PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ADAPTACIÓN DE LOS ALGORITMOS SIFT Y LSH PARA LA DIFERENCIACIÓN DE ARCHIVOS DE IMÁGENES

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

Tania Gabriela Ramirez Franco
Ila Ibañez Quispe

ASESOR: Mg. Claudia Zapata del Rio

Lima, Mayo de 2016

Resumen

El almacenamiento digital de información se ha vuelto un proceso cotidiano para todo aquel que disponga de algún dispositivo electrónico. Al tratarse de un proceso tan frecuente, es muy común que se almacenen grandes cantidades de datos/información, volviéndose ardua su administración. Esto aplica a todos los tipos de datos digitales.

El presente proyecto se enfoca en los problemas de almacenamiento de archivos de imágenes, como la gran repetición de archivos, elaborando una solución que permita aminorar el problema. El objetivo del proyecto es construir una herramienta que facilite la búsqueda de archivos de imagen que contengan contenidos similares. Para lograr el objetivo, se evaluaron herramientas que permitieran manipular la información de los archivos de imagen de manera que se puedan obtener los datos necesarios para realizar un proceso de comparación. Se decidió utilizar las herramientas SIFT y LSH y se procedió a adecuarlas para su funcionamiento de acuerdo a los criterios establecidos durante la investigación.

Finalmente, se pudo elaborar una solución que permite realizar la comparación de un grupo de imágenes, mostrando porcentajes de similitud entre estas para así poder saber que imágenes son similares entre sí.

En el primer capítulo del presente documento se desarrolla el problema a tratar y se explican los términos que se utilizan a lo largo de todo el documento. En el siguiente capítulo se encuentran los objetivos de la tesis, así como los resultados que se pretende obtener y las herramientas que se utilizaron para la elaboración de la solución. En los capítulos siguientes, se desarrollan uno por uno los objetivos alcanzados y en el último capítulo se encuentran las conclusiones y comentarios sobre el trabajo realizado.

FACULTAD DE
CIENCIAS E INGENIERÍA
ESPECIALIDAD DE
INGENIERÍA INFORMÁTICA

PONTIFICIA
UNIVERSIDAD CATÓLICA
DEL PERÚ

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: **ADAPTACIÓN DE LOS ALGORITMOS SIFT Y LSH PARA LA DIFERENCIACIÓN DE ARCHIVOS DE IMÁGENES**

ÁREA: CIENCIAS DE LA COMPUTACIÓN

PROPONENTE: Ing. Rosalia Caya

ASESOR: Mg. Claudia Zapata Del Río
Ing. Rosalia Caya

ALUMNO: Ila Ibañez Quispe
Tania Gabriela Ramírez Franco

CÓDIGO: 20062329
20064580

TEMA N°: 617

FECHA: San Miguel, 03 de diciembre del 2015

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
Facultad de Ciencias e Ingeniería
04 DIC. 2015
MIGUEL MEJIA PUENTE
DECANO

DESCRIPCIÓN

Debido a que es posible obtener una gran variedad de información desde diferentes tipos de fuentes (blogs, galerías, bibliotecas digitales, entre otros) en algunos casos, suelen recopilarse grandes cantidades de archivos ya sea en computadoras, u otros dispositivos de almacenamiento, pudiendo llegar al punto de tener grandes volúmenes de información con data repetida o muy similar.

Un caso particular de esta situación es la de los archivos de imágenes, que por su naturaleza almacenan más detalles que otros archivos, ocupando mayor espacio. Asimismo, estos son muy frecuentes en la red y pueden ser creados por uno mismo de forma sencilla, como por ejemplo las fotografías digitales.

Esta acumulación de información origina diversas dificultades, dependiendo tanto de la magnitud de datos como de la importancia que ellos tengan. Uno de los problemas que se generan es el desperdicio de espacio de almacenamiento, pues se puede tratar de una gran cantidad de archivos o que estos sean muy pesados (o una combinación de ambas situaciones).

Otro problema que se presenta es que la realización de acciones sobre estos conjuntos de archivos se vuelve una tarea tediosa, pues al haber tantos archivos, el rendimiento del dispositivo donde son almacenadas disminuye. También se pueden presentar problemas al realizar copias de seguridad, ya que si se tiene información duplicada esta también seria respaldada, lo que sería innecesario y conllevaría a incrementar los costos relacionados al almacenamiento y al tiempo ocupado en realizar el respaldo.

Por otro lado, la edición y alteración de la información digital es un tema en constante crecimiento. En el caso particular de la manipulación imágenes digitales, ésta se ha vuelto más sencilla debido al constante desarrollo de nueva tecnología tanto software

como hardware. Muchas veces la edición es tan sutil que los cambios no son perceptibles a simple vista o no se puede determinar un nivel de similitud, por lo que la comparación de imágenes de manera manual se hace poco precisa.

Debido a los problemas antes mencionados, vemos que es necesaria la utilización de una herramienta que permita determinar con cierto nivel de precisión la similitud de dos o más archivos de imágenes.

Nos enfocamos en las imágenes en formato bmp (mapa de bits), JPG (y JPEG), GIf, Png y Tiff debido a que son los formatos más comunes usados en el diseño web y fotografía.

Para el desarrollo de esta herramienta se han seleccionado los siguientes algoritmos:
Algoritmo SIFT (Scale-invariant feature transform)
Este algoritmo, publicado por David Lowe en 1999, permite la detección de características en las imágenes a través de los "Puntos de interés" que se obtienen como resultado. Estos Puntos de interés contienen información de la imagen y además son independientes de la rotación o magnitud de la misma, de manera que pueden ser comparados con puntos de interés de otras imágenes fácilmente.

Algoritmo LSH (Locality-sensitive hashing)
Propuesto por Indyk y Mot-wani, es una técnica de búsqueda de similitud aproximada que no presenta problemas al trabajar con grandes cantidades de data.
Básicamente, lo que hace es obtener un hash de los elementos y los ordena utilizándolos como índices, de manera que items similares sean agregados dentro de la misma fila.

OBJETIVO GENERAL

Adaptación de los algoritmos SIFT y LSH para la construcción de una herramienta que permita la diferenciación de archivos de imágenes para su posible depuración u ordenamiento.

