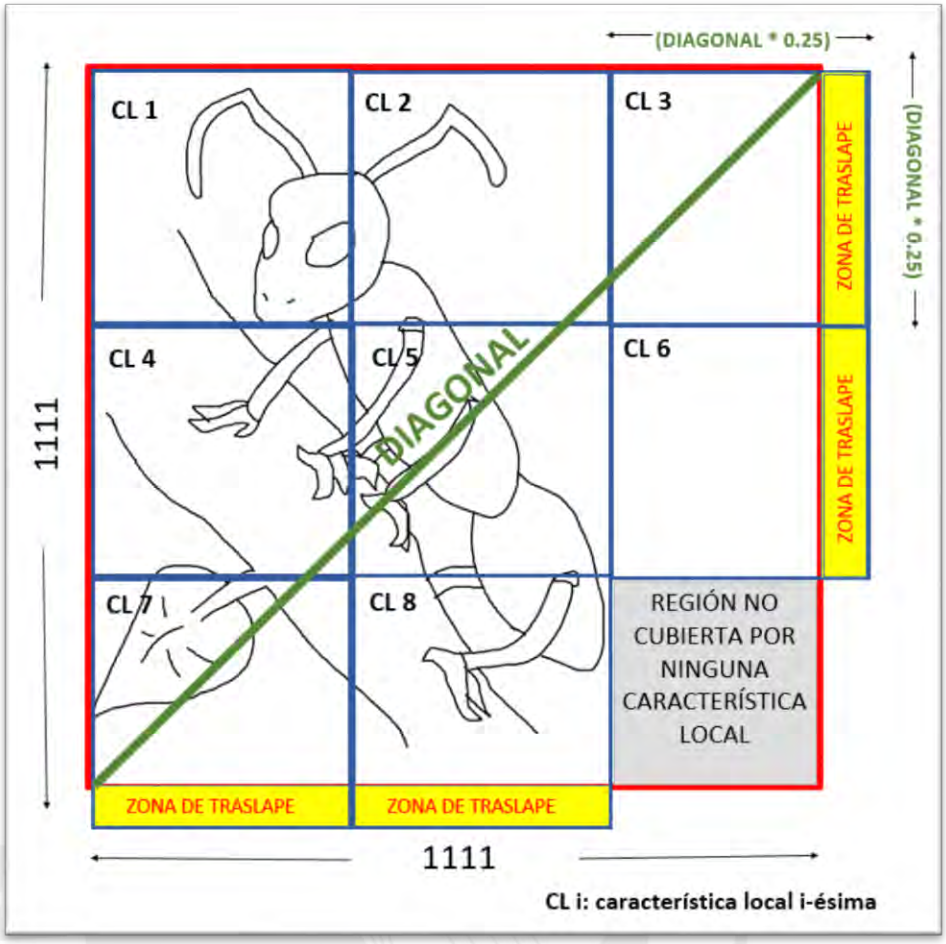


Imagen 19: Representación gráfica de las características locales en una imagen y los posibles traslapes que este conlleva (imagen de autoría propia).

Un aspecto importante tomado en cuenta fue que, al momento de seleccionarse aleatoriamente los trozos o regiones de las imágenes, fue necesario verificar que no se seleccionen aquellas que contuviesen pocos trazos comparados con la totalidad de la imagen seccionada. Otro, considerado como el peor caso, es que se seleccione aleatoriamente una región que no contiene trazo alguno: una porción completamente en blanco (ver imagen 20). En ambos casos el aporte obtenido sería casi nulo debido a la ausencia de pixeles descriptivos de un trazo o silueta, por lo que no serán considerados como una característica local válida en el proceso de representación y extracción de características.

Por todo lo expuesto, el *framework* se desarrolló de tal forma que validase que cada una de las características locales que se extraigan al azar de una imagen contenga una cantidad aceptable de trazos en su interior. Para asegurar esto último, se realizó una validación de que la proporción del número de pixeles color negro (RGB = 0) respecto

a la totalidad de píxeles en dicha característica local sea mayor o igual al 2%; en caso contrario, se generaría aleatoriamente otra región y volvería a ser validada una y otra vez hasta que se cumpla la condición mencionada. Cabe señalar que la decisión de tomar el valor de 2% como umbral mínimo para la aceptación de características locales obedece a un proceso de experimentación hecho por el autor del presente proyecto, en este se fue evaluando con diferentes valores hasta encontrarse uno que garantice que la característica local generada tuviese una considerable cantidad de trazos.

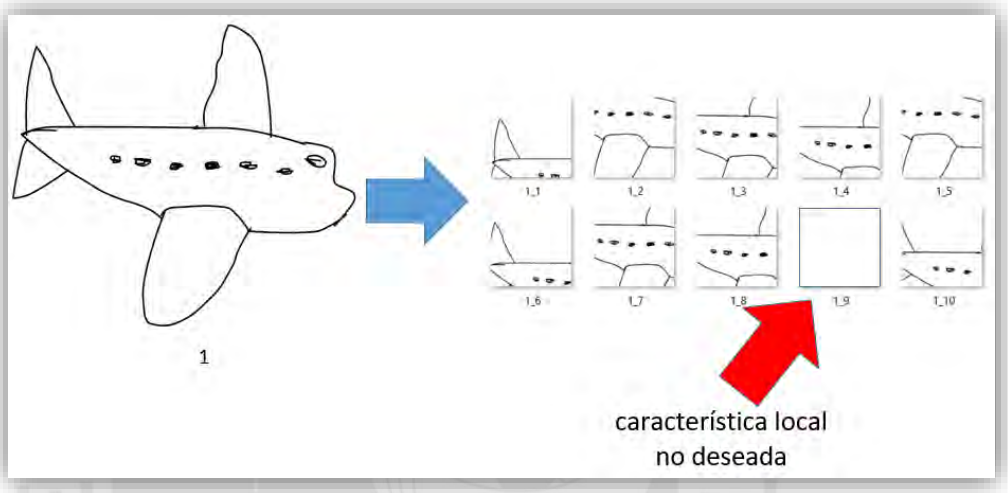


Imagen 20: Ejemplificación de la obtención de las características locales (imagen de autoría propia).

Otra forma de verificación más compleja que se decidió emplear en el proyecto fue verificar el valor promedio de los pixeles de cada característica local aleatoria a fin de que dicho valor se encuentre dentro de un rango previamente establecido por el autor. Debido a que se trabaja únicamente con imágenes en blanco y negro, los pixeles de cada una de las características locales solo contienen dos posibles valores numéricos RGB los cuales son (0, 0, 0) para representar los colores negros y (255, 255, 255) para los colores blancos. Dicho esto, mientras más pixeles sean de color negro, se tendrá una mayor cantidad de valores asociados con el valor cero, ocasionándose una reducción en la proporción del número de pixeles blancos respecto a la cantidad total de pixeles presentes en dicha región y; por consiguiente, un aumento en la proporción del número de pixeles negros respecto a la totalidad de la imagen tal y como se planteó en la primera alternativa de verificación.

Expuestas ambas opciones, resultó más práctico utilizar la primera aunque el uso del valor promedio de los pixeles se vislumbra como la más adecuada si lo que se desea

es buscar características locales con una distribución en sus pixeles similar a otra región en específico. En resumen, para este proyecto solo se consideró una porción de imagen como válida si es que la proporción del número de su pixeles color negro (RGB = (0, 0, 0)) es mayor o igual al 50% de la cantidad total de pixeles presentes en la imagen. Para el proyecto se buscaba recuperar la mayor cantidad posible de imágenes dentro de un repositorio local previamente establecido, el cual contuviese un gran volumen de imágenes. Haciendo una estimación aproximada del volumen de imágenes con las que se trabajó, solo con el conjunto de imágenes formado por dibujos a mano alzada se tenían 20’000 imágenes (250 categorías x 80 imágenes/categoría). Manejándose 80 dibujos por categoría y si a cada uno de éstos se le extraían 10 trozos de imagen, se obtenía un total de 800 (80 x 10) regiones o características locales por cada categoría existente. Solo con 250 categorías de imágenes, se tenían doscientos mil (250 x 80 x 10 = 200’000) características locales en total, una cantidad lo suficientemente robusta para poderse obtener descriptores que permitan definir rasgos cuantitativos de las imágenes. Ahora, si a esta cantidad se le agrega el conjunto de imágenes a color, en total se tiene un total de aproximadamente 26’000 imágenes las cuales el *framework* debería poder recuperar en base a los dibujos que el usuario final ingresase.

Ya mencionadas de forma general las pautas y pasos considerados en el desarrollo del proyecto, resulta válido comenzar con una descripción más detallada de cada uno de los procesos. Para la extracción de características locales en las imágenes se tuvo que desarrollar un algoritmo el cual permitiese la obtención de estos de forma aleatoria y que validase que su contenido sea útil para la recuperación de imágenes. Puesto que se estaba trabajando con imágenes, el procesamiento a gran escala de éstas se tradujo en un tiempo de ejecución demasiado largo debido al fuerte consumo de recursos por parte del procesador de la computadora local sobre la cual se estaba desarrollando el proyecto. Ante esto, se decidió realizar la extracción de características en un computador con una mayor potencia de procesamiento; usándose para ello un servidor *Centos* 7 con *GPU NVidia Grid* proporcionado por la universidad. Además, con la finalidad de reducir aún más el tiempo de obtención de características locales en las imágenes, se implementó un algoritmo basado en la multiprogramación en paralelo mediante hilos (*threads*) con el lenguaje de programación Python; obteniéndose mejoras considerables en la velocidad de ejecución de este mas no lo suficientemente aceptables para el autor.

Realizando una investigación en el manejo de hilos en Python, se encontró que la ejecución de múltiples hilos simultáneos se encuentra limitado por el GIL (*Global Interpreter Lock*), de modo que estos se ejecutan casi de forma secuencial independientemente del número de procesadores con que cuente el equipo bajo el cual se esté corriendo el programa. No obstante, el GIL se puede deshabilitar realizándose para ello configuraciones básicas a nivel del funcionamiento del lenguaje Python (Dúque, 2011).

Lamentablemente la realización de dichas modificaciones del lenguaje Python a nivel del servidor proporcionado por la universidad no se encontraban habilitadas puesto que es un servidor compartido por otros investigadores y no de uso exclusivo del autor del presente proyecto. Debido a esta limitante externa se decidió trabajar solo con una fracción de la base de datos de imágenes descrita inicialmente con la finalidad de demostrar la efectividad del *framework* de recuperación de imágenes; no obstante, es importante aclarar que el algoritmo a desarrollar más adelante será capaz de funcionar tanto con el GIL activado como desactivado.

Como muestra representativa de la base de datos de imágenes inicialmente definida se eligió 4 categorías ("*airplane*", *"alarm clock"*, "*angel"* y *"ant"*) teniendo cada una 80 imágenes enumeradas y 320 imágenes en total. Finalmente cada una de estas se ingresaron dentro de una carpeta llamada "*Sketches"* y se generó un archivo de texto llamado "*filelist.txt*" conteniendo los nombres de estas 320 imágenes (un nombre por fila).

- **Implementación del algoritmo de extracción de características locales**
  El algoritmo creado para la extracción masiva de características locales (ver imagen 20) busca ser lo más rápido posible en cuanto a tiempo de ejecución se trata, siendo su implementación en pseudocódigo la siguiente:

```
Algoritmo: Extracción de características locales de la base de datos de imágenes

Entrada: Archivo de texto "filelist.txt" que contiene los nombres de cada una de las imágenes
         con las que se trabajará.
Salida: Conjunto de características locales de cada imagen mencionada en el archivo de texto.

1: Procedimiento extraeTrozo (dibujo, tamDiagonal, tamAncho, tamAlto)
2:       caractLocal ← obtenerCaracteristicaLocalValida(dibujo, tamDiagonal, tamAncho, tamAlto)
3:       Guardar característica local en la base de datos
4: FinProcedimiento
5: Procedimiento leeListaNombres(listaNombres, cont)
6:       dibujo ← ObtenerImagenDeBaseDatos(listaNombres, cont)
7:       tamDiagonal, tamAncho, tamAlto← ObtenerDimensionesImagen(dibujo)
8:       numCaractLocales ←10
9:       Para i ← 1 Hasta numCaractLocales Hacer
10:            Ejecutar hilo extraeTrozo(dibujo, tamDiagonal, tamAncho, tamAlto)
11:      FinPara
12: FinProcedimiento
13: Procedimiento lecturaArchivo_ExtracCaractLocales()
14:      listaNombres ← leerArchivo("filelist.txt")
15:      N ← CalcularTamanioLista(listaNombres)
16:      Para i ← 1 hasta N Hacer
17:            Ejecutar hilo leeListaNombres(listaNombres, i)
18:      FinPara
19:      Esperar que terminen los hilos
20: FinProcedimiento
21: Inicio
22:      lecturaArchivo_ExtracCaractLocales()
23: Fin
```

Imagen 21: Pseudocódigo del algoritmo de extracción de características locales de la base de datos de imágenes (imagen de autoría propia).

En el pseudocódigo mostrado se parte del procedimiento con nombre "*lecturaArchivo_EstracCaractLocales()*" y conforme éste va avanzando se puede apreciar la creación anidada de hilos con la finalidad de obtener una mayor rapidez en la obtención de características locales en el servidor. Tomando el mencionado pseudocódigo como base, es posible desarrollar un diagrama de flujo que permita representar de forma gráfica y más entendible el conjunto de pasos que fueron realizados en el algoritmo creado (ver imagen 22).

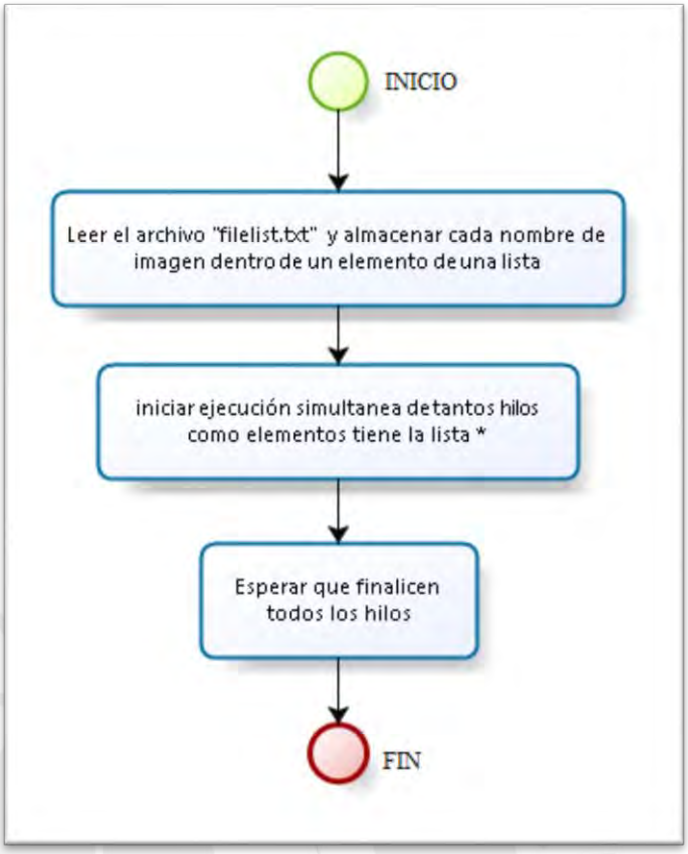


Imagen 22: Diagrama de flujo principal del proceso de extracción de características locales (imagen de autoría propia).

En el diagrama de flujo mostrado en la parte superior se puede ver que el algoritmo parte por leer el archivo "*filelist.txt*" generado por el autor y almacenar el contenido de este en una lista simplemente enlazada. Este paso inicial es importante puesto que acto seguido permitirá iniciar la ejecución tantos hilos como imágenes se tiene en la lista y por ende en la carpeta "*Sketches"* (nuestra actual base de datos local de 320 imágenes).

Cada uno de estos hilos comenzará con la extracción de características de una imagen en específico. Una vez se estén ejecutando en simultaneo todos los hilos, el programa principal (el "*hilo padre*") procederá a esperar que estos finalicen antes para dar por terminado su propio flujo. Ya explicado el funcionamiento del flujo principal, se procederá a explicar en detalle el conjunto de pasos que se siguen dentro del proceso "iniciar ejecución simultanea de tantos hilos como elementos tiene la lista *" de la imagen 22.