OBJETIVOS ESPECÍFICOS

- Modelar las estructuras de datos necesarias para poder comparar las imágenes de los formatos: jpg, png, gif y bmp.
- Adaptar el algoritmo SIFT para la identificación de los puntos relevantes de las imágenes (keypoints) de los tipos seleccionados y permitir la comparación euclidiana.
- Adaptar el algoritmo LSH para ordenar los keypoints de acuerdo a sus semejanzas y así comparar más de dos archivos a la vez.
- Implementar una herramienta que uniendo las adaptaciones de los algoritmos mencionados en los objetivos anteriores permita comparar más de un archivo de imágenes.
- Probar y medir los resultados que produce la herramienta.

ALCANCE

Se espera llegar a adaptar las herramientas para su correcto funcionamiento en conjunto. Luego de esto se creará un programa sencillo para demostrar el funcionamiento de la adaptación realizada.

Dado que no se encuentra mucha información acerca de algoritmos que realizan una función similar se buscará utilizar una versión implementada de SIFT y una adaptación en JAVA de LSH para que en conjunto permitan resolver el problema pudiendo instalarse en plataformas con distintas características.

Esta solución es orientada a un uso cotidiano y que no implique la revisión de detalles mínimos.

Los subsistemas que comprenden el proyecto y el tesista responsable son:

- Subsistema de conversión de imágenes a formato intermedio (pgm) para SIFT: Tania Ramírez
- Subsistema de lectura de archivos para obtención de puntos clave (keypoint) en la imagen utilizando SIFT: Ila Ibañez
- Subsistema de estructuración de imágenes en función a los keypoint utilizando LSH: Tania Ramírez
- Subsistema de comparación en función a los keypoint obtenidos: Ila Ibañez

Máximo: 100 páginas

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Generalidades

En este capítulo se definirán a profundidad el problema a abordar a lo largo del proyecto y los conceptos y definiciones básicas para su correcta comprensión.

1.1. Problemática

Gracias a la popularidad de los medios digitales el intercambio de información se hace cada vez más sencillo y los días de ir a la biblioteca o sacar copias se hacen más escasos. Esta tendencia se basa en la facilidad, rapidez y mayor eficiencia que brindan estos medios.

Debido a que es posible obtener una gran variedad de información desde diferentes tipos de fuentes (blogs, galerías, bibliotecas digitales, entre otros) en algunos casos se suelen recopilar grandes cantidades de archivos ya sea en computadoras u otros dispositivos de almacenamiento, pudiendo llegar al punto de tener grandes volúmenes de información conteniendo data repetida o muy similar.

Un caso particular de esta situación es la de los archivos de imágenes, que por su naturaleza almacenan más detalles que otros archivos por lo que ocupan mayor espacio, además este tipo de archivo es muy común ya que se encuentran fácilmente en la red y pueden ser creados por uno mismo de forma sencilla (como por ejemplo las fotografías digitales)

Esta acumulación de información origina diversas dificultades, dependiendo de la magnitud de los datos y de la importancia que ellos tengan. Uno de los problemas

que se generan es el desperdicio de espacio de almacenamiento, pues se puede tratar de una gran cantidad de archivos o que estos sean muy pesados (o también una combinación de ambos).

Otro problema que se presenta es que el realizar acciones sobre estos conjuntos de archivos se vuelve una tarea tediosa, pues al haber tantos archivos el rendimiento del dispositivo donde son almacenadas disminuye.

También se pueden presentar problemas al realizar copias de seguridad, ya que si se tiene información duplicada esta también seria respaldada, lo que sería innecesario y conllevaría a incrementar los costos relacionados al almacenamiento y al tiempo ocupado en realizar el respaldo.

Por otro lado, la edición y alteración de la información digital es un tema en constante crecimiento. En el caso particular de la manipulación imágenes digitales, ésta se ha vuelto más sencilla debido al constante desarrollo de nueva tecnología tanto en software como hardware. Muchas veces la edición es tan sutil que los cambios no son perceptibles a simple vista o no se puede determinar si efectivamente son diferentes o no, por lo que la comparación de imágenes de manera manual se torna poco precisa.

Debido a los problemas antes mencionados, vemos que es necesaria la utilización de una herramienta que permita determinar con cierto nivel de precisión la similitud de imágenes, para ello se plantea una solución que realice la comparación de un grupo de imágenes. Para lograr este objetivo se planea hacer uso de herramientas más sencillas que permitan extraer información de las imágenes, para posteriormente unirlas y lograr una herramienta más completa. Esta herramienta permitiría que el usuario pueda tomar decisiones sobre la data evaluada, como por ejemplo, poder depurar imágenes que sean iguales o muy similares o poder reorganizar las imágenes basándose en los resultados obtenidos (agrupando imágenes iguales, o por nivel de similitud, etc.).

Para el desarrollo de esta herramienta se han analizado primero los tipos de imágenes digitales existentes, su forma de almacenamiento y sus principales características, a continuación se muestra la información encontrada al respecto.

1.2. Marco Conceptual

Para lograr el completo entendimiento del presente proyecto, en esta sección se presentan conceptos relevantes y detallados utilizados en su desarrollo.

Pixeles: Elemento más pequeño de los que componen una imagen digital [2].

1.2.1. Clasificación de los archivos de imagen

Los archivos de imágenes se pueden clasificar dependiendo de la forma en que son almacenados, en imágenes vectoriales e imágenes de mapas de bits.

- **Imagen de Mapa de Bits(ó Raster):** La imagen se expresa mediante una matriz de pixeles, donde cada cuadricula tiene un color en particular. La resolución de estas imágenes esta determinada por la cantidad de pixeles almacenados, por ello cuando se le hace alguna transformación genera una perdida de información (aliasing).

  Esta forma de almacenamiento permite que los archivos no dependan de la aplicación con la que fueron creados, ya que casi todas las aplicaciones que procesan imágenes de pixeles pueden leer diversos formatos.

  Estas imágenes se expresan en formatos como: Bmp, psd, jpeg, gif, tif, entre otros.

- **Imagen Vectorial:** Este tipo de imágenes se expresa mediante un conjunto de formas vectoriales[l], que se almacenan en un archivo junto con sus vectores componentes, posición y propiedades.