Dentro de este proceso se realizan los siguientes pasos descritos a continuación:

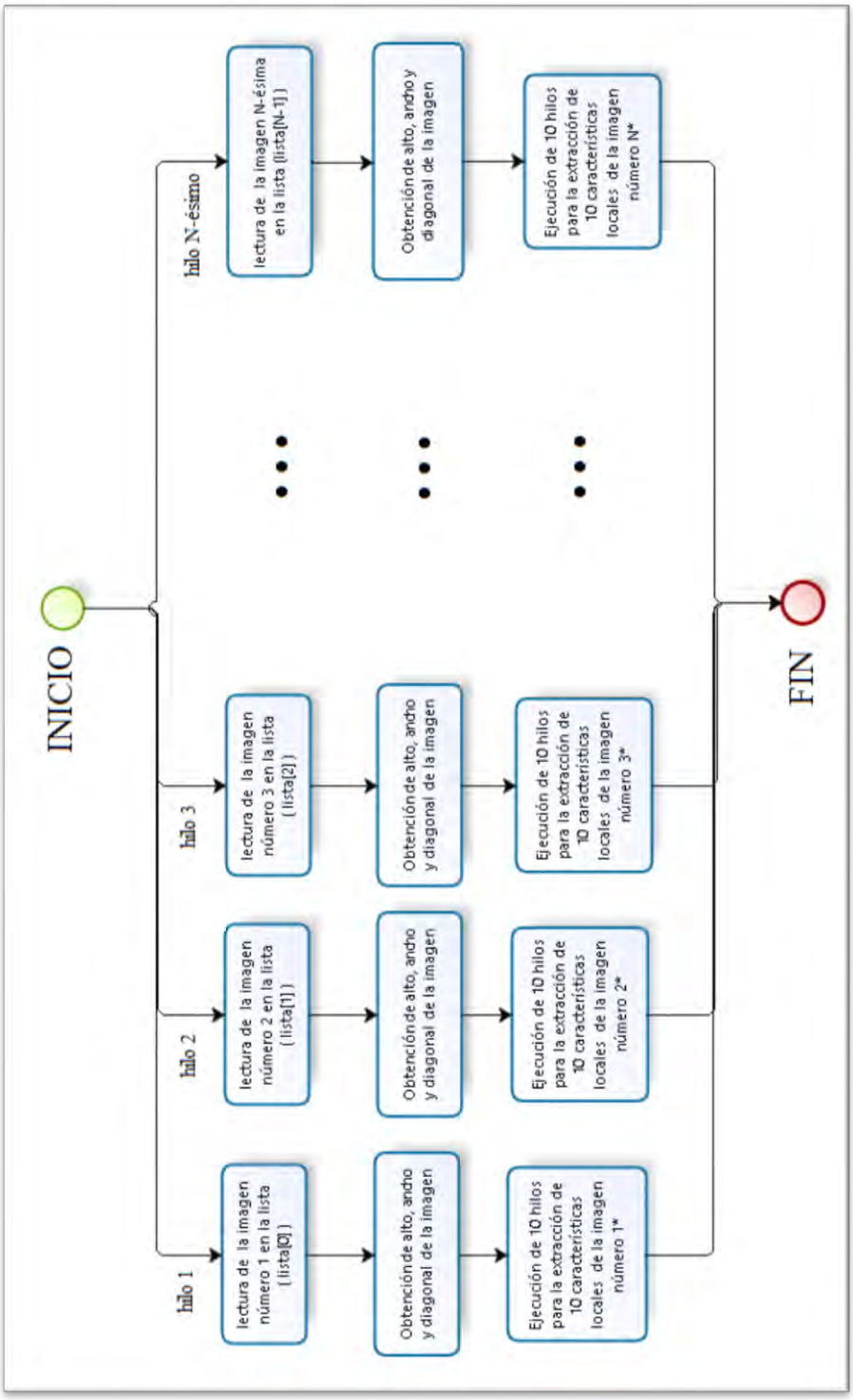


Imagen 23: Diagrama de flujo del proceso "iniciar ejecución simultanea de tantos hilos como elementos tiene la lista *" (imagen de autoría propia).

Como se puede ver en la imagen 23, se inician de forma paralela tantos hilos como imágenes se tienen en la base de datos de imágenes (carpeta "*Sketches*"), en cada hilo se obtienen las dimensiones de la imagen que se le ha asignado para luego iniciar la ejecución anidada de 10 hilos simultáneos adicionales: uno por cada característica local de la imagen que se desea obtener.

Este proceso llamado "Ejecución de 10 hilos para la extracción de 10 características locales de la imagen número i*", en donde evidentemente i toma valores que van desde 1 hasta la cantidad de imágenes en la base de datos "*Sketches"* (320 imágenes) abarca internamente un flujo de procesos representado gráficamente líneas más abajo (ver imagen 24). En este, primero se obtienen valores aleatorios para la ubicación de la característica local, los cuales son debidamente validados a fin de no obtener características locales cuyo aporte a la recuperación de imágenes sea nula, este proceso abarca un conjunto de procedimientos que líneas más abajo se detallarán con mayor detalle.

Una vez obtenido una ubicación válida dentro de la imagen completa, se prosigue con extraer de allí la característica local y almacenarla en una carpeta con el nombre de la categoría a la cual pertenece la imagen. Cabe señalar que esta carpeta se encuentra dentro de otra llamada "*Local Features*", con lo cual se obtiene una bolsa de características locales categorizadas (ver imagen 25). Como se comentó, el proceso de nombre "Obtener ubicación aleatoria de la característica local en la imagen i-ésima" también incluye las validaciones necesarias para determinar si una característica local es aceptada o no para ser empleado por el proyecto. Este incluye los aspectos mencionados previamente respecto al valor promedio de los pixeles, así como la proporción de pixeles negros respecto a los de color blanco (ver imagen 26).

Primero se verificará que la imagen candidata a característica local incluya trazos en su contenido, de no tenerlos se pasará a buscar otra región que los tenga. En caso la característica local si presente trazos en su contenido, se verificará que la proporción de pixeles negros respecto a los pixeles blancos sea mayor o igual a 2%, caso contrario se rechazará la región y se procederá a buscar otra región candidata a característica local. En caso la región candidata pase ambas validaciones, el proceso "Obtener ubicación aleatoria de la característica local en la imagen i-ésima" retornará las coordenadas de dicha región al proceso de "Ejecución de 10 hilos para la extracción de 10 características locales de la imagen i-ésima*" y este continuará su flujo con normalidad.

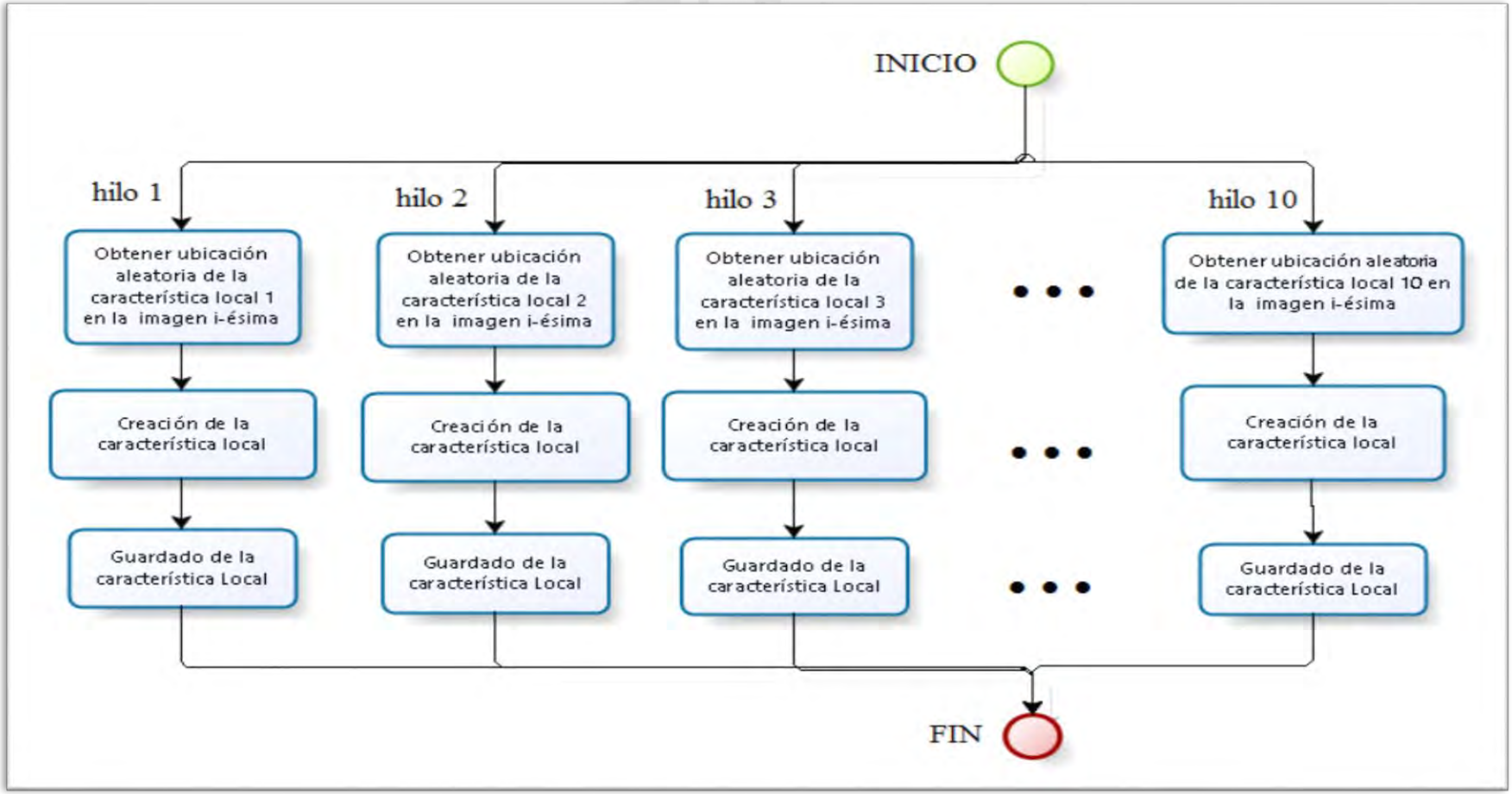


Imagen 24: Diagrama de flujo del proceso "Ejecución de 10 hilos para la extracción de 10 características locales de la imagen i-ésima*" (imagen de autoría propia).

Ya terminado el flujo de cada uno de los hilos iniciados simultáneamente, se obtuvo un repositorio local de características de determinadas regiones importantes de la imagen (no de todas), las cuales contienen los trazos más representativos de cada una de estas: se tiene la bolsa de características (*bag-of-features*).

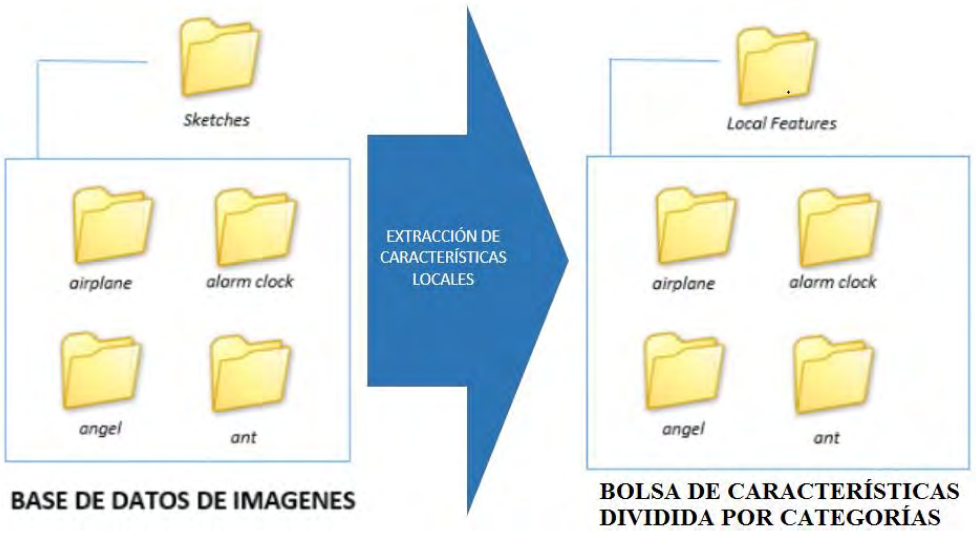


Imagen 25: Estructura de almacenamiento de las imágenes originales y sus respectivas características locales (imagen de autoría propia).

A partir de estas regiones es posible obtener información cuantificable y medible con el cual generar el diccionario o *codebook* que permita diferenciar una imagen de otra, esto se verá en capítulos posteriores.

- **Selección de un descriptor**

  Ya obtenidas las características locales por cada imagen que se tiene en el banco de imágenes, se procedió a obtener el vector de características de cada una de estas. Esto es posible mediante la obtención de valores cuantitativos para cada uno de los trozos de imágenes, con lo cual se permite establecer rasgos diferenciadores entre cada uno de ellos. En la actualidad existen diversos descriptores que permiten obtener el ya mencionado vector de características y, según la literatura revisada, se tiene constancia de que en trabajos previos basados en la metodología de *Bag-of-Features* se han empleado una diversidad de descriptores válidos como el *Shape Context*, histogramas especializados y SIFT; resaltando éste último del resto puesto que permite una mayor extracción de rasgos característicos en la imagen (Hildebrand, Eitz, Boubekeur, & Alexa, 2011).

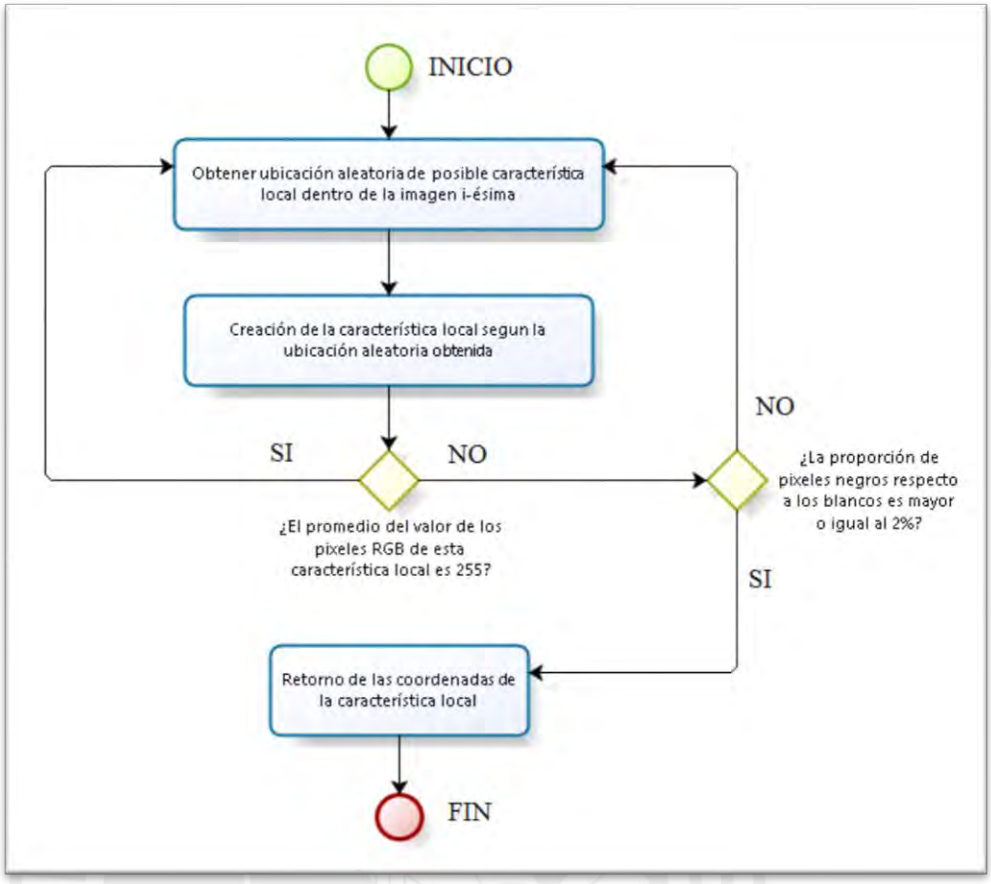


Imagen 26: Diagrama de flujo del proceso "Obtener ubicación aleatoria de la característica local en la imagen i-ésima" (imagen de autoría propia).

No obstante, el descriptor SIFT presenta una enorme desventaja precisamente en una de sus propiedades más utilizadas: el empleo del color. En efecto, este descriptor permite la obtención de una mayor gama de características pero se apoya principalmente en la identificación y detección del color y como estas varían a lo largo de toda la imagen. Para el proyecto de recuperación de imágenes expuesto, utilizar el descriptor SIFT no tiene demasiado sentido puesto que la extracción de características se realizará sobre siluetas de imágenes o trazos de un solo color y, de usarse, los resultados devueltos serían demasiado escuetos e insuficientes (Hildebrand, Eitz, Boubekeur, & Alexa, 2011). Ante esta situación es evidente la necesidad de emplear otra metodología para la extracción de vectores de características.

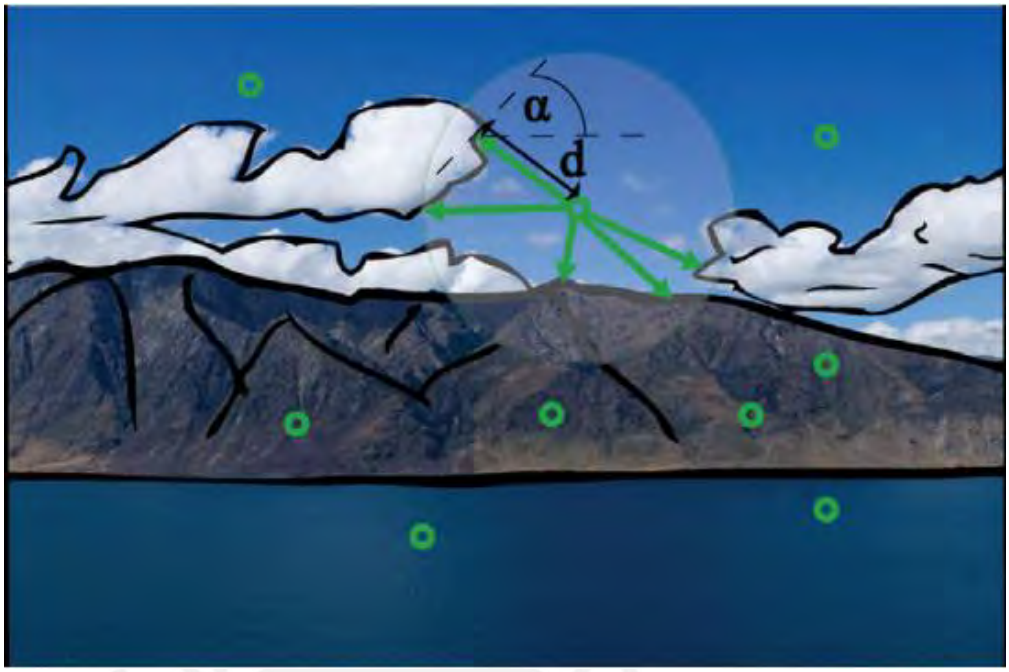


Imagen 27: Ejecución del descriptor *Spark* aplicado en imágenes a color. (Imagen obtenida del url: http://cybertron.cg.tu-berlin.de/eitz/pdf/2010_tvcg_prelim.pdf)

Revisando en la bibliografía se encontraron registros del funcionamiento de un descriptor en particular de nombre *Spark* el cual presentaba un nivel de afinidad considerable con el proceso de extracción de características numéricas a partir de dibujos o siluetas en blanco y negro. No obstante, su funcionamiento se encontraba inicialmente orientado en la obtención de vectores de características en imágenes a color (ver imagen 27) y enfatizaba en que se utilice el algoritmo de detección de bordes con el mismo nombre que el descriptor: *Shape Context*. No obstante, este algoritmo no era tan eficiente como el algoritmo de Canny (Hildebrand, Eitz, Boubekeur, & Alexa, 2011) ya utilizado en pasos previos del proyecto, por lo que se decidió desarrollar una variante de éste descriptor que trabajase con imágenes transformadas en siluetas a través del algoritmo de Canny.

El uso de este descriptor consiste en generar puntos aleatorios dentro de la imagen o porción de ésta. Además, estos puntos deberán estar ubicados en zonas libres de trazos. Para el presente proyecto se decidió que se utilizarán por cada característica local de imagen un total de 500 puntos aleatorios, una vez ubicados estos se pasará a trazar al azar líneas rectas desde cada una de éstos hasta que se intersecte con algún trazo o silueta de la imagen. Cuando se produzca el cruce entre la recta y el trazo/silueta de la imagen, se procederá a almacenar las diversas características numéricas que se encuentren presentes en el mencionado punto de intersección (ver imagen 28).

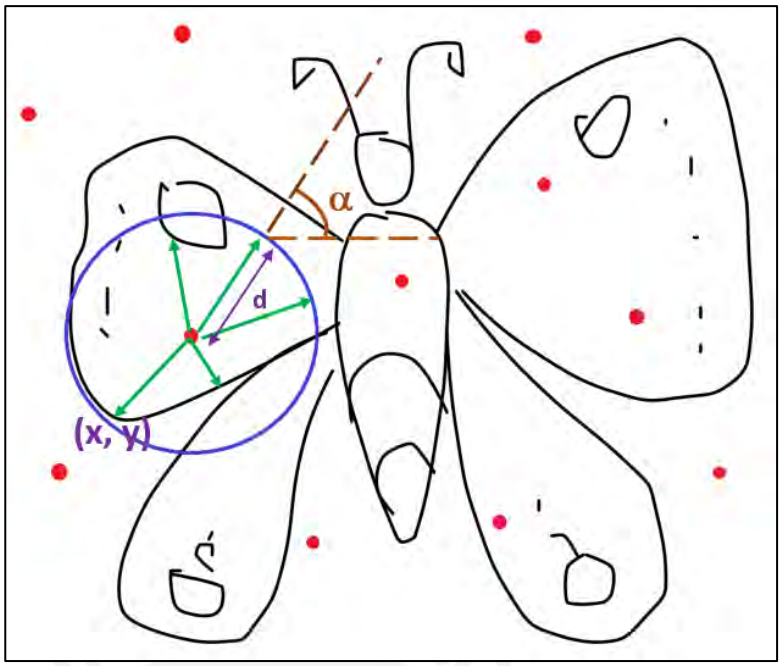


Imagen 28: Funcionamiento del descriptor *Spark* aplicado en trazos/siluetas (imagen de autoría propia).

Las características numéricas que se desprendan de estos cruces serán registrados dentro de un vector de características por cada característica local con la que se trabaje (Hildebrand, Eitz, Boubekeur, & Alexa, 2011). Para el proyecto de fin de carrera, se decidió trabajar inicialmente con las siguientes características:

- ✓ Ubicación del punto de cruce en el eje de coordenadas X.
- ✓ Ubicación del punto de cruce en el eje de coordenadas Y.

Dada la forma versátil de trabajar de este descriptor con los trazos sobre una imagen, resulta factible su uso tanto para la obtención de las características de las imágenes a mano alzada como de las imágenes transformadas en siluetas dentro de su respectivo repositorio local. Ya definido el descriptor y las características a obtener por cada punto de cruce en las características locales, se pasó a implementar el algoritmo *Spark* con algunas modificaciones hechas sobre su versión original.

- **Implementación de la variante del descriptor *Spark***

  La variante del descriptor *Spark* implementado presenta la siguiente estructura mostrada en pseudocódigo:

Algoritmo: Detección de trazos dentro de una imagen y su vector de características

Entrada: Archivo de texto "*lista_categorias.txt*" que contiene los nombres de cada una de las categorías de imágenes con las que se trabajará.

Salida: Lista de imágenes conteniendo un vector de características por cada una de las características locales (trozos de imágenes) de cada imagen.

1: Función: varianteDescriptorSpark ()

2: *listaCategorias* ← CrearListaConNombresDeLasCategorías ("*lista_categorias.txt*")

3: */*Se obtiene el total de categorías de imágenes con que se trabajará */*

4: *cantidadCateg* ← obtenerTamanhoLista (listaCategorias)

5: */* Se inicializan los índices, contadores y el total de características locales que se desea extraer */*

5: *numCL* ← *1; totalCL* ← *10;*

6: *contador* ← *contPunto* ← *cont* ← *0*

7: *ptoCruceX* ← *ptoCruceY* ← *- 1 /*Inicializamos variables auxiliares con -1*/*

8: *Inicializar la lista vacía "lista de vectores de características"*

9: *Inicializar la lista vacía "lista de listas de vectores de características"*

10: *Inicializar la lista vacía "lista de imágenes"*

11: **Mientras** (*contador < cantidadCateg*) **Hacer**

12: *nombreCategoria* ← *obtener elemento de la lista "listaCategorias" en la posición dada por la variable "contador"*

13: *Leer característica local de una imagen con nombre con formato "<cont>.png" perteneciente a la categoría "nombreCategoria"*

14: **Mientras** *(contPunto < 500)* **Hacer**

15: *Obtener coordenada (x1, y1) de un punto que no esté sobre un trazo de la imagen*

Imagen 29-1: Pseudocódigo de la variante del descriptor *Spark* aplicado para imágenes formadas únicamente por trazos o siluetas (imagen de autoría propia).

```
16:     Mientras (VERDADERO) Hacer
17:         Mientras (VERDADERO) Hacer
18:             obtener coordenada (x2, y2) aleatoriamente
19:             generar recta que va desde (x1, y1) hasta (x2, y2)
20:             (ptoCruceX, ptoCruceY) ← buscarPrimeraInterseccion_
                                         EntreTrazoYrecta(x1, y1, _
                                         x2, y2)
21:             Si (ptoCruceX, ptoCruceY) no se obtuvo antes y no es _
                (-1, -1) Hacer
22:                 Almacenar el punto encontrado en una lista _
                    de puntos de cruces
23:                 FinMientras /* se termina el bucle Mientras*/
24:             FinSi
25:             Si ((ptoCruceX, ptoCruceY) es igual a (-1, -1)) Hacer
26:                 FinMientras /* se termina el bucle*/
27:             FinSi
28:         FinMientras /* se termina el bucle*/
29:         Si ((ptoCruceX, ptoCruceY) es diferente a (-1, -1)) Hacer
30:             generar un "vector de características" a partir del
                punto de cruce
31:             insertar el "vector de características" en la posición
                "contPunto" de la lista "lista de vectores de
                características"
32:             FinMientras /* se termina el bucle*/
33:         FinSi
```

Imagen 29-2: Pseudocódigo de la variante del descriptor *Spark* aplicado para imágenes formadas únicamente por trazos o siluetas (imagen de autoría propia).

A través del uso del lenguaje de programación Python, lo primero que se estableció fue la forma en cómo se leería cada una de las imágenes presentes en la base de datos local, para ello se creó un archivo de texto con nombre "*lista_categorias.txt*" conteniendo un listado de los nombres de cada una de las categorías de imágenes existentes en la carpeta "*Sketches*" anteriormente mencionada. Para efectos prácticos en la programación del descriptor, se almacenó los nombres de todas las categorías en una lista (línea 2 en el pseudocódigo de la imagen 29-1). Lo que hace en términos generales la variante del descriptor *Spark* desarrollada por el autor es obtener 500 puntos sobre los trazos o siluetas (también llamados "puntos de interés") de cada una

de las 10 características locales extraídas de cada imagen que conforma el repositorio local de imágenes "*LocalFeatures"*.

```
34:                     FinMientras /*se termina el bucle*/
35:                     contPunto ← contPunto + 1
36:             FinMientras /* se termina el bucle*/
37:             ordenar de menor a mayor la "lista de vectores de características"
38:             insertar la lista "lista de vectores de características" en la posición
                "numCL – 1" de la lista "lista de listas de vectores de características"
39:             Borrar contenido de todas las listas, menos la "lista de listas de vectores
                de características"
40:             numCL ← numCL + 1
41:             Si (numCL > totalCL) Hacer
42:                     insertar la lista "lista de listas de vectores de características" en
                        la posición "cont" de la lista "lista de imágenes"
43:                     Borrar contenido de la lista "lista de listas de vectores
                        de características"
44:                     cont ← cont + 1 /* se incrementa el contador "cont" para cambiar
                        de imagen en el repositorio local*/
45:                     numCL ← 1 /* se resetea el índice*/
46:                     Si (la variable "cont" es múltiplo de 80) Hacer /* 80 es la cantidad
                        de imágenes por categoría*/
47:                             contador ← contador + 1 /* se incrementa el contador
                                del mismo nombre para cambiar de categoría*/
48:                     FinSi
49:             FinSi
50:     FinMientras
51:     Retornar la lista "lista de imágenes" /*lista devuelta por la función */
52: FinFunción
```

Imagen 29-3: Pseudocódigo de la variante del descriptor *Spark* aplicado para imágenes formadas únicamente por trazos o siluetas (imagen de autoría propia).

Luego de ser cargada la característica local con la cual trabajará el descriptor, se obtiene un punto que no se encuentre sobre trazo o silueta alguna en la región (línea 15 en el pseudocódigo de la imagen 29-1). A partir de este punto se comienzan a trazar rectas de longitud aleatoria una tras otra hasta que se encuentre que una de ellas intersecta a un trazo o silueta, una vez obtenido dicha intersección se pasa a registrar las coordenadas de cruce en una lista a fin de que en el futuro no se almacenen puntos

de cruce iguales en una misma característica local (líneas 16 al 34 en el pseudocódigo de la imagen 29-2). A continuación se genera el vector de características para ese punto y se inserta dentro de una lista de vectores de características asociada a esa característica local en específico (líneas 29 al 33 en el pseudocódigo de la imagen 29-2). Los pasos descritos se realizan 500 veces, obteniéndose en consecuencia la lista de vectores de características con 500 elementos: uno por cada punto de cruce. Durante el desarrollo del proyecto, se decidió adicionalmente hacer uso de las herramientas de OpenCV para poder tener una representación visual del comportamiento del algoritmo creado conforme este se va ejecutando, mostrándose como puntos de color rojo a aquellos obtenidos al azar y que se encuentran en regiones libres de trazos o siluetas; y de color verde a los puntos de cruce con los trazos o siluetas (ver imagen 30).

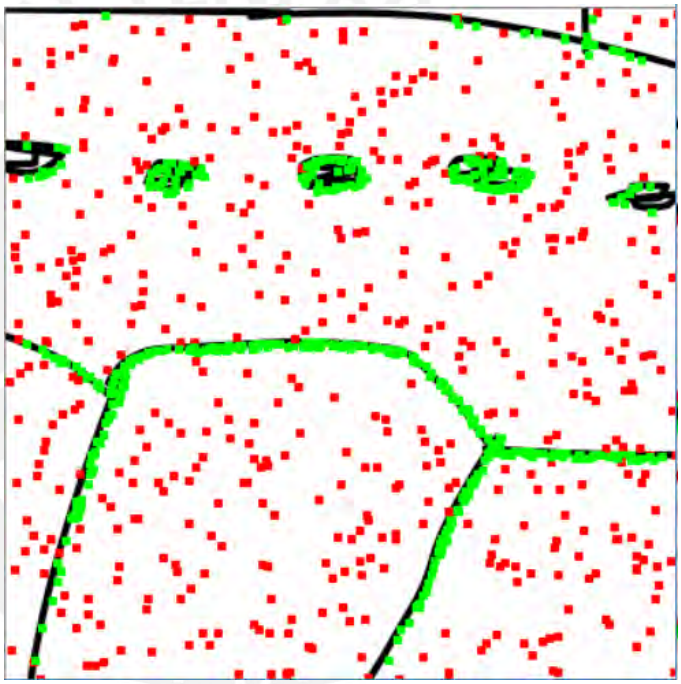


Imagen 30: Visualización de los resultados obtenidos utilizando el descriptor *Spark* (imagen de autoría propia).

Volviendo al pseudocódigo, ya con la lista de vectores de características conteniendo 500 de estos mencionados vectores, se procedió a hacer un ordenamiento burbuja ascendente (líneas 37 en el pseudocódigo de la imagen 29-2) de los puntos de cruce presentes en la imagen con la finalidad de que estos simulasen un determinado recorrido por toda la característica local, lo cual facilitará la posterior comparación de trazos. Para el proyecto se decidió que el orden de los puntos de cruce en la lista de vectores de características fuese de tal forma que, partiendo del origen de coordenadas,

primero se realice un desplazamiento ascendente por el eje de las abscisas y luego por el eje de las ordenadas (ver imagen 31).

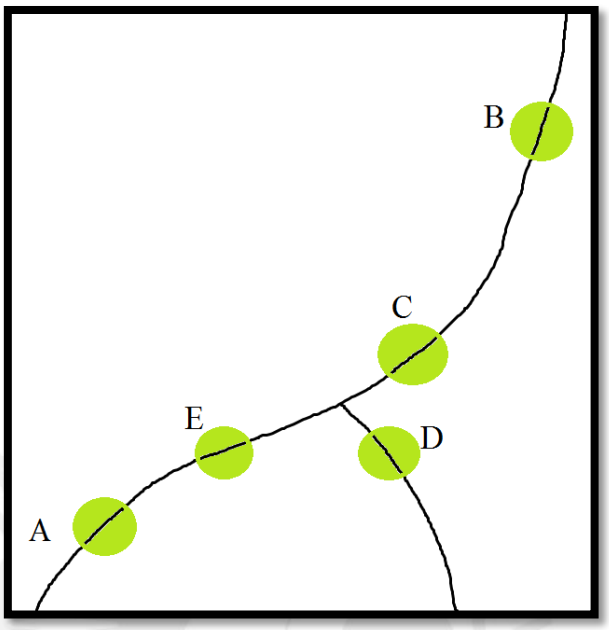


Imagen 31: Simulación del recorrido de puntos en una característica local, donde los puntos verdes son los detectados por el descriptor y A, E, D, C y B el orden en que serían ingresados en la lista de puntos de cruce (imagen de autoría propia).