  Los archivos que almacenan la información son muy pequeños puesto que la imagen se define con formulas matemáticas; cuanto más trazos tenga el archivo mayor sera su tamaño. Una de sus características principales es que son independientes de la resolución, ya que al estar definidas por puntos o nodos son escalables sin perdida de calidad.

[l] Descripción geométrica (matemática) de una imagen [3]. La imagen no es un conjunto de pixeles sino es una formula que al ser interpretada da forma a la imagen. Por tanto, a pesar de cambiar el tamaño de la imagen esta no se verá alterada.

Los formatos de vectores están muy ligados al tipo de software que se utiliza para crearlos o interpretarlos, y aunque existen maneras de convertir de un formato a otro, siempre existe pérdida de información en dicha conversión. Una imagen vectorial puede convertirse a una imagen de pixeles mediante un proceso de renderización. Los formatos más utilizados para imágenes de este tipo son: AI(illustrator), CDR(coreldraw), DXF(autocad) y SVG.

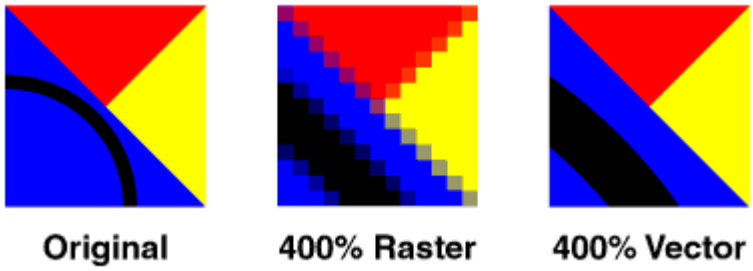


Figura 1.1: Variación de resolución para una imagen almacenada en mapa de bits y en forma vectorial. [3]

En este trabajo nos enfocaremos en el manejo de imágenes de mapa de bits, ya que son muy similares independientemente del formato en que sean almacenados, almacenan información real en sus píxeles(color, resolución, forma) y son las que se generan con mayor frecuencia mediante medios digitales como cámaras y escáneres.

En cambio las imágenes vectoriales son muy dependientes de la aplicación en que se crearon, por lo que para compararlas, primero se debería validar su extensión, y si no son iguales tratar de homogenizarlas (proceso por el cual se perdería información).

A continuación se profundizará en las características y formatos de las imágenes de la clase mapas de bits.

1.2.2. Características de los archivos de imagen

- **Tamaño de Imagen:** Se refiere a la dimensión de la imagen y se determina por el número de píxeles a lo largo y ancho de la misma.

- **Tamaño de archivo:** Representa la cantidad de información que contiene un archivo ya que depende de las características propias de la imagen como son sus dimensiones (tamaño de imagen), la carga de color, su resolución, entre otros. Este se mide en bits o en alguno de sus múltiplos.
- **Resolución de Imagen:** Es la medida de cantidad de píxeles por unidad de longitud. Se deduce que a mayor resolución, mayor será la calidad de la imagen y por ende mayor será el tamaño de archivo de la misma.
- **Profundidad en bits:** Es la cantidad de bits necesarios para representar el color de un píxel en la imagen.

1.2.3. Formatos de archivos de imagen

A continuación se listan los formatos de imagen más comunes y utilizados:

- **Mapa de bits (extensión bmp):** Este formato almacena la imagen mediante una grilla de píxeles, para formar la imágen por cada píxel se registra una carga de color. Dado que generalmente no tienen compresión son archivos de gran tamaño y pueden mostrar un buen nivel de calidad. El principal inconveniente de este formato es que al ampliar la imagen, esta se distorsiona perdiendo su nitidez y resolución (a este tipo de pérdida se le conoce como pixelización y sucede cuando los píxeles de una imagen son demasiado evidentes al ojo humano)
- **Formato JPEG/JPG (Joint Photographic Experts Group):** Es uno de los formatos mas populares y más utilizados tanto en internet como en los dispositivos de captura de imagen, debido sobretodo a que las imágenes en este formato ocupan poco espacio y se pueden almacenar sin problemas, con cierto detalle y sin representar pérdida de calidad notoria. En términos generales, lo que este formato permite es graduar el nivel de compresión (basado en la reducción de información, promediando zonas de degradado de las imágenes), pudiéndose de este modo decidir entre: guardar una imagen en baja calidad pero de un menor tamaño o una imagen de alta calidad con un mayor peso.

- **Formato GIF (Graphic Interchange Format):** Este formato fue creado con el objetivo de obtener archivos muy pequeños, por ello utiliza un protocolo de compresión llamado LSW y solo permite usar 256 colores. Hoy en día se utiliza para almacenar imágenes simples como logotipos, o animaciones cortas (soporte de imágenes múltiples). y por su bajo peso y compatibilidad con diferentes plataformas es utilizado extensamente en Internet.

- **Formato PNG (Portable Network Graphics):** Formato de imagen de mapa de bits y codec de video que utiliza la compresión sin perdida y que soporta diferentes modelos de color de imagen además de soportar transparencias. Este formato permite realizar compresiones sobre las imágenes sin representar pérdida de calidad. Inicialmente fue diseñado para reemplazar a los formatos GIF y TIFF, puesto que tiene ciertas ventajas sobre ellos (en calidad, compresión y en su momento precio) a pesar de no soportar todas sus características (por ejemplo, PNG no tiene la opción de soporte de imágenes múltiples que tiene GIF).

Los mencionados son los formatos de imágenes que se utilizarán en nuestra solución. En capítulos posteriores se detallará sus estructuras de datos.

Así mismo para lograr el desarrollo de la herramienta se han revisado los siguientes conceptos relacionados al manejo de imágenes:

A. Hash Criptográfico (Cryptographic Hash)

Procedimiento determinístico que toma un bloque de datos de manera arbitraria y devuelve una cadena de bits de tamaño fijo. Las funciones de este tipo suelen tener mucha información para aplicaciones de seguridad. También suelen usarse como funciones de hash ordinarias, para indexar tablas de datos, de huellas dactilares y para detectar data duplicada o archivos de identificación única.

B. Edge detection (detección de bordes)

Operación que permite la detección de los 'bordes' dentro de una imagen, entendiéndose por bordes a los cambios abruptos en la intensidad del color de una imagen. Esta operación es comúnmente utilizada en el análisis de imágenes. Existen diversos

métodos que permiten la realización de esta operación, desde los métodos mas antiguos hasta los del actual estado del arte. En la figura 1.2 se aprecia a la izquierda la imagen original y a la derecha la misma imagen después de aplicado el Edge detection.

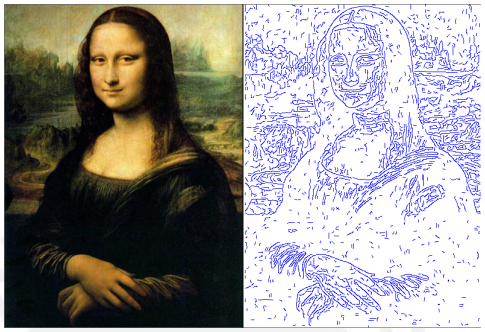


Figura 1.2: Mona Lisa antes y después de aplicado el Edge detection [Elaboración Propia]

C. Laplaciano de Gauss

Se trata de una medida que resalta las regiones que presentan un rápido cambio de intensidad en la imagen, por lo que se le suele usar para la detección de bordes. Se usa generalmente en imágenes previamente suavizadas[II] para reducir el ruido. Se le conoce también como Laplaciano, LoG(Laplacian of Gaussian), Marr Filter. [5]

D. Diferencia de Gausianas

Se conoce así a la sustracción de dos difuminados de una imagen, uno menos difuminado que el otro. Esta técnica también se suele utilizar para poder determinar los bordes dentro de una imagen

E. SIFT (Scale-invariant feature transform)

Algoritmo desarrollado por David Lowe que permite la detección de características en las imágenes haciendo posible la obtención de sus “Puntos de interés” (Keypoints), estos contienen mucha información además de ser independientes de la rotación o

[II]Suavizar.- Crear una función aproximada que intenta capturar patrones importantes en la data, reduciendo así el ruido en esta. [4]

magnitud de la imagen. Los keypoints son representados como vectores por lo cual es posible comparar imágenes con características distintas con bajo margen de error.

En la imagen 1.3 se muestra la aplicación del SIFT a la ilustración de un castor, mostrándose los vectores mas resaltantes.

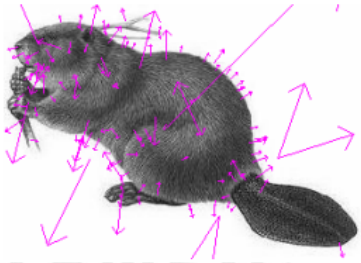


Figura 1.3: Ejemplo de aplicación del SIFT. Lineas representan áreas importantes [6]

Como se puede apreciar, este algoritmo no realiza una comparación de imágenes, sin embargo en el desarrollo del algoritmo se han utilizado sus resultados para poder realizar esta acción.

Dada su importancia en el desarrollo de la solución, este algoritmo se explicara con mayor detalle en capítulos posteriores.

F. LSH (*Locality Sensitive Hashing*)

De acuerdo a Andoni et al. [7], es la función que permite la resolución de la búsqueda de Vecino más Cercano, teniendo la ventaja de ser eficiente aún si la búsqueda se realiza sobre data muy extensa.

Resulta ser un algoritmo muy complejo, pues utiliza diversas variables matemáticas, sin embargo, puede dar una solución bastante acertada.

Esta solución se basa en la premisa de que dos puntos (‘p’ y ‘q’), similares y cercanos, guardan la misma relación después de ser proyectados por una función. Como se muestra en la figura 1.4, se aprecia que la cercanía entre los 2 puntos rojos, se mantiene tanto en la esfera, como en su proyección en el plano (en este caso el presente documento). Además, se puede ver que en (b) la relación se mantiene a pesar de que esta proyección es posterior a la rotación la esfera.

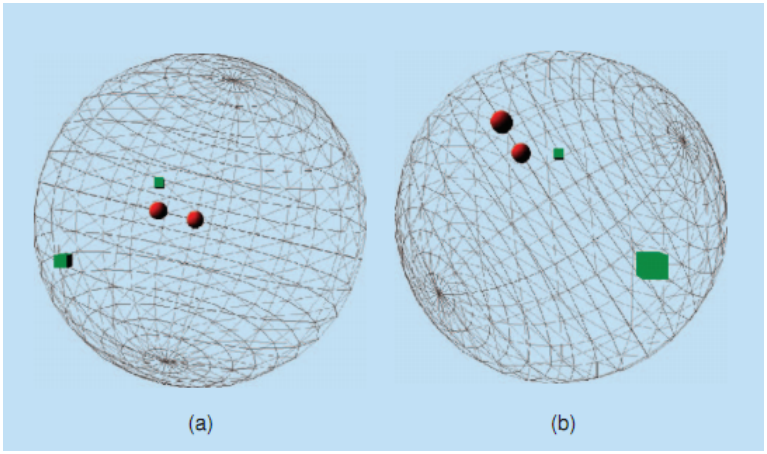


Figura 1.4: LSH: la distancia entre dos puntos cercanos se mantiene luego de su proyección en el plano [8]

Este es el concepto empleado por el LSH, ya que aplica funciones hash[III] a un objeto con la finalidad de obtener una expresión más sencilla que permita una comparación simple.

1.3. Estado del Arte

El Estado del Arte es un concepto que se usa comúnmente para referirse al conocimiento disponible hasta el momento en torno a un tema determinado. Dentro de la tradición de la investigación este es un trabajo de revisión que debe hacerse antes de iniciar un proyecto con la finalidad de que el conocimiento que se produzca tenga un carácter novedoso; el revisar lo que se ha trabajado en torno a un tema específico, ver las herramientas metodológicas empleadas, analizar las conclusiones alcanzadas y las sugerencias propuestas, permiten dar un paso adelante en la acumulación del conocimiento desde una perspectiva académica.

En el presente caso se ha realizado una investigación tradicional de la literatura actual en la que se busca mostrar la información recabada respecto de las técnicas, algoritmos y herramientas existentes para la comparación de archivos.