Tomándose como ejemplo la imagen 31 que representa una sección pequeña de una característica local, se puede apreciar que la variante del descriptor *Spark* ya ha obtenido 5 puntos de cruce representados de color verde en la imagen. Estos puntos serán almacenados en el orden aleatorio en que son obtenidos; no obstante, luego de ordenarse la lista de vectores de características tendrá el orden A, E, D, C y B. Para lograr ello bastó con visualizar una característica local como una matriz de "m" filas y "n" columnas, donde cada uno de sus elementos representa un pixel a evaluar, siguiéndose para ello el desplazamiento descrito a continuación (ver imagen 32).

```
Algoritmo: Recorrido por los pixeles que conforman una imagen.
Entrada: Una imagen (matriz de pixeles) conformada por "m" filas y "n" columnas
1: Procedimiento LecturaMatriz(matriz[][])
2:      /*Dada una matriz de "m" filas y "n" columnas */
3:      Para i ← 1 Hasta m Hacer
4:          Para j ← 1 Hasta n Hacer
5:              evaluarContenidoPixel (matriz[i][j])
6:          FinPara
7:      FinPara
8: FinProcedimiento
9: Inicio
10:     LecturaMatriz (imagen) /*una imagen es una matriz de pixeles*/
11: Fin
```

Imagen 32: Pseudocódigo del proceso de recorrido de pixeles utilizado en el proyecto (imagen de autoría propia).

Una vez ordenada la lista de vectores de características, esta se inserta dentro de una lista de listas de vectores de características que se encargará de almacenar las listas de vectores de características pertenecientes a las 10 características locales que le corresponden a una imagen; para luego insertarse dentro de una "lista de imágenes" en la que cada uno de sus elementos tiene asignada una imagen del repositorio "*Sketches*" en específico. Finalmente, los pasos descritos se repiten por cada una de las características locales de la base de datos "*LocalFeatures"*, obteniéndose como resultado que la "lista de imágenes" tenga información cuantificable de 10 características locales por cada imagen en la base de datos de imágenes "*Sketches"*, en el próximo capítulo del documento se describirá con mayor detalle esta estructura de indexación (ver imagen 33). Cabe señalar que para la ejecución de este algoritmo también se hizo uso del servidor *Centos 7* con *GPU NVidia Grid* a fin de aprovechar su rapidez de procesamiento. Implementada la variante del descriptor *Spark*, es correcto afirmar que se consigue cumplir con el segundo objetivo específico del proyecto: **"Permitir la identificación de los objetos o individuos contenidos en las imágenes mediante la generación de un vector de características que represente los atributos más distintivos en las siluetas."**

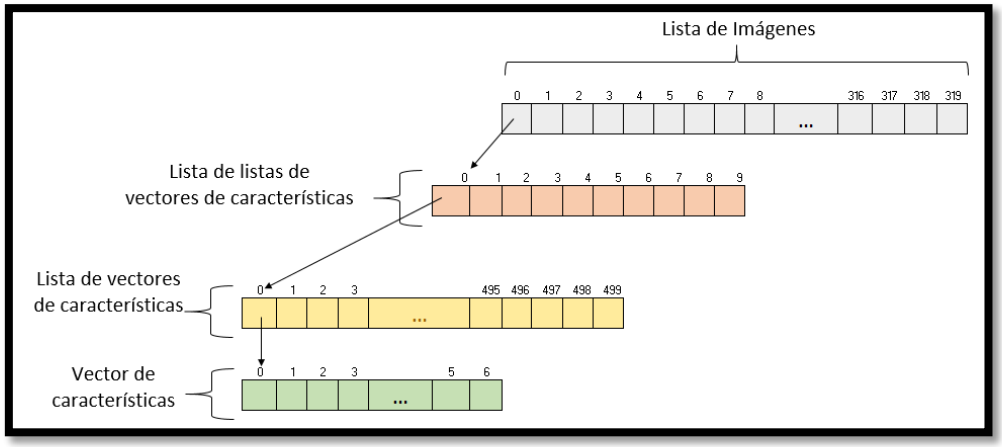


Imagen 33: Estructura usada para el guardado de información cuantitativa de las características locales de cada imagen (imagen de autoría propia).

4.4 Creación del Diccionario de Palabras Visuales (*Codebook*)

- **Guardado del contenido de la lista de imágenes en un bloc de notas**

  Hasta este momento se tiene la información de cada imagen existente en la base de datos "*Sketches*" dentro de la lista de imágenes. No obstante, independientemente del equipo utilizado, el tiempo de obtención de esta lista cambiará según la cantidad de imágenes con las que se desee trabajar. Durante el desarrollo del proyecto se decidió almacenar el contenido de la lista de imágenes dentro de un archivo de texto "*Diccionario.txt*" a fin de que cada vez que se ejecutase el *framework* no se tenga que volver a procesar todas las imágenes del repositorio "*LocalFeatures"*, sino que sea suficiente con cargar una lista con toda la información del mencionado archivo de texto, respetando la estructura inicialmente empleada (ver imagen 33). Esto último es significativamente más rápido ya que solo es un proceso de lectura/guardado de datos y no de obtención de puntos en una imagen a partir de la toma de valores aleatorios.

5. DISEÑO E IMPLEMENTACIÓN DEL *FRAMEWORK* DE RECUPERACIÓN DE IMÁGENES

5.1 Introducción

En el presente capítulo se describen los pasos a seguir una vez se hubo realizado con éxito la representación de cada una de las imágenes presentes en la base de datos local de imágenes, a través de la metodología de *Bag-of-Features*. Es importante señalar que los pasos seguidos en el capítulo anterior constituyen la base sobre la cual se desarrollará el *framework*, por lo que la ejecución de éstos solo se realizará una vez para poder generar el diccionario (*codebook*) del cual se valdrá la aplicación para realizar las futuras comparaciones. Las acciones realizadas (a nivel de código) a partir del presente capítulo sí se realizarán cada vez que el usuario ingrese un dibujo a mano alzada. A continuación se irán describiendo los pasos a seguir por el autor del presente documento, señalando los objetivos específicos que se van cumpliendo en el transcurso del desarrollo del proyecto; partiendo por intentar lograr el resultado del tercer objetivo mencionado con anterioridad: elaborar una estructura de indexación que facilite la comparación de los dibujos hechos a mano por el usuario con las imágenes presentes en el repositorio local de imágenes.

5.2 Indexación del espacio de características

Hasta el momento se tiene ya representada de forma numérica las características correspondientes a las 10 regiones extraídas al azar de cada una de las imágenes provenientes de diferentes categorías. Evidentemente se tiene una gran cantidad de información, por lo que es necesario que se desarrolle una adecuada estructura de indexación que no solo permita localizar de forma abstracta la ubicación de cada una de las imágenes, regiones y/o vector de características correspondiente a una determinada región y/o dato específico; sino que además favorezca a la implementación y obtención de resultados del *framework* de recuperación de imágenes.

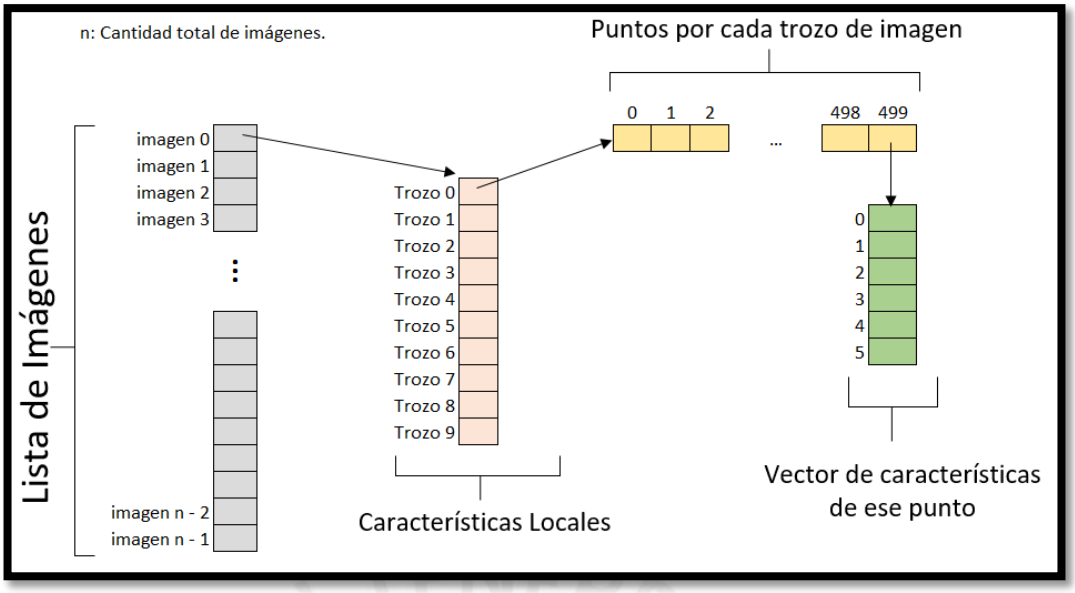


Imagen 34: Estructura de indexación del diccionario "lista de imágenes" (imagen de autoría propia).

En el capítulo anterior se manejó un conjunto de listas las cuales terminaron por integrarse entre sí para formar una lista que contenía a otras listas. Para esta etapa del proyecto se empleó una estructura de indexación similar a la ya utilizada (ver imagen 34) que simula el uso de estructuras de datos vistos mayormente en programas basados en el lenguaje de programación C/C++ como arreglos y punteros simples, dobles y triples; pero cuya adecuación al lenguaje Python 2.7.1 (en el cual se desarrolló también el *framework*) a través de listas no representó obstáculo alguno. En la estructura empleada se manejó un vector de 320 elementos, en donde cada uno de éstos responde a una imagen en específico del repositorio local de imágenes ya pre procesadas, categorizadas y debidamente transformadas en siluetas (de ser necesario). Por cada uno de estos elementos se tenía otro vector de diez bloques para las regiones aleatorias que se extrajeran de las imágenes, estos elementos contaban asimismo con su respectivo vector de puntos ubicados aleatoriamente dentro de cada región. Finalmente, este último vector tenia, por cada uno de sus 500 elementos, una lista simple de características en el cual se almacenaron los datos necesarios para la identificación de una región en particular.

Con la creación de la estructura de indexación necesaria para tener un mayor control sobre las imágenes y demás datos asociados a éstas, se logra cumplir con el tercer objetivo específico del presente proyecto de fin de carrera, el cual es **"Definir e implementar una estructura de indexación de imágenes para el manejo de éstas dentro de una base de datos."**

5.3 Implementación de *framework* de recuperación de imágenes

Con la base sobre la cual se apoyará el *framework* ya establecida e implementada tanto a nivel abstracto mediante la elaboración de una estructura bajo la cual indexar cada una de las imágenes con las que se trabajará, como a nivel funcional mediante la creación de un esquema con pautas a seguir para la correcta generación de un diccionario de palabras visuales conciso y altamente representativo de las diferentes imágenes presentes en el repositorio local de imágenes (ver imagen 34), es válido recapitular e indicar el orden bajo el cual se desarrollará el *framework* de recuperación de imágenes en su totalidad.

5.3.1 Módulo de consolidación de la base de datos de imágenes

En esta sección se desarrolló todo lo relacionado a la carga de imágenes a color y dibujos en blanco y negro. Una vez cargadas las imágenes en la aplicación, estas pasan a ser procesadas a fin de eliminar cualquier tipo de ruido visual que pudiese afectar la calidad de los resultados devueltos por el *framework* y, en el caso de las imágenes a color, ser transformadas en siluetas a través del uso de herramientas propias de la librería de OpenCV.

5.3.2 Módulo de extracción de características de las imágenes

Como ya se describió en el capítulo anterior, en esta etapa se procedió con el empleo de la metodología de *Bag-of-Features* mediante la selección de una estrategia para la obtención de características locales en una imagen. Para el caso del proyecto, se empleó una variante del descriptor *Spark* el cual se presta para ser empleado tanto en siluetas como en imágenes de dibujos hechos a mano alzada por el usuario.

5.3.3 Módulo de generación del *Codebook*

Obtenidos los vectores de características a través del uso del descriptor *Spark*, se procedió a generar el diccionario de palabras visuales (*Codebook*) en la estructura de datos "lista de imágenes" ya mencionada (ver imagen 34) y en un bloc de notas llamado "*Diccionario.txt*". Con el fin de que cada vez que se inicie el proceso de recuperación de imágenes a través del *framework*, primero se cargue rápidamente el diccionario desde un archivo de texto para luego guardarse en una estructura similar a la "lista de imágenes".

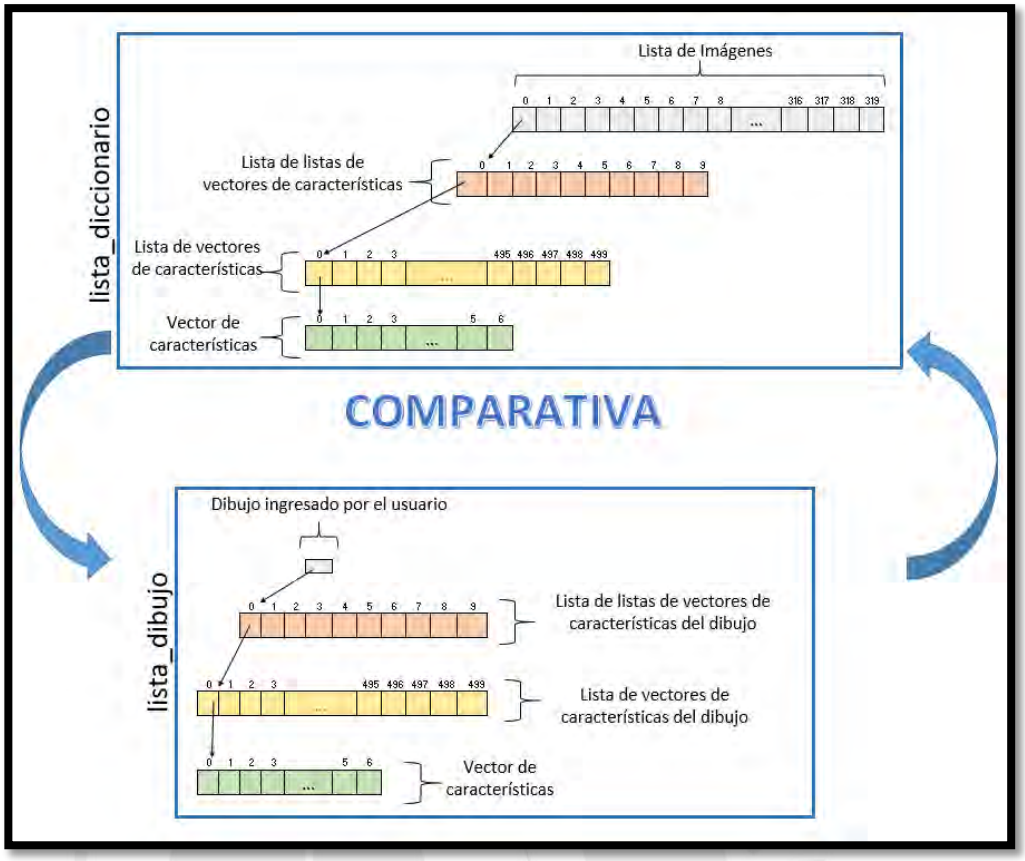


Imagen 35: Comparativa entre la estructura de indexación de las imágenes del diccionario llamada "lista_diccionario" y la del dibujo ingresado por el usuario llamada "lista_dibujo" (imagen de autoría propia).

5.3.4 Módulo de recuperación de imágenes

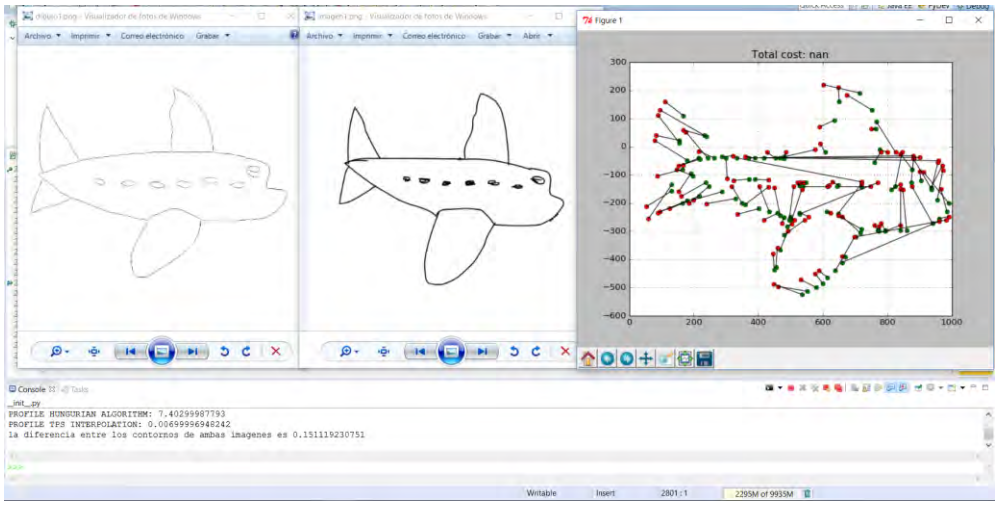
A partir de este módulo comienza la implementación del *framework* de recuperación de imágenes. En este apartado se realizaron las comparaciones entre las características extraídas del dibujo a mano alzada hecho por el usuario final y aquellas que se encuentran almacenadas e indexadas dentro del diccionario global de imágenes pertenecientes al repositorio local. Como se mencionó en el apartado anterior, los algoritmos creados para el procesamiento de imágenes pertenecientes al repositorio local también fueron utilizados en el procesamiento de los dibujos ingresados por el usuario; por lo que se adecuaron para que una vez el usuario terminase de realizar el dibujado a mano alzada de las imágenes similares que desea obtener, este pasase por las mismas etapas de procesamiento descritas en el capítulo anterior.