[III] Función que relaciona estructuras complejas asignandoles identificadores similares y fácilmente comparables conocidos como hashes.

1.3.1. Técnicas de comparación

La comparación de archivos de imagen se viene tratando desde hace ya un tiempo, y se han propuesto diversas soluciones para este problema. Sin embargo, dichas soluciones no son siempre eficientes.

A continuación se describen brevemente algunos algoritmos y herramientas usados para la comparación de archivos de imagen.

1.3.1.1. Comparación Binaria

Esta comparación ofrece un resultado exacto es decir permite identificar si los archivos son completamente idénticos o no. La comparación se realiza analizando byte por byte los archivos, garantizando de esta forma que el resultado obtenido sea 100 % confiable. Al ser una comparación que recorre todo el archivo, este método se vuelve lento a medida de que el tamaño de los archivos se incrementan, por ello este método solo sería eficiente para el caso de comparaciones de pocos archivos de tamaño moderado

Puesto que se trata de una comparación byte por byte, ésta se puede realizar también a través de hashes, creando un hash para cada una de las imágenes y se comparan dichas cadenas. Esta forma es mucho más rápida pero sigue siendo poco eficiente, pues solo identifica si los archivos son idénticos o no.

1.3.1.2. CRC Comparison

La comprobación de redundancia cíclica (cyclic redundancy check - CRC) es un método de comparación exacto, que permite identificar archivos posiblemente iguales. La comparación se realiza por medio del CRC, que es el valor en 4 bytes del checksum del contenido del archivo. El checksum se obtiene analizando el contenido de un archivo byte por byte, y su principal característica es que dos archivos distintos difícilmente generarán un valor checksum igual, por lo que los resultados de esta comparación no son fiables del todo. Al tener que leer el archivo byte por byte, se vuelve poco eficiente.

1.3.1.3. SURF (*Speeded Up Robust Features*)

Algoritmo de detección de características (puntos de interés), presentado por Herbert Bay en el 2006 y que puede ser utilizado para tareas de visión artificial, como detección y reconocimiento de objetos o reconstrucción 3D. Esta parcialmente inspirada por el descriptor del SIFT y se dice que su version estandar es mucho más rápida que el SIFT.

1.3.1.4. Características de otros algoritmos investigados

Además de los algoritmos que se evaluaron para formar parte de la solución, se revisaron otros algoritmos que también se enfocaban en el procesamiento de imágenes, tales como:

- Segmentación de imágenes (dividir la imagen en segmentos de acuerdo al contenido de esta) [9]
- Restauración de imágenes sin usar métodos clásicos o estimaciones estadísticas [10]
- Realce de imágenes naturales [11]
- Librerías de imágenes y como recuperar una imagen de estas basándose en preferencias de usuarios [12]
- Mejor calidad en acercamiento de imagen (*zooming*) [13]

Se observa que se ha tocado el tema con mucha frecuencia pero aun sigue sin presentarse una solución óptima y sencilla al problema de acumulación de data.

1.3.2. *Intel Paper: Efficient Near-duplicate Detection and Sub-image Retrieval*

Paper realizado con el respaldo de Intel [14], con el objetivo de desarrollar un sistema que permita la identificación de imágenes similares, y contenidas dentro de otras imágenes.

Los conceptos aplicados en este sistema son los de local descriptor, mediante la aplicación del algoritmo Lowe's Difference of Gaussian; locality - sensitive hashing, combinados con optimizaciones de acceso eficiente al disco, esto último para incrementar la eficiencia, en términos de tiempo, del sistema. La idea de desarrollo de este sistema es tener una gran base de datos contra la cual se puedan hacer comparaciones de otras imágenes, esto con el objetivo de hallar violaciones de derechos de autor (*Copyrigth*) e imágenes alteradas.

1.3.3. Aplicativos y usos

A raíz de nuestra investigación, hemos podido encontrar diversas aplicaciones que realizan la tarea de comparación de archivos de imágenes, audio, texto, entre otros, pero cuyos resultados no satisfacen del todo a los usuarios, ya que no les ofrecen todas las opciones que ellos esperan.

De los aplicativos encontrados podemos mencionar:

- Aquellos que solo se centran en la comparación de 2 archivos que se proporcionan como entrada.
- Aquellos que comparan archivos dentro de una ruta dada.
- Los que ofrecen comparaciones con estadísticas.
- Aquellos que permiten, a través del programa, la eliminación de los archivos duplicados encontrados.
- Aquellos que permiten combinar archivos similares encontrados (solo para el caso de los archivos de texto)

A. Tabla de Comparación

Se compararon los softwares encontrados obteniéndose la siguiente tabla:

También hemos descubierto que, la mayoría de software de comparación de archivos, independientemente del tipo de archivo que se compare, tienen licencia propietaria, siendo muy pocos los que tiene licencia GPL y menos aun los que tiene

Cuadro 1.1: Tabla de Comparación

Software	Soporta diferentes formatos	Selección de Carpetas	Vista previa de archivos	Acciones sobre archivos	Ajuste de nivel de similitud	Licencia
Duplicate file finder	Si	Si	Si	No	Si	Libre
Image Comparer	Si	Si	Si	No	No	Propietario

licencia Libre. Ya entrando más en el tema, son pocos los que comparan archivos de imágenes dentro de un determinado directorio.

Por tanto, ya sea por que las aplicaciones no ofrecen todo lo que el usuario necesita, o porque su uso sea muy complejo, o que su adquisición es muy costosa, no se ha encontrado un aplicativo que llegue a satisfacer ampliamente las necesidades de los usuarios.

1.3.4. Entorno nacional

Dentro de nuestro ámbito nacional, el tema también se ha ido tratando aunque a una menor escala. Por ejemplo, en uno de los cursos de nuestra universidad se hizo la implementación un algoritmo similar al propuesto, pero para la comparación de archivos de texto. También se han encontrado algunas tesis referentes al tema, como la de un algoritmo para comparación de imágenes dactilares [15], presentada por un alumno de la facultad de Electrónica en el 2011. Finalmente, también se han encontrado diversos cursos de especialidad acerca del tema, como un curso de maestría que se dicta en nuestra universidad de “Procesamiento de Imágenes Digitales”.

Los algoritmos mencionados (CRC1.3.1.2 y Comparación binaria 1.3.1.1) son métodos exactos de comparación, cuya función es la establecer si 2 imágenes son idénticas o no. Sin embargo el objetivo del proyecto es determinar una similaridad entre las imágenes procesadas, no siendo necesario que estas sean exactamente iguales, por lo cual se debe realizar un análisis mas riguroso de las imágenes. Y el algoritmo SURF1.3.1.3 por otro lado, presta muchas más opciones de las necesarias para el proyecto. Es por tanto que se propone utilizar el conjunto de herramientas

SIFT (E) y LSH (F) para lograr el cometido mencionado de acuerdo a los objetivos establecidos.

A continuación algunos ejemplos del empleo de las herramientas seleccionadas:

- Ambas herramientas son utilizadas por Intel, para la creación de un sistema de detección de casi-duplicados (“near-duplicate”, imágenes alteradas con transformaciones comunes) y reconocimiento de sub-imágenes (imágenes incrustadas dentro de otras imágenes.) [14]

- El algoritmo SIFT es utilizado por el Departamento de Ciencias de la Computación de la Universidad de Columbia Británica para la elaboración de un algoritmo de localización y mapeo para un robot móvil. [16]

- El VisualRank, algoritmo para la búsqueda de imágenes propuesto por el equipo de Google, utiliza ambas herramientas [17]

- El Departamento de Ciencias de la Computación de la Universidad de Columbia Británica también utiliza el algoritmo SIFT para la construcción automática de panoramas. [18]

- El LSH se propone en el artículo de Hisashi Koga et al. [19] para el mejoramiento del método de Agrupamiento jerárquico

Capítulo 2

Objetivos y Alcances

En este capítulo se presentarán los objetivos de la tesis con sus correspondientes resultados esperados. También, se presenta un resumen de las herramientas a utilizar, así como el alcance y la justificación de la solución.

2.1. Objetivo general

Adaptación de los algoritmos SIFT y LSH para la construcción de una herramienta que permita la diferenciación de archivos de imágenes para su posible depuración u ordenamiento.

2.2. Objetivos específicos

- O1: Modelar las estructuras de datos necesarias para poder comparar las imágenes de los formatos: jpg, png, gif y bmp.
- O2: Adaptar el algoritmo SIFT para la identificación de los puntos relevantes de las imágenes (keypoints) de los tipos seleccionados y permitir la comparación euclidiana.
- O3: Adaptar el algoritmo LSH para ordenar los keypoints de acuerdo a sus semejanzas y así comparar más de dos archivos a la vez.

- O4: Implementar una herramienta que uniendo las adaptaciones de los algoritmos mencionados en los objetivos anteriores permita comparar más de un archivo de imágenes.
- O5: Probar y medir los resultados que produce la herramienta.

2.3. Resultados esperados

- R1 (O1): Estructuras de datos que representan los datos necesarios para comparar archivos de imágenes de los formatos: jpg, png, gif y bmp.
- R2 (O2): Módulo de comparación uno a uno implementado.
- R3 (O3): Módulo de comparación de más de dos imágenes implementado.
- R4 (O4): Herramienta que unifica los módulos de comparación implementada.
- R5 (O5): Pruebas y validación de resultados obtenidos.

2.4. Herramientas, métodos y procedimientos

A continuación se describirán las herramientas utilizadas para la construcción de la solución.

2.4.1. Lenguaje

Para poder elegir el lenguaje de programación a utilizar, se tuvieron en cuenta los siguientes puntos:

- Compatibilidad con varios sistemas operativos
- Velocidad de ejecución
- Bajo consumo de memoria
- Bajo consumo de procesador
- Curva de aprendizaje del lenguaje

- Disponibilidad del lenguaje (si es de pago o libre)

Finalmente se llego a la conclusión que el lenguaje 'Java' era el mas adecuado para las necesidades de la solución.

2.4.2. Mapeo

A continuación se presentan los resultados esperados con las herramientas a utilizarse para lograrlos.

Resultados esperados	Herramientas a usarse
R1 (O1): Estructuras de datos que representan los datos necesarios para comparar archivos de imágenes de los formatos: jpg, png, gif y bmp. R2 (O2): Módulo de comparación uno a uno implementado	- ImageMagick (se utilizan algunos módulos de esta librería para poder hacer las conversiones de imágenes) - Desarrollo propio en lenguaje Java - SIFT (herramienta que se utiliza para la obtención de los keypoints de las imágenes)
R3 (O3): Módulo de comparación de más de dos imágenes implementado.	- Adaptación de algoritmo LSH (herramienta que permitirá comparar y organizar las imágenes de manera rápida y eficiente)
R4 (O4): Herramienta que unifica los módulos de comparación implementada.	Desarrollo de interfaces para la unificación de todos los módulos.
R5 (O5): Pruebas y validación de resultados	Experimentación numérica y validación de resultados

Cuadro 2.1: Mapeo

Como se puede observar, las herramientas más utilizadas son el algoritmo SIFT y el LSH, para los módulos de comparación.

2.4.3. Herramientas

A continuación se muestran las herramientas utilizadas en el presente proyecto.

2.4.3.1. ImageMagick

El ImageMagick es una suite de software que permite la edición, creación, composición y conversión de imágenes en mapa de bits, pudiendo leer y escribir imágenes en diversos formatos. Esta herramienta además, es multiplataforma y tiene distribuciones en diferentes lenguajes.

También se utilizarán las herramientas **SIFT** (E) y **LSH** (F), las cuales ya han sido mencionadas en el capitulo anterior.

2.5. Alcance

Se espera llegar a adaptar las herramientas para su correcto funcionamiento en conjunto. Luego de esto se creará un programa sencillo para demostrar el funcionamiento de la adaptación realizada.

2.5.1. Limitaciones

Algunas de las limitaciones que se presentan para el proyecto son:

- Poca información sobre algoritmos similares.- No se encuentra mucha información acerca de algoritmos que realizan una función similar. Es más común encontrar aplicaciones que ya los implementen.
- Pocas herramientas multiplataforma.- No se encuentran muchas herramientas que nos ayuden para el proyecto y que sean multiplataforma.
- Diversidad de lenguajes.- Muchas de las herramientas encontradas están en diferentes lenguajes de programación, haciendo complicada su interacción.