Con la finalidad de agilizar la obtención de resultados, así como la calidad de las imágenes recuperadas por el *framework,* se optó por apoyar éste último en un algoritmo de comparación desarrollado por el autor y basado en el recorrido a través de las estructuras de indexación tanto del dibujo ingresado por el usuario como del repositorio local de imágenes según la

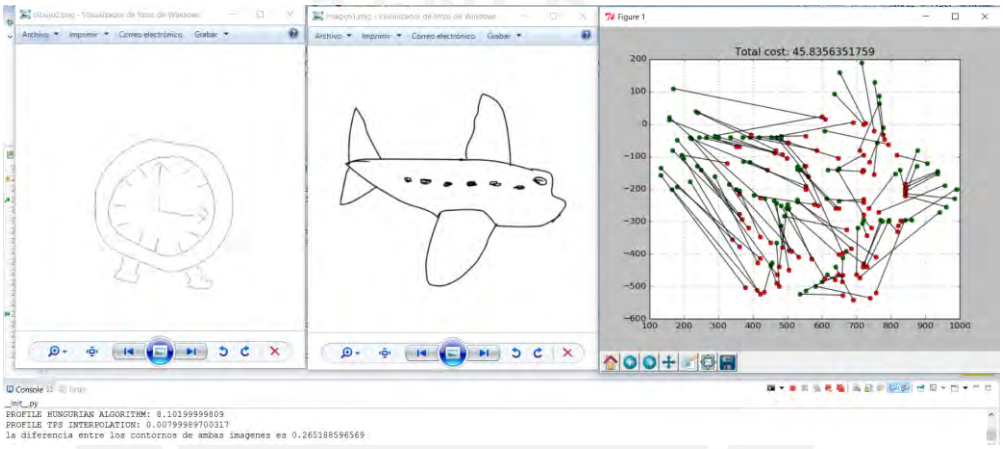
categoría cuyos datos más se parezcan (ver imagen 35). Sin embargo, dada la naturaleza del algoritmo y la cantidad de características locales manejadas, únicamente se conseguía obtener de forma acertada la imagen más parecida al dibujo ingresado pero no aquellas que guardasen un parecido ligeramente menor (al menos no en una cantidad considerable) y/o que al menos perteneciesen a la misma categoría. Como resultado del *framework* se deseaba obtener un conjunto de imágenes parecidas al dibujo ingresado y no solo una imagen, por lo que se decidió hacer uso de un descriptor adicional llamado *Shape Context* previo a la ejecución del algoritmo de comparación elaborado por el autor.

El descriptor *Shape Context* realiza comparaciones entre dos imágenes y devuelve un valor decimal entre cero y uno (que puede ser visto también como un porcentaje de disimilitud entre las imágenes) (Hildebrand, Eitz, Boubekeur, & Alexa, 2011) el cual mientras menor sea, mayor será la similitud entre éstas (ver imagen 36). Es decir, a modo de filtro inicial, primero se procedió a someter el dibujo ingresado por el usuario a la técnica de comparación propia del mencionado descriptor con todas las imágenes de la base de datos local, almacenando sus resultados en una lista de listas formada por 320 elementos, cada uno conformado por dos elementos: uno indicando la posición de cada imagen en la lista de imágenes (*lista_diccionario*) mencionada líneas arriba y otro señalando el porcentaje de disimilitud entre el dibujo del usuario y la imagen perteneciente al repositorio local.

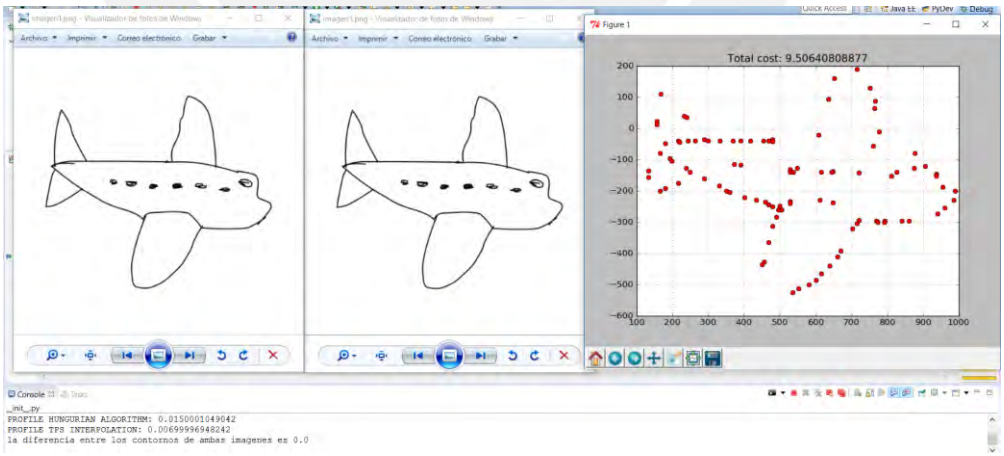
Finalmente, ya con la lista creada se procedió a ordenarla de forma ascendente y a extraer de ésta solo los "n" primeros elementos con la finalidad de disminuir el universo de imágenes con las que trabajará el algoritmo creado por el autor y por ende el costo computacional que implica la ejecución de éste. Es importante señalar que si bien el descriptor *Shape Context* es una herramienta capaz de encontrar similitudes considerables entre dos figuras que pueden estar incluso rotadas, aún presenta una tasa de error que se acentúa al usarse en figuras con demasiadas curvas en su contorno (Hildebrand, Eitz, Boubekeur, & Alexa, 2011): cuando se tiene demasiadas regiones curvas en las figuras, el descriptor no es capaz de asignar un valor alto de disimilitud entre ambas imágenes (ver imagen 36).



(a)



(b)



(c)

Imagen 36: Representación gráfica del funcionamiento del descriptor *Shape Context* cuando son dos imágenes parecidas (a), dos imágenes diferentes (b) y cuando ambas son la misma imagen (c) (imagen de autoría propia).

Razón por la cual esta herramienta será empleada solamente a modo de filtro inicial, seleccionando aquellas imágenes de la base de datos con el suficiente potencial como para posteriormente ser consideradas o no como similares al dibujo del usuario por el algoritmo de comparación mediante el uso de la metodología de bolsa de características descrito en capítulos anteriores del presente documento.

El *framework* de recuperación de imágenes como acto final realizará un listado en orden descendente de las diez imágenes más parecidas al ingresado por el usuario. En caso de que una o varias imágenes seleccionadas por el descriptor *Shape Context* no sean consideradas como similares al dibujo ingresado mediante la metodología mencionada, se les asignará a éstas los últimos lugares dentro del listado mencionado.

Para la realización de las comparaciones entre estructuras de indexación se implementó un algoritmo en lenguaje Python cuyo pseudocódigo obedece al esquema mostrado en la página a continuación:

Algoritmo: Comparación entre el dibujo ingresado por el usuario y un subconjunto de imágenes pertenecientes a un repositorio local

Entrada: Estructura de indexación de los vectores de características pertenecientes al repositorio local de imágenes (*lista_diccionario*),
estructura de indexación del vector de características del dibujo ingresado por el usuario (*lista_dibujo*),
total de características locales pertenecientes por cada imagen (*totalCL*),
número de puntos detectados por donde pasa un trazo por cada característica local (*numPuntos*),
número de imágenes presentes en cada categoría (*numImagxCateg*),
número de categorías presentes en el repositorio local de imágenes (*numCateg*),
lista de imágenes con cierta similitud al dibujo ingresado por el usuario según el descriptor *Shape Context* (*listaMatching*)

Salida: Lista de las 10 imágenes más parecidas al dibujo ingresado por el usuario (*listaResultados*).

```
1: Función comparaImagenes (lista_diccionario, lista_dibujo, totalCL, numPuntos,
        numImagxCateg, numCateg, listaMatching)
2:      /* se inicializan las variables auxiliares */
        contAciertoPunto ← contAciertoCL ← contAciertoImag ← 0
3:      Inicializar como una lista vacía la variable "listaResultados"
4:      Inicializar como una lista vacía la variable "listaAux"
5:      /* Se obtiene el número de imágenes más similares según Shape Context */
        tamListaMatching ← obtenerNumeroElementosDeLista (listaMatching)
6:      Para i ← 0 hasta (totalCL - 1) Hacer
7:          Para elem ← 0 hasta (tamListaMatching - 1) Hacer
8:              posListaImagenes ← Obtener en la posición "elem" de
                  la lista "listaMatching" la ubicación en "lista_diccionario" de
                  la imagen con la que se va a comparar el sketch.
9:              k ← 0 /* se inicializan la variable auxiliar */
10:             contAciertoCL ← 0  /* se inicializan la variable auxiliar */
11:             Para k ← 0 hasta (totalCL-1) Hacer
12:                 /* se inicializan las variables auxiliares */
```

Imagen 37-1: Pseudocódigo del algoritmo de comparación entre elementos de las estructuras de indexación de *"lista_diccionario"* señaladas por el descriptor *Shape Context* y la lista *"lista_dibujo"* (imagen de autoría propia).

```
                        m ← 0; contAciertoPunto ← 0
13:                     /* se inicializan las variables auxiliares */
                        contPuntosSimilares ← 0; es_Similar ← FALSO
14:                     /* numPuntos = 500 */
                        Para m ← 0 hasta (numPuntos-1) Hacer
15:                         es_Similar ← comparar vectores de características
                                         entre la característica  local del dibujo
                                         y la imagen.
16:                         Obtener distancia entre puntos de la caract. local
                            k-esima de la imagen elem-esima de "lista_diccionario" y
                            la caract. local i-esima de la lista "lista_dibujo"
17:                         Si (distancia <= 120) Hacer
18:                             contPuntosSimilares ← contPuntosSimilares + 1
19:                         FinSi
20:                     FinPara
21:                     proporcionPuntos ← (contPuntosSimilares/numPuntos) * 100
22:                     Si (proporcionPuntos >= 70) Hacer
23:                         contCLsimilares ← contCLsimilares + 1
24:                     FinSi
25:                 FinPara
26:                 proporcionCL ← (contCLsimilares/totalCL) * 100
27:                 Si (proporcionCL >= 10) Hacer
28:                     contImagSimilares ← contImagSimilares + 1
29:                     Insertar nombre de la imagen y su proporcionCL
                        en una lista auxiliar.
30:                     Insertar la lista auxiliar en la lista "listaResultados"
31:                 FinSi
32:             FinPara
33:         FinPara
34:     /* hasta aquí se tiene en "listaResultados" (lista no ordenada) aquellas
           imágenes que pasaron los 3 filtros */
35:     /* Se agregan a continuación las imágenes que no pasaron los filtros
           pero le ponemos un valor negativo como "proporcionCL" */
```

Imagen 37-2: Pseudocódigo del algoritmo de comparación entre elementos de las estructuras de indexación de *"lista_diccionario"* señaladas por el descriptor *Shape Context* y la lista *"lista_dibujo"* (imagen de autoría propia).

```
36:   Para n_iter ← 0 hasta (tamListaMatching-1) Hacer
37:       listaAux ← extraer el elemento "n_iter" de la lista "listaMatching"
                     correspondiente a una imagen.
38:       Asignar el valor de "proporcionCL" más bajo (- 999) a la imagen
          referenciada en "listaAux"
39:       /* verifica si listaAux está en "listaResultado" */
          yaExiste ← verificarElementoEnLista (listaResultado, listaAux)
40:       Si (yaExiste == FALSO) Hacer /* si no está, se añade a la lista
                "listaResultado" con el valor de -999 en "proporcionCL"*/
41:           Agregar "listaAux" a la lista "listaResultado"
42:       FinSi
43:   FinPara
44:   Ordenar de mayor a menor valor de "proporcionCL" la lista "listaResultados"
45:   Retornar la lista "listaResultados" con las 10 imágenes más parecidas al
      dibujo ingresado por el usuario final.
46: FinFunción
```

Imagen 37-3: Pseudocódigo del algoritmo de comparación entre elementos de las estructuras de indexación de *"lista_diccionario"* señaladas por el descriptor *Shape Context* y la lista *"lista_dibujo"* (imagen de autoría propia).

El pseudocódigo de la imagen 37 consiste en la realización de búsquedas de similitud entre cada una de las características locales del dibujo ingresado por el usuario con cada una de las características locales pertenecientes a las imágenes del repositorio local "*LocalFeatures*" (ver imagen 38). Con la finalidad de recuperar las imágenes más parecidas posibles al dibujo ingresado por el usuario, se buscó que existiese un rango de igualdad entre los vectores de características obtenidas a través del descriptor *Spark*, tomándose mayor relevancia a la distancia euclidiana entre los puntos sobre los trazos hechos por el usuario y los pertenecientes a la base de datos de imágenes "*LocalFeatures"* en cada una de sus 10 características locales (línea 15 en el pseudocódigo de la imagen 37). Ya que las listas de puntos de cruce de cada característica local (tanto del dibujo hecho por el usuario como de las imágenes pertenecientes al repositorio local) se encuentran ordenadas de menor a mayor con la finalidad de simular un recorrido ordenado a lo largo de toda la matriz de pixeles que representa una imagen (ver imagen 31), existe una alta probabilidad de que estos puntos tengan una distancia parecida si es que los trazos a comparar presentan formas similares entre sí (ver imagen 39).

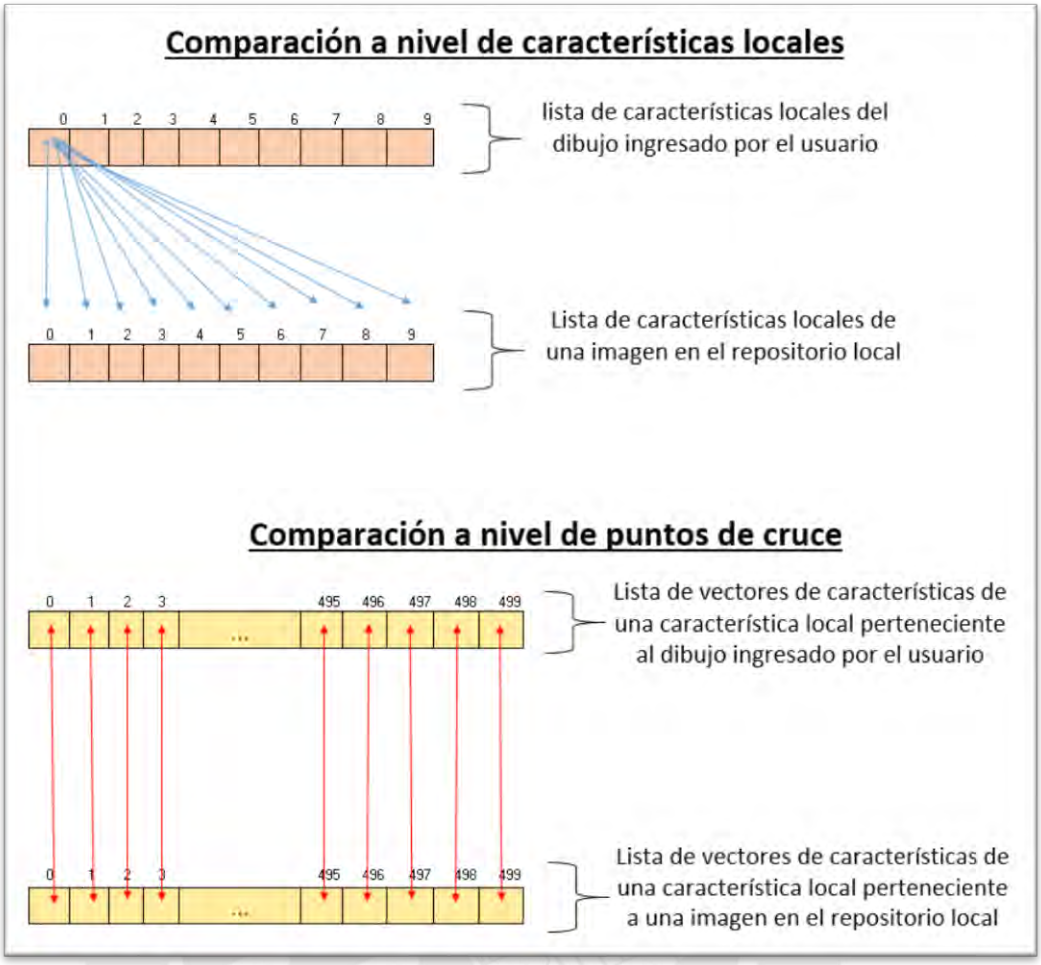


Imagen 38: Esquema de comparación utilizado para las estructuras de indexación a nivel de características locales y puntos de cruce (imagen de autoría propia).