2.5.2. Riesgos

2.6. Justificación y viabilidad

Se presenta a continuación la viabilidad del proyecto desarrollado.

Riesgo identificado	Impacto en el proyecto	Medidas correctivas para mitigar
Avance de las tecnologías en el tiempo	Las herramientas que se utilizan pueden quedar obsoletas para algunos sistemas	Se diseña una solución que no sea muy dependiente del sistema en el que es desarrollado y/o ejecutado
Los resultados no sean los mas satisfactorios	Baja confiabilidad de la solución elaborada	Se realizan suficientes pruebas para poder medir la confiabilidad de la solución

Cuadro 2.2: Riesgos

2.6.1. Justificación del proyecto de tesis

Este proyecto vendrá a ser, en primera instancia, una gran ayuda para los usuarios que deseen depurar imágenes sin tener que pasar por la tediosa tarea de revisar una por una.

Adicionalmente, al tratarse de una solución de forma algorítmica, esta podrá se adaptada para diversos fines, y no solo para cumplir el objetivo inicial que se presenta en el proyecto. La solución se trabaja de manera que permite que se puedan integrar nuevos módulos sin dificultades.

2.6.2. Análisis de viabilidad del proyecto de tesis

A continuación se analiza la viabilidad del proyecto desde las perspectivas técnicas y económica.

Viabilidad Técnica.- Los módulos implementados han sido elaborados en lenguaje C, puesto que es un lenguaje de alto nivel y es utilizado en diversas plataformas.

Viabilidad Económica.- Todas las herramientas utilizadas en el presente proyecto son de uso gratuito y/o académico, por lo que no se corre con ningún gasto por estos.

2.7. Posibles Aplicaciones Prácticas de la solución

Esta solución es orientada a un uso no profesional, es decir, para un uso mas cotidiano y que no implique la revisión de detalles mínimos.

2.8. Resumen de la elaboración de la solución

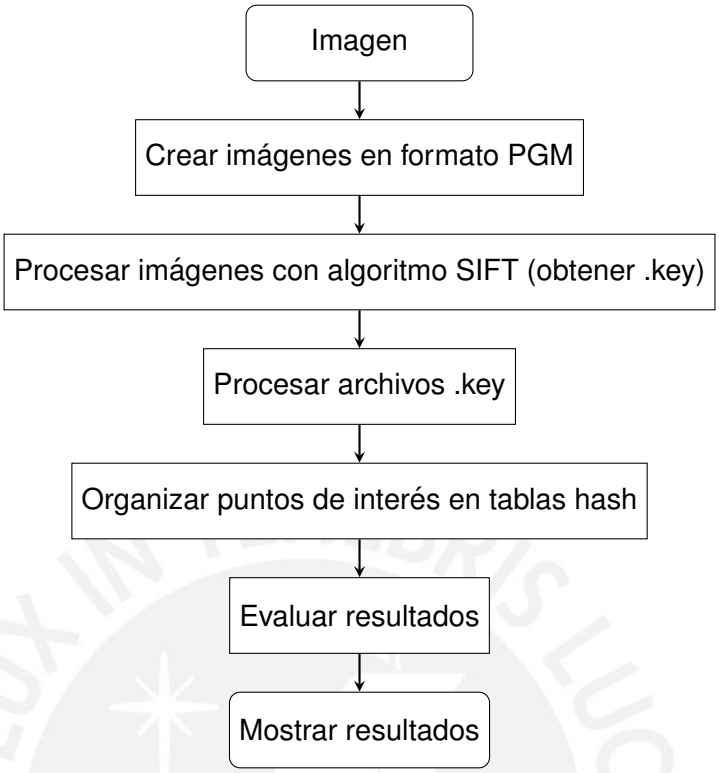
Para una mejor comprensión de lo que se presenta en los capítulos posteriores se describe en rasgos generales el desarrollo de la solución.

Se tiene una cantidad “n” de imágenes que serán evaluadas y comparadas entre si. Por cada una de estas imágenes, se crea una copia en formato PGM (Portable Gray Map), de manera que estas copias puedan ser evaluadas por el algoritmo SIFT. Si la imagen original superan una medida determinada, las dimensiones de la copia serán reducidas hasta las dimensiones máximas que el algoritmo puede procesar. Una vez creadas las copias se ejecuta el algoritmo SIFT, el cual creará un archivo “.key” por cada una de las imágenes. Este tipo de archivo contiene la información de los puntos de interés de cada imagen (mayor detalle sobre este archivo en el capítulo 3 sección 3.1.3).

Luego de que se haya terminado con la creación de los archivos .key, se procederá a leer cada uno de ellos, obteniendo los puntos de interés de cada imagen. En un inicio, se elaboró un módulo que realizaba la comparación de los puntos de interés uno a uno (es decir, se comparaban todos los puntos de interés de una imagen contra los puntos de interés de otra imagen), sin embargo, se observó que este método de comparación era poco eficiente y que demoraba mucho tiempo. Por consiguiente, se hizo uso del algoritmo LSH, el cual permite clasificar cada uno de los puntos de interés de las imágenes dentro de tablas y de acuerdo a su similitud. De esta manera obtendremos una cantidad determinada de tablas, con los puntos de interés distribuidos dentro de ellas de modo que los que sean más similares se encuentren bajo el mismo índice (concepto de “clusterización”[l]). Finalmente, se evalúan los resultados de las tablas, se obtienen los porcentajes de similitud y se presentan en un archivo.

En resumen, el flujo sería de la siguiente forma:

[l] División de la data en grupos(clusters) que son significativos, útiles o ambos [20]



En el siguiente capítulo, se mostrarán los conceptos necesarios para llevar a cabo la parte inicial del proceso.

Capítulo 3

Algoritmo SIFT y comparación uno a uno

En este capítulo se muestra como funciona a detalle el algoritmo SIFT y esta conformado el módulo de comparación uno a uno. Este módulo se encargará de realizar la comparación de 2 imágenes al mismo tiempo.