Para el algoritmo se le asignó el valor de 50 a la distancia máxima entre puntos mencionada; sin embargo, la existencia de dos puntos (cada una perteneciente a diferentes características locales) con distancias en ese rango no es información lo suficientemente significativa como para decir que una característica local es similar a otra. Por tal motivo también se estableció que dos características locales serán consideradas iguales solo si el 40% o más de los 500 puntos pertenecientes a ambas resultan con distancias similares (línea 21 en el pseudocódigo de la imagen 37). Hasta el momento el algoritmo de comparación es capaz de diferenciar una característica local de otra, pero esto no es suficiente, ya que pueden existir imágenes distintas pero que presenten formas similares en una determinada región de sus imágenes, por ello se añadió una validación que establece que el porcentaje de características locales reconocidas como iguales (por cada imagen) sea mayor o igual al 10% (línea 26 en el pseudocódigo de la imagen 37). Finalmente tenemos que un dibujo ingresado será reconocido como similar a una imagen si y solo sus vectores de características pasan de forma exitosa por cada una de las tres validaciones mencionadas.

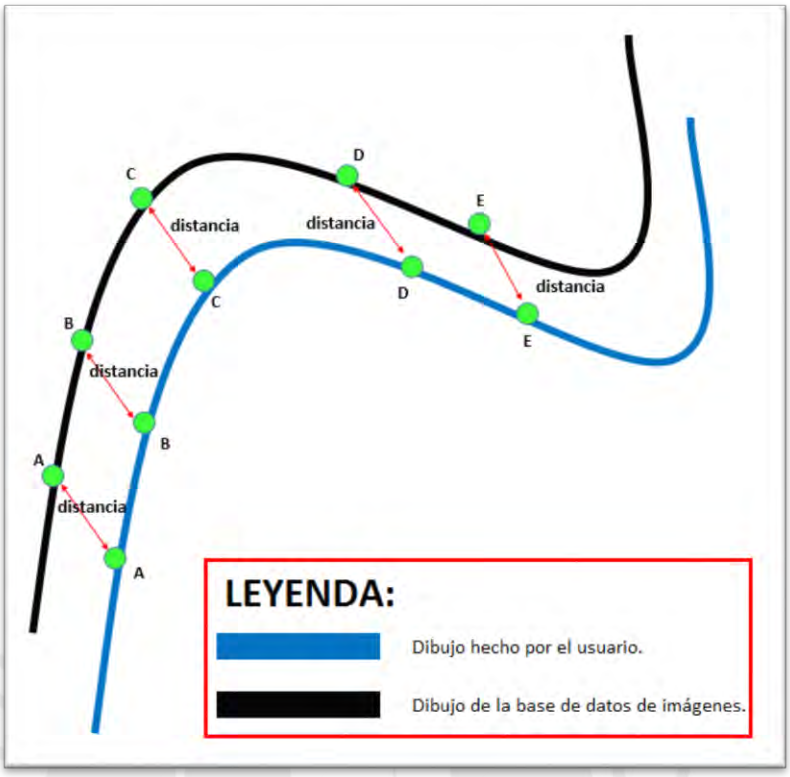


Imagen 39: Representación gráfica de la comparación a nivel de puntos de cruce A, B, C, D y E entre el dibujo hecho por el usuario y uno existente en la base de datos de imágenes (imagen de autoría propia).

5.3.5 Módulo de desarrollo de interfaz para el usuario final

En este apartado se propone la implementación opcional de una ventana a través de la cual el usuario final interactuará, tanto para el ingreso de sus dibujo a mano alzada como para la visualización de los resultados obtenidos producto de la ejecución del algoritmo de recuperación de imágenes. Puesto que el objetivo final del proyecto no es desarrollar un motor de búsqueda, sino más bien elaborar un *framework* cuyos componentes puedan ser empleados en diversas tareas, la interfaz por donde el usuario ingresa su dibujo es lo más elemental aunque intuitiva posible (ver imagen 40). Lo que se busca con el desarrollo de la interfaz es proporcionarle al usuario una visión clara de las herramientas con las que cuenta a su disposición.

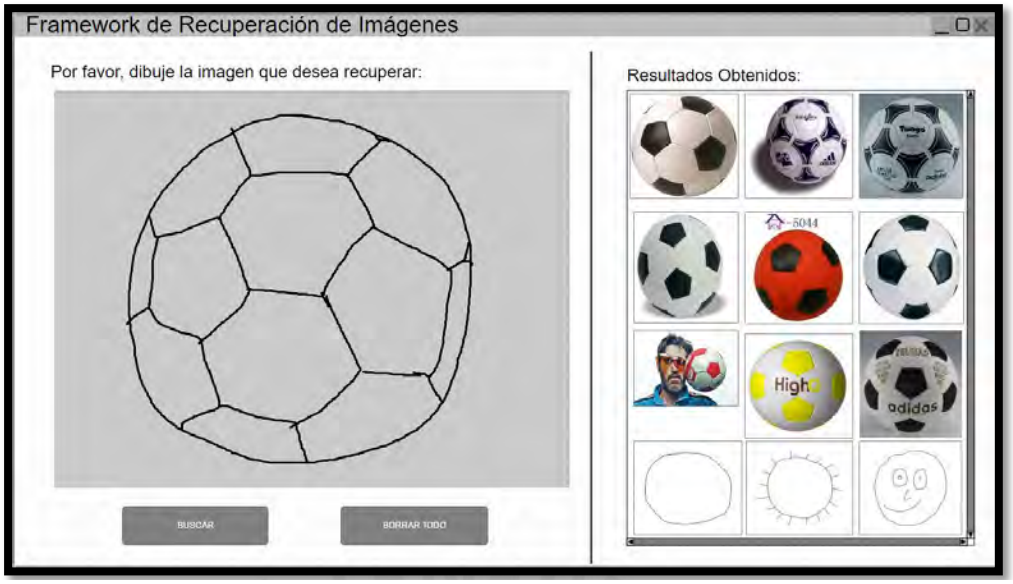


Imagen 40: Boceto de la interfaz mediante la cual el usuario final será capaz de interactuar con el *framework* (imagen de autoría propia).

En la interfaz se muestra un lienzo de tamaño mediano a través del cual el usuario puede plasmar el dibujo de la imagen que desee recuperar. Una vez realizado el dibujo, el usuario deberá hacer *click* en el botón "BUSCAR" o, en caso desee realizar otro dibujo, hacer *click* en la opción "BORRAR TODO" con lo cual se procederá a borrar por completo el contenido del lienzo. Como se comentó en capítulos previos, el dibujo ingresado por el usuario pasa por un proceso de limpieza con la finalidad de eliminar todo ruido visual que desmerezca el funcionamiento normal del *framework* de recuperación de imágenes.

Una vez procesada la imagen del dibujo hecho por el usuario, se procederá a buscar imágenes similares a la ingresada a través del algoritmo de recuperación de imágenes desarrollado para este proyecto. Los resultados devueltos por este último serán comunicados al *framework* y éste los mostrará en la parte derecha de la ventana mediante un listado con las 12 imágenes de mayor parecido al dibujo ingresado inicialmente por el usuario (ver imagen 40). Como ya se comentó, esta interfaz obedece al logro de objetivos netamente experimentales, (siendo uno de ellos **"desarrollar un mecanismo no textual que permita al usuario hacer búsquedas de imágenes abstractas pertenecientes al repositorio local de imágenes"**) por lo que no es materia de análisis que tan rápida es la obtención de los resultados mostrados sino que tan similares al dibujo hecho a mano alzada es el conjunto de imágenes devueltas por el *framework.*

5.4 Integración de módulos de caracterización e indexación

En el módulo de caracterización, se estableció la forma bajo la cual se obtendrán y almacenarán cada una de las imágenes así como sus respectivos vectores de características. Puesto que se trabajará con varias características locales (800 por categoría), es importante que cada uno de los mencionados vectores se encuentre debidamente enlazado con la característica local de la cual proviene. No obstante, dada la complejidad del proyecto y el volumen de información que éste maneja, no es suficiente con tener un conjunto de vectores asociados con imágenes. Se necesita de una estructura más elaborada la cual abarque todo el conjunto de imágenes presente en el repositorio local: una estructura de indexación que facilite el acceso a las imágenes mediante el manejo de índices que redirijan hacia todo tipo de información asociada a un determinado elemento de dicha estructura.

La estructura de indexación, como ya se mencionó en el capítulo anterior, obedece a un esquema práctico y libre de demasiadas complejidades abstractas. Esta busca simular estructuras fáciles de entender y recorrer, evitándose el riesgo de obtener datos erróneos como consecuencia de un fallido proceso de abstracción del funcionamiento del *framework* o de la forma en que están asociadas cada una de las imágenes y sus regiones parciales. Una vez implementada dicha estructura dentro del *framework* es posible recorrer cada una de las imágenes y los respectivos datos que se desprenden de éstas puesto que se cuenta con un índice y un enlace que les direcciona hacia el resto de la estructura junto con los demás datos (ver imagen 34).

Ambos módulos se complementan una vez integrados, siendo el vector de características la base sobre la cual se realizará todo el proceso de recuperación de imágenes. Esto se debe a que allí se encuentran los datos que empleará el algoritmo de recuperación de imágenes y la estructura de indexación bajo el cual trabajará el repositorio local.

6. OBTENCIÓN Y VALIDACIÓN DE RESULTADOS

6.1 Introducción

En esta sección se realiza un análisis de los resultados obtenidos a partir del uso de la gráfica precisión-exhaustividad (*precision recall*). Mediante esta herramienta se llevará a cabo un estudio de la efectividad del *framework* de recuperación de imágenes a través del ingreso de dibujos a mano alzada pertenecientes a las cuatro categorías de imágenes presentes en la base de datos local de imágenes: avión (llamado "*airplane"* en el repositorio "*Sketches"*), reloj de alarma (llamado "*alarm clock"* en el repositorio "*Sketches"*), hormiga (llamado "*ant"* en el repositorio "*Sketches"*) y ángel (llamado "*angel"* en el repositorio "*Sketches*").

6.2 Evaluación de la extracción de características locales en el dibujo ingresado por el usuario

A continuación se describirá brevemente los dibujos ingresados, así como las características locales obtenidas como consecuencia del procesamiento de estas a través de los algoritmos detallados en capítulos previos.

6.2.1 Características locales obtenidas al ingresar un dibujo perteneciente a la categoría Reloj de Alarma ("*alarm clock*")

Se ingresó de forma secuencial los dibujos de un reloj de arena (ver imagen 41), éste las pre procesó, y les extrajo 10 características locales con las cuales hacer la comparación con las imágenes del repositorio local. Estas fueron las siguientes:

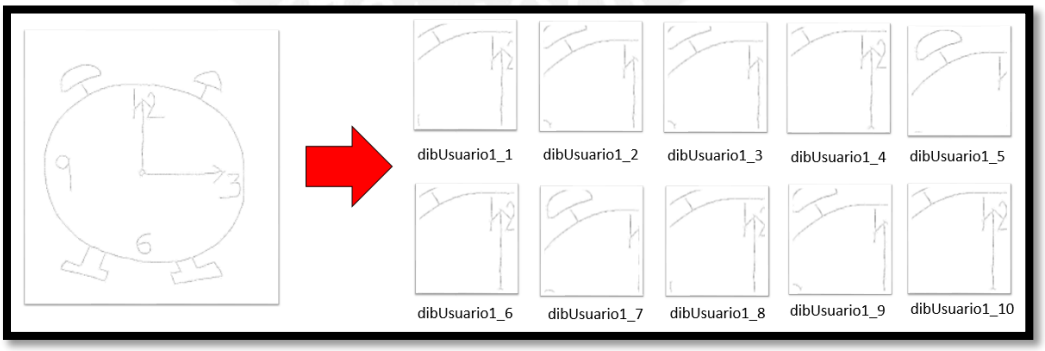


Imagen 41: Procesamiento del dibujo de un reloj de alarma ingresado por el usuario (imagen de autoría propia).

Como se puede apreciar el algoritmo de extracción de características extrajo las zonas más representativas de la imagen, siendo en su mayoría zonas del contorno circular de éste donde se encontraban las manecillas del reloj.

6.2.2 Características locales obtenidas al ingresar un dibujo perteneciente a la categoría Avión ("*airplane*")

Se ingresó el dibujo de un avión en el *framework* (ver imagen 42), éste la pre proceso, y le extrajo 10 características locales con las cuales hacer la comparación con las imágenes del repositorio local. Estas fueron las siguientes:

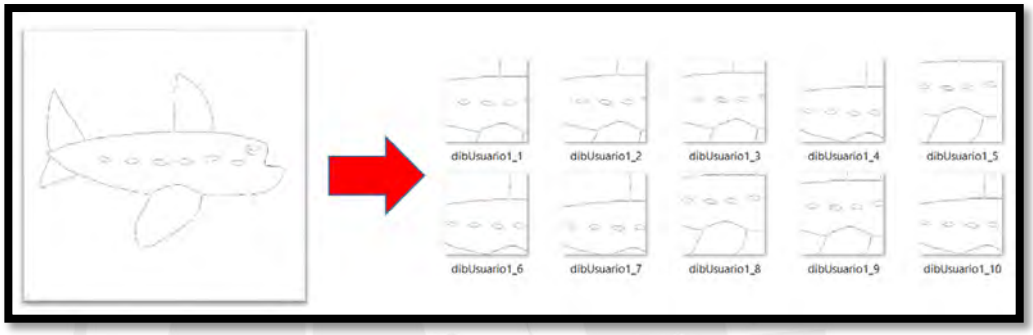


Imagen 42: Procesamiento del dibujo de un avión ingresado por el usuario (imagen de autoría propia).

De forma similar al ejemplo anterior, el algoritmo de extracción de características ha escogido las zonas o características locales con mayor cantidad de trazos, y por ende más significativa para la recuperación de imágenes. En este caso, las características locales escogieron zonas de la parte central del dibujo a mano alzada del avión.

6.2.3 Características locales obtenidas al ingresar un dibujo perteneciente a la categoría Ángel ("*angel*")

Se ingresó el dibujo de un ángel en el *framework* (ver imagen 43), éste la pre proceso, y le extrajo 10 características locales con las cuales hacer la comparación con las imágenes del repositorio local. Estas fueron las siguientes:

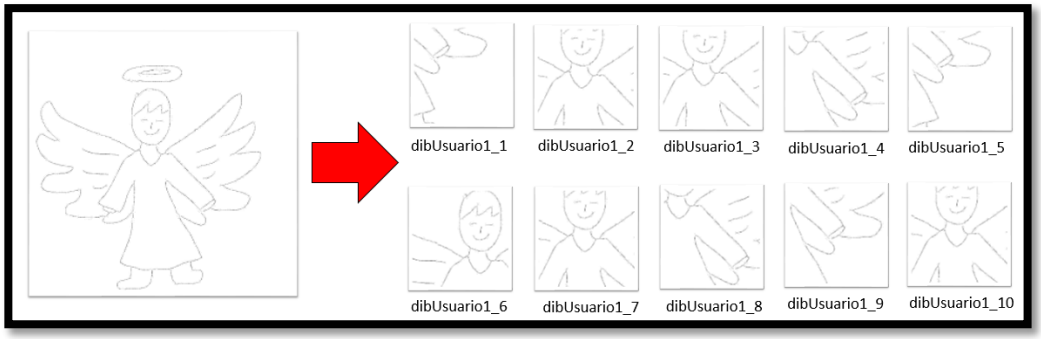


Imagen 43: Procesamiento del dibujo de un ángel ingresado por el usuario (imagen de autoría propia).

El algoritmo de extracción de características ha generado las 10 regiones con la mayor cantidad de trazos para la posterior recuperación de imágenes con siluetas similares, siendo en la mayoría regiones de la parte central del dibujo ya que allí se concentran los trazos.

6.2.4 Características locales obtenidas al ingresar un dibujo perteneciente a la categoría Hormiga ("*ant*")

Se ingresó el dibujo de una hormiga en el *framework* (ver imagen 44), éste la pre proceso, y le extrajo 10 características locales con las cuales hacer la comparación con las imágenes del repositorio local. Estas fueron las siguientes:

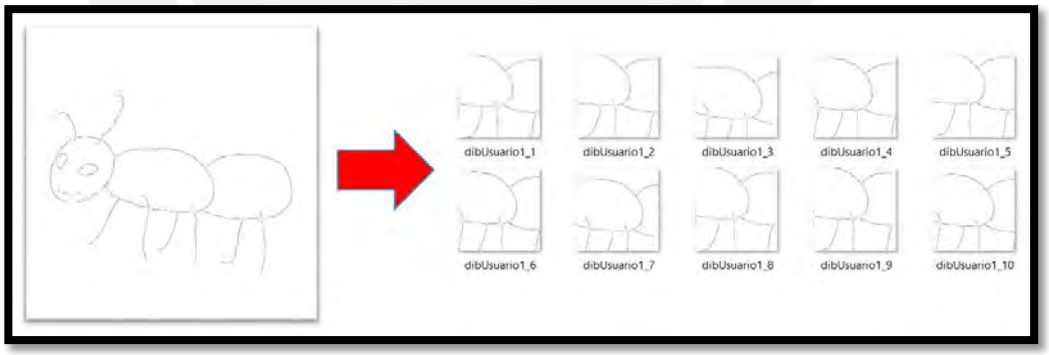


Imagen 44: Procesamiento del dibujo de una hormiga ingresado por el usuario (imagen de autoría propia).

Dado la escasez de trazos lo suficientemente agrupados, el algoritmo de extracción de características únicamente ha detectado como válidas aquellas características locales asociadas al tronco de la hormiga.

6.3 Resultados obtenidos mediante experimentación

Ya obtenidas las características locales del dibujo ingresado por el usuario y del repositorio local de imágenes a través de la metodología de bolsa de características (*Bag-of-Features*), se procedió a utilizar el descriptor *Shape Context* con la finalidad de reducir el rango de imágenes a evaluar a solo aquellas con una un parecido razonable al dibujo ingresado por el usuario para luego aplicársele el algoritmo basado en la metodología de bolsa de características. Los resultados fueron los siguientes:

6.3.1 Resultados obtenidos ante el ingreso del dibujo de un Reloj ("*alarm clock*"):

Cuando se ingresó el dibujo de un reloj los resultados fueron satisfactorios, devolviendo un listado de las diez imágenes más parecidas al dibujo ingresado por el usuario (ver imagen 45).

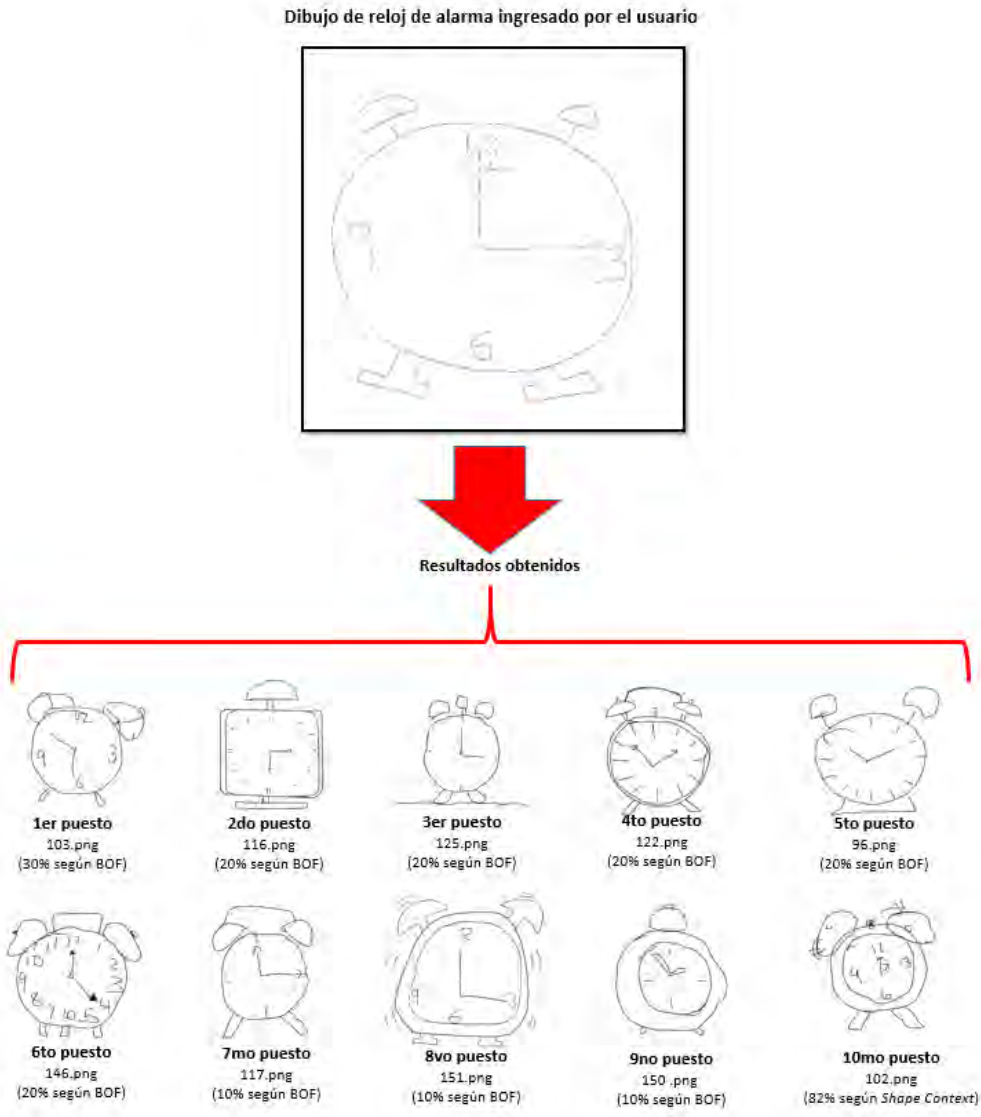


Imagen 45: Top 10 de las imágenes de relojes más parecidas a las ingresadas por el usuario según *Bag-of-Features* (BOF) y *Shape Context* (imagen de autoría propia).

Cabe señalar que de las diez imágenes mostradas una fue rechazada por el algoritmo basado en la metodología de bolsa de características desarrollado por el autor aunque si aceptada por el descriptor *Shape Context*: la imagen 102.png (ver imagen 45); otorgándole un porcentaje de similitud mucho mayor al resto imágenes. Aquí se puede apreciar claramente el margen de error presente en el descriptor *Shape Context* y como el algoritmo de comparación desarrollado por el autor contribuye a corregir dicha falencia elaborando un reajuste de cuales imágenes son verdaderamente más parecidas al ingresado por el usuario a través de filtros más estrictos basados en la extracción aleatoria de características locales (ver imagen 37), siendo este también la razón por la cual a los nueve primeros puestos se le asigna un porcentaje de similitud bajo comparado con el otorgado por el descriptor *Shape Context* a la imagen 102.png.

6.3.2 Resultados obtenidos ante el ingreso del dibujo de un Avión ("*airplane*"):

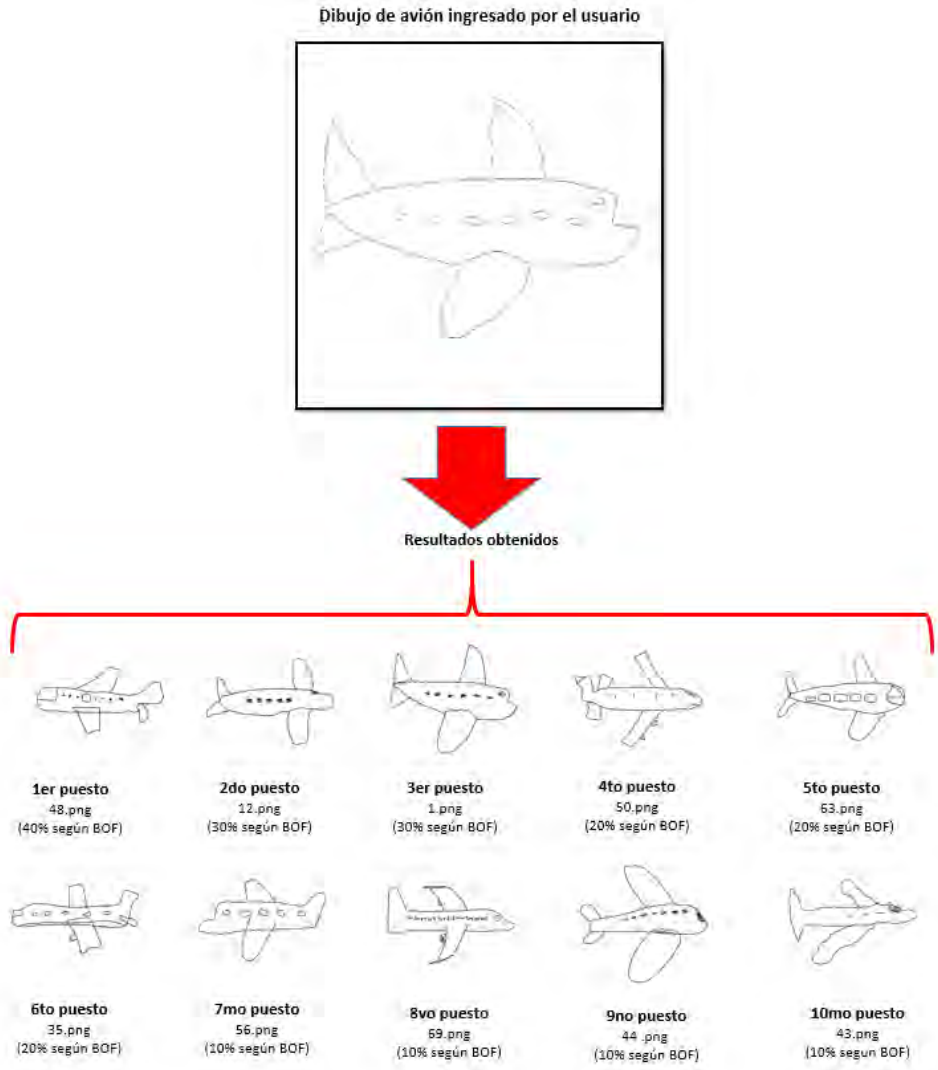


Imagen 46: Top 10 de las imágenes de aviones más parecidas a las ingresadas por el usuario según *Bag-of-Features* (BOF) y *Shape Context* (imagen de autoría propia).

Cuando se hizo el ingreso de un dibujo a mano alzada de un avión se obtuvieron también resultados aceptables salvo tres imágenes de aviones en sentido opuesto pero que pertenecen a la categoría Avión (ver puestos 1, 6 y 7 de la imagen 46). El resto de imágenes devueltas por el *framework* fueron correctas en la forma y sentido del dibujo ingresado por el usuario, ubicándose en los primeros puestos aquellas más similares con porcentajes que varían del 10 al 30 por ciento debido nuevamente a los filtros rigurosos y la metodología de bolsa de características (ver imagen 37).

6.3.3 Resultados obtenidos ante el ingreso del dibujo de un Ángel ("*angel*"):

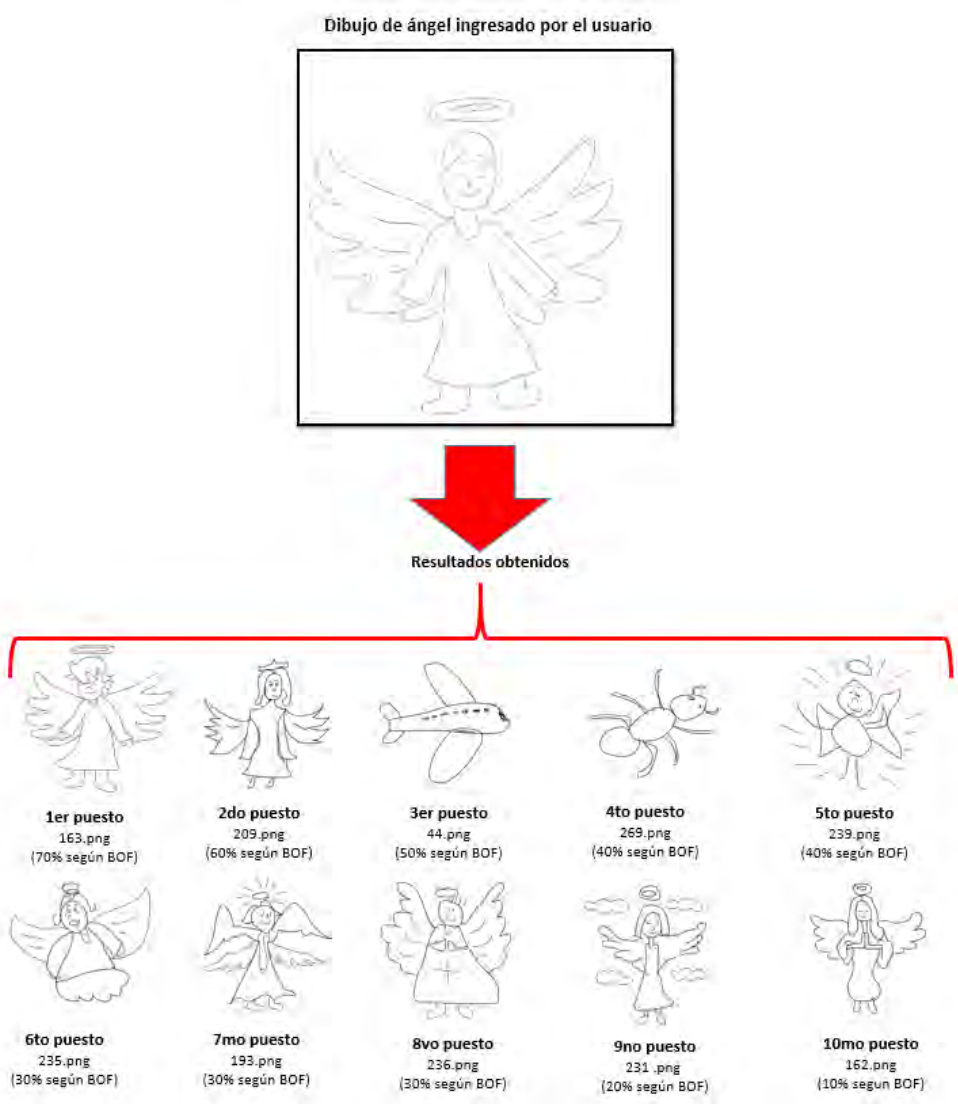


Imagen 47: Top 10 de las imágenes de ángeles más parecidas a las ingresadas por el usuario según *Bag-of-Features* (BOF) y *Shape Context* (imagen de autoría propia).

En la imagen mostrada en la parte superior se muestran los resultados obtenidos como consecuencia del ingreso de un dibujo con forma de ángel en el *framework*, obteniéndose resultados óptimos salvo en los puestos 3 y 4 en donde se tienen imágenes pertenecientes a una categoría errónea. Esto último debido a que el algoritmo desarrollado por el autor extrae características locales en las zonas más representativas para luego analizarlas, pudiendo haber confundido las alas del avión de la imagen 44.png con las alas del ángel así como el torso de la hormiga con el rostro del dibujo de ángel ingresado (ver imagen 47). No obstante las restantes 8 imágenes si son similares, incluso el primer puesto es muy parecido al dibujo ingresado, evidenciándose una asignación de porcentajes correcta para la mayoría de imágenes.

6.3.4 Resultados obtenidos ante el ingreso del dibujo de un Hormiga ("*ant*"):

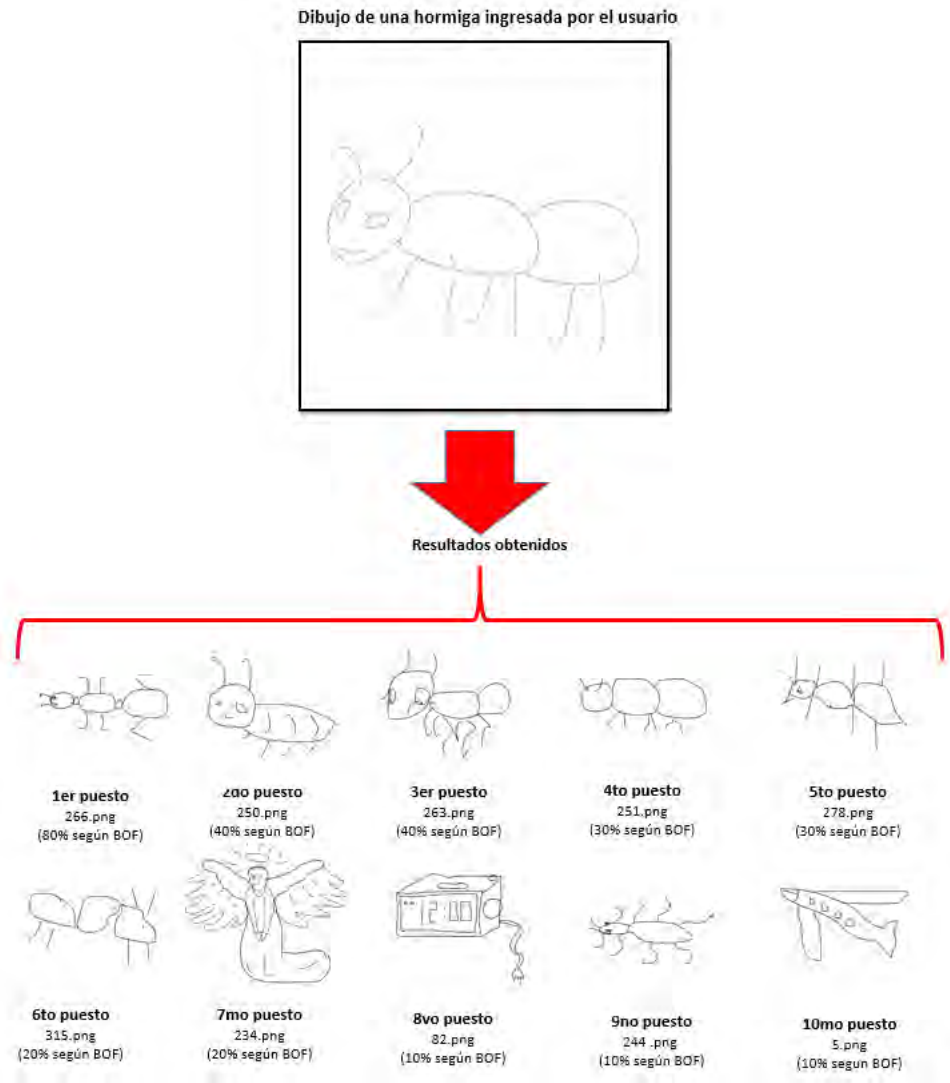


Imagen 48: Top 10 de las imágenes de hormigas más parecidas a las ingresadas por el usuario según *Bag-of-Features* (BOF) y *Shape Context* (imagen de autoría propia).

Análogamente a los experimentos anteriormente mencionados, se puede apreciar que los resultados en esta categoría también son acertados mayoritariamente, teniéndose únicamente resultados erróneos para las imágenes pertenecientes a los puestos 7, 8 y 10 (ver imagen 48). No obstante, estos se encuentran en las últimas posiciones a diferencia del resto de imágenes que si se asemejan al dibujo original las cuales se encuentran en las primeras posiciones de la lista. Viendo los resultados obtenidos de forma global se puede deducir que en cierto modo el margen de error del *framework* desarrollado es como máximo de tres por ciento. No obstante, se necesita de un mayor análisis para establecer el grado de eficiencia de la herramienta desarrollada, razón por la cual a continuación se hará uso de la curva de precisión-exhaustividad.

6.4 Análisis de resultados mediante una curva de precisión-exhaustividad

En base a los resultados obtenidos por el *framework*, mediante una gráfica de precisión-exhaustividad obtenemos lo siguiente:

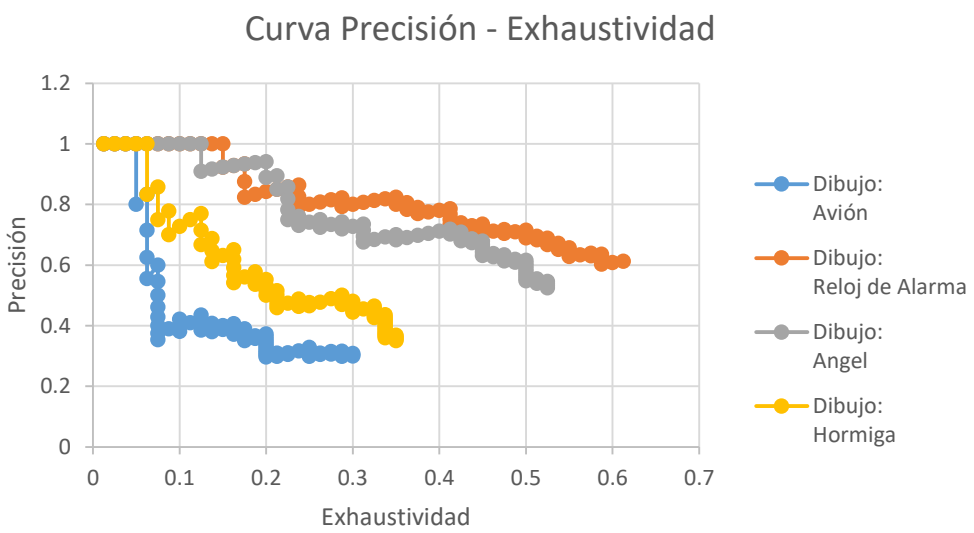


Imagen 49: Curva de precisión-exhaustividad en base a los resultados obtenidos (imagen de autoría propia).

Para el desarrollo de la curva de precisión-exhaustividad (ver imagen 49) se realizó un total de 80 consultas por imagen, aumentándose de forma incremental el número de imágenes a obtener (ver ANEXO). En ésta podemos observar que el *framework* de recuperación de imágenes obtiene mejores resultados cuando se está trabajando con imágenes pertenecientes a la categoría de "Reloj de Alarma", seguido por dibujos de "Ángel", "Hormiga" y "Avión". Específicamente, cuando se hicieron pruebas ingresando un dibujo perteneciente a la categoría "Reloj de Alarma", el *framework* fue capaz de devolver correctamente hasta 12 imágenes

pertenecientes a esta categoría y de rasgos similares al dibujo ingresado. De forma análoga para las categorías "Ángel", "Hormiga" y "Avión" se obtuvieron 10, 5 y 4 resultados correctos respectivamente, coincidiendo en todos los casos en que la imagen más similar se encontraba siempre en los primeros puestos dentro del top de resultados (ver ANEXO).

Basado en la experimentación hecha, es posible decir que los resultados conseguidos en cada una de las categorías se deben en gran medida al grado de detalle capturado al momento de extraer de forma aleatoria las características locales más representativas en cada imagen, en donde mientras la región capturada tenga mayor cantidad de detalles en su interior, más alta será la tasa de acierto del *framework*. Por tal motivo, las imágenes de relojes de alarma en la base de datos son más fáciles de recuperar; es decir, dada la ubicación de sus trazos, en sus características locales siempre se encuentran las siluetas pertenecientes a las manecillas de reloj, ninguna otra categoría tiene trazos similares a esos, haciendo que la categoría resulte fácil de identificar. En capítulos anteriores se indicó que lo que se deseaba obtener como respuesta por parte del *framework* era un listado de las diez imágenes más parecidas al dibujo ingresado por el usuario y; como se puede apreciar en la curva de precisión-exhaustividad y en los resultados mostrados líneas arriba para cada una de las categorías de imágenes dibujadas, estos se cumplen exitosamente en gran medida con la salvedad de un pequeño margen de error asociado a la ya mencionada aleatoriedad en la selección de características locales. No obstante, esto último puede corregirse si se llevan a cabo determinadas acciones como mejorar las imágenes originales con la finalidad de que presenten siluetas más diferentes entre sí, realizar un ajuste más estricto al umbral bajo el cual una imagen es considerada una característica local válida (ver imagen 37) y, sobre todo, optar por trabajar con un descriptor de mayor complejidad matemática y costo computacional para el procesamiento de las imágenes.

7. CONCLUSIONES Y RECOMENDACIONES

7.1 Introducción

En el presente apartado se detallan a continuación las conclusiones y recomendaciones desprendidas de la realización del proyecto de fin de carrera. Estas buscan orientar al lector sobre las formas de explotar la usabilidad del *framework* de recuperación de imágenes desarrollado, así como aportar algunas ideas que permitan expandir el uso de éste mediante la aplicación de la aún no tan conocida metodología de *Bag-of-Features* en ámbitos aún no explorados.

7.2 Conclusiones

Producto del desarrollo del proyecto se obtuvieron las siguientes conclusiones:

- Si se desea trabajar con una cantidad relativamente grande de imágenes, es indispensable que se ejecute el *framework* sobre un equipo con alta capacidad de procesamiento, sobre todo para la generación del diccionario de datos y los pasos previos que su elaboración conlleva. Para el presente proyecto, dicho procesamiento se realizó en un servidor *Centos* 7 con *GPU NVidia Grid* facilitado por la universidad.

- La calidad de las imágenes con las que el *framework* trabaje es de mucha importancia, puesto que depende de ésta para la extracción de características que posteriormente permitan diferenciar una imagen respecto a otras.

- El desarrollo de un *framework* para la recuperación de imágenes a partir del ingreso de dibujos a mano alzada es una herramienta útil, puesto que permite incrementar el rango de imágenes abstractas obtenibles como resultado de una búsqueda por parte del usuario. Si bien presenta un margen de error, éste es mínimo comparado con los resultados acertados que devuelve. Para efectos del presente proyecto se ha verificado que funciona correctamente cuando se desea obtener un total de las diez imágenes más similares al ingresado inicialmente por el usuario. Una mayor cantidad de resultados correctos implica una mejora a nivel de hardware las cuales escapan del alcance del proyecto de fin de carrera descrito inicialmente.

- Existe la posibilidad de mejorar los resultados obtenidos si se consigue adecuar otro tipo de descriptor de figuras al proyecto. En este caso se ha desarrollado el descriptor *Shape Context*; sin embargo, existen en la bibliografía otros tipos de descriptores que se adecuan a la morfología de la imagen que se desea recuperar.

7.3 Recomendaciones

Finalmente, se dan las siguientes recomendaciones:

- Las imágenes a incluir dentro del repositorio local de imágenes, para ser posteriormente recuperadas, deben ser las más grandes posibles; garantizándose de esta forma la obtención de características locales bien definidas.

- De preferencia utilizar imágenes de buena calidad y libre de ruido visual, de esta forma se consigue obtener el mayor rango de características distintivas en una imagen.

- El uso de un descriptor de imágenes es crucial para la recuperación de estos últimos, por lo que su elección es una de las fases más importantes para el desarrollo del *framework*.

- Si bien la cantidad de información contenida en una característica local depende de la calidad de la imagen de origen, esto no lo es todo. También es posible determinar qué tanta información se desea que una característica local contenga, mediante el establecimiento de un valor mínimo sobre el cual tenga que estar el valor del pixel promedio de cada característica local.

REFERENCIAS BIBLIOGRÁFICAS

Agustini Melchor, M., & Valiente Gonzáles, J. (2001). *Bases de datos para Multimedia: Recuperación por Contenido.* Valencia, España.

Bizagi. (5 de Diciembre de 2016). *Bizagi.com*. Obtenido de http://www.bizagi.com/es/

Bordignon, G. H. (2008). *Introducción a la Recuperación de Información.* Buenos Aires, Argentina: Universidad Nacional de Luján.

Cabello, R. (2014). Reflexiones sobre inclusión digital como modalidad de inclusión social. *VII Jornada de Sociología de la UNLP.* Ensenada, Argentina.

Cáceres, D. M. (2011). *Administración de Base de Datos.* Lima, Perú.

Cao, Y., Wang, H., Wang, C., Li, Z., Zhang, L., & Zhang, L. (2010). *MindFinder: Interactive Sketch-based Image Search on Millions of Images.* . Firenze, Italy.

Crespín Luis, J. C., & Julián García, R. A. (2014). *Sistema Detector de Somnolencia en Secuencias de Video de Conductores Manejando Usando Visión Computacional.* Trujillo: Universidad Nacional de Trujillo.

Curiel, R. L. (2014). *Las TIC en el aula de Tecnología. Guía para su aplicación a la metodología de proyectos.* United Kingdom : Lulu.com.

Dibujosinfantiles. (18 de Abril de 2016). *Dibujosinfantiles.org*. Obtenido de http://www.dibujosinfantiles.org/aprender-a-dibujar/dibujo-a-mano-alzada.php

doxygen. (22 de Agosto de 2016). *Open Source Computer Vision*. Obtenido de http://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html

Dúque, R. G. (2011). *Python para todos.* España: Creative Commons Reconocimiento 2.5 España.

Eclipse Fundation. (18 de Setiembre de 2016). *Eclipse*. Obtenido de https://eclipse.org/

Eitz, M., Hays, J., & Alexa, M. (18 de Setiembre de 2016). *How do humans sketch objects?* Obtenido de http://cybertron.cg.tu-berlin.de/eitz/projects/classifysketch/

Feijo, S. P. (2009). *Aplicación del Modelo Bag-of-Words al Reconocimiento de Imágenes.* Madrid, España: Universidad Carlos III de Madrid.

Galipienso, M. A. (2003). *Inteligencia artificial: modelos, técnicas y áreas de aplicación.* Madrid, España: Paraninfo.

García Cambronero, C., & Gómez Moreno, I. (2006). Algoritmo de aprendizaje: KNN & KMEANS. *Intelgencia en Redes de Comunicación*.

GazoPa Team. (22 de Abril de 2016). *Gazopa.com*. Obtenido de http://www.gazopa.com/

Hildebrand, K., Eitz, M., Boubekeur, T., & Alexa, M. (2011). *Sketch-Based Image Retrieval: Benchmark and Bag-of-Features Descriptors.* Visualization and Computer Graphics, IEEE Transactions.

Imagenet. (27 de Mayo de 2016). *Imagenet*. Obtenido de http://image-net.org/about-overview

Instituto de Tecnologías Educativas (ITE). (17 de Abril de 2016). *www.ite.educacion.es*. Obtenido de http://www.ite.educacion.es/formacion/materiales/86/cd/m2/bitmap.html

Instituto de Tecnologías Educativas (ITE). (17 de Abril de 2016). *www.ite.educacion.es*. Obtenido de http://www.ite.educacion.es/formacion/materiales/86/cd/m2/qu_es_la_imagen_digital.html

Instituto de Tecnologías Educativas (ITE). (17 de Abril de 2016). *www.ite.educacion.es*. Obtenido de http://www.ite.educacion.es/formacion/materiales/86/cd/m2/vectorial.html

Instituto de Tecnologías Educativas (ITE). (18 de Abril de 2016). *www.ite.educacion.es*. Obtenido de http://www.ite.educacion.es/formacion/materiales/86/cd/m2/el_pxel.html

iPixel Estudio. (21 de Abril de 2016). *iPixelEstudio.com*. Obtenido de http://www.ipixelestudio.com/bancos-imagenes-gratis/#Free_Photo

Langreiter, C. (21 de Abril de 2016). *labs.systemone.at*. Obtenido de http://labs.systemone.at/retrievr/about

Lew, M. S. (2013). *Principles of visual information retrieval.* USA: Springer Science & Business Media.

LK. (21 de Abril de 2016). *techjourney.net*. Obtenido de https://techjourney.net/search-online-photos-and-images-from-flickr-by-drawing-and-sketching-with-retrievr/

López, J. C. (7 de Abril de 2016). *Xataka*. Obtenido de http://www.xataka.com/robotica-e-ia/si-te-creias-bueno-al-pictionary-la-inteligencia-artificial-ha-llegado-para-que-no-ganes-nunca

Martin. (2 de Diciembre de 2016). *WinSCP: Free SFTP, SCP and FTP client for Windows*. Obtenido de https://winscp.net/eng/docs/lang:es

Marzal, A. &. (2003). *Introducción a la Programación con Python.* Castellón, España: Universitat Jaume I.

Melliy al Annamalai, R. j. (2000). Indexing Images in Oracle8i. *ACM SIGMOD Record*, 539-547.

Microsoft. (27 de Mayo de 2016). *Microsoft*. Obtenido de https://www.microsoft.com/es-es/download/details.aspx?id=29062

Mysen, C. C. (2011). *Washington, DC: U.S. Patente nº 7,987,185.*

Nora La Serna Palomino, W. C. (2010). Procesamiento Digital de textura: Técnicas utilizadas en aplicaciones actuales de CBIR. *Revista de Investigación de Sistemas e Informática*, 57-64.

OpenCV Developers Team. (28 de Mayo de 2016). *OpenCV*. Obtenido de http://opencv.org/about.html

Pascual, D. P. (2007). Algoritmos de agrupamiento . *Método Informáticos Avanzados.*

Piñar, B. A. (2012). *Detección automática de objetos en imágenes mediante el uso de puntos característicos.* Madrid, España: Universidad Autonoma de Madrid.

putty.org. (30 de Noviembre de 2016). *putty.org*. Obtenido de http://www.putty.org/

Q. Yu, Y. Y.-Z. (2015). *Sketch-a-Net that Beats Humans.* London, UK.

Rahman, M., Antani, S. K., & Thoma, G. R. (2011). *Biomedical CBIR using "Bag of Keypoints" in a Modified Inverted Index.* Bethesda, USA: IEEE.

Rosado Rodrigo, P., Figueras Ferrer, E., Planas Roselló, M., & Ferran Reverter, C. (2015). La imagen toma la palabra: Construcción de un vocabulario visual. *2nd Art, Science, City International Conference ASC2015*. Valencia, España: Universitat Politècnica de València.

Sampedro, J. (19 de Abril de 2016). *elpais.com*. Obtenido de http://elpais.com/elpais/2016/01/26/ciencia/1453809513_840043.html

T. Chen, M.-M. C. (19 de Abril de 2016). *cg.cs.tsinghua.edu.cn*. Obtenido de http://cg.cs.tsinghua.edu.cn/montage/main.htm

Tao Chen, M.-M. C.-M. (2009). *Sketch2Photo: Internet Image Montage.* Pekín. Recuperado el 19 de Abril de 2016, de http://cg.cs.tsinghua.edu.cn/papers/SiggraphAsia_2009_sketch2photo.pdf

The CentOS Project. (2 de Diciembre de 2016). *CentOS*. Obtenido de https://www.centos.org/

Valdivia, D. P. (11 de Abril de 2016). *Revista Cubana de Informática Médica*. Obtenido de http://www.rcim.sld.cu/revista_3/articulos_html/articulo_pedro.htm#t

Vega, I. R. (2010). *Operaciones auxiliares con tecnologías de la información y la comunicación.* Madrid, España: Paraninfo.

Veltkamp, R. B. (2013). *State-of-the-art in content-based image and video retrieval* (Vol. 22). USA: Springer Science & Business Media.

Wan, J. W. (2014). Deep learning for content-based image retrieval: A comprehensive study. *Proceedings of the ACM International Conference on Multimedia*, 157-166.