3.1. Herramienta SIFT

Como se mencionó anteriormente, este algoritmo permite identificar características en una imagen, las cuales podrán ser posteriormente utilizadas para comparar diversas imágenes con un nivel de confianza alto.

3.1.1. Archivos de entrada para el algoritmo

Para poder utilizar el algoritmo SIFT obtenido de la página oficial [21] es imprescindible que todos los archivos se encuentren en el formato PGM (pues todas las imágenes debe de estar en escala de grises). Por tanto, se vio necesario crear un módulo que permita convertir todas las imágenes de entrada a dicho formato. Para lograr esto se utilizó el programa libre ImageMagick, de la cual se agregaron algunas de sus herramientas obtenidas de su sitio web [22]. Éstas permite realizar diversas manipulaciones a archivos de imágenes sin mucha complicación. Las herramientas que se utilizaron en este proyecto fueron las siguientes:

Identify.- Permite obtener gran cantidad de información de la imagen.

Convert.- Permite manipular la imagen de manera que se puede cambiar su tamaño, formato, entre otros.

Ambas herramientas se usaron para poder hacer las conversiones de formato y, de ser necesario, el redimensionamiento de las imágenes que sobrepasen el límite de 1800 pixeles, pues el algoritmo SIFT obtenido no procesa imágenes que tengas dimensiones mayores a estas.

Una vez convertidas todas las imágenes al formato PGM, estas ya podrán ser tomadas como archivos de entrada del algoritmo para poder obtener sus puntos de interés.

3.1.2. Funcionamiento del SIFT

La siguiente explicación fue obtenida del sitio AI Shack [1], el autor de la página explica el funcionamiento del algoritmo según los siguientes pasos:

- **Construcción de espacios de escala.- (*Scale spaces*)** Se crea representaciones internas de la imagen para asegurar la no variación de la escala (escala invariable), lo cual se logra creando ”espacios de escala”.
- **Aproximación al *Laplaciano de Gauss (LoG).-*** El Laplaciano de Gauss es excelente para encontrar puntos de interés (keypoints) pero es muy costoso en términos computacionales, así que se utilizan las representaciones anteriormente creadas para aproximarlo.
- **Encontrando puntos de interés (keypoints).-** Con la mencionada aproximación se intenta encontrar los puntos de interés. Estas son las máximas y mínimas de la Diferencia Gausiana que se calculo en el punto anterior.
- **Deshacerse de malos *keypoints*.-** Los bordes y regiones de bajo contraste son malos puntos de interés, por lo que eliminarlos hará mas robusto al algoritmo.
- **Asignar una orientación a los puntos de interés.-** La orientación es calculada por cada punto de interés. El determinarla permite que el punto de interés se

vuelva invariante a la rotación. Cualquier cálculo posterior se realizará tomando en cuenta dicha orientación.

- **Generar características SIFT.-** Finalmente, se genera una representación más, la cual ayuda a identificar de forma única las características.

3.1.2.1. Construcción de espacios de escala (*Scale spaces*)

Los espacios a escala son imágenes que tratan de simular la naturaleza de las múltiples escalas del mundo real en forma digital. Dicho de otra forma, en el mundo real, los objetos son ”significativos”(dentro del contexto visual) solo a cierta escala y los espacios a escala intentan simular las diferentes escalas en las que se podría ver el objeto de la imagen en cuestión. Por ejemplo, al ver un árbol de lejos se ve este completo, pero si uno se acerca a cierta distancia se pierde la visión del árbol y más bien se ve a mayor detalles las hojas, frutos, etc.

Los espacios de escala se crean tomando la imagen original y generando progresivamente imágenes difuminadas. Esto se realiza con la finalidad de eliminar detalles innecesarios en la imagen (evitando agregar nuevos detalles falsos), lo cual solo es posible con el Difuminando Gausiano (Gaussian Blur)[l].

El algoritmo SIFT utiliza los espacios de escala del siguiente modo: Toma la imagen original y genera progresivamente imágenes difuminadas. Luego, redimensiona la imagen original a la mitad y nuevamente generan imágenes difuminadas a partir de este redimensionado. Este proceso se repite por una determinada cantidad de veces.

Las imágenes del mismo tamaño forman una octava y cada imagen individual representa una escala diferente, la cual va incrementándose (En este caso, el ”incremento de escala” se considera como el aumento de difuminado).

Para ejemplificar el proceso, se toma la imagen de la figura 3.1 y se procede a hacer copias aplicándoles difuminados en diferentes escalas.

Aplicando 5 escalas de difuminado se obtienen las imágenes de la figura 3.2. Estas imágenes obtenidas pasarán a ser parte de la primera octava.

[l]Difuminar es hacer perder la nitidez o claridad a una imagen. Matemáticamente, ”difuminado” es referido como la circunvolución del operador Gausiano y la imagen. El difuminado gausiano tiene una expresión u operador” particular que es aplicado a cada pixel, resultando en la imagen difuminada.

Figura 3.1: Imagen original. [1]

Luego se procede a hacer copias redimensionadas de la imagen original y a dichas copias se procede a realizar el paso anterior (creándoles copias difuminadas). Finalmente se obtendrán varias octavas como las que se muestran en la figura 3.3.

Como se muestra en dicha figura (3.3) se tienen 4 octavas de 5 imágenes cada una. La cantidad de octavas y escalas dependerá del tamaño de la imagen, sin embargo, Lowe sugiere que 4 octavas y 5 niveles de difuminando son ideales para el algoritmo, por lo tanto la herramienta que se utiliza en nuestra solución utiliza estos valores.

3.1.2.2. Aproximación al *Laplaciano de Gauss (LoG)*

Como se mencionó, la operación con el Laplaciano de Gauss (*Laplacian of Gaussian - LoG*) es muy buena para encontrar puntos de interés. El proceso funciona de la siguiente manera:

- Se toma la imagen y se difumina un poco.(La derivada de segundo orden es

Figura 3.2: Imágenes difuminadas. [1]

muy sensible al ruido, por lo que se utiliza el desenfoque para suavizar hacia fuera el ruido y estabiliza la derivada.)

- Sobre el difuminado, se calcula las derivadas de segundo orden.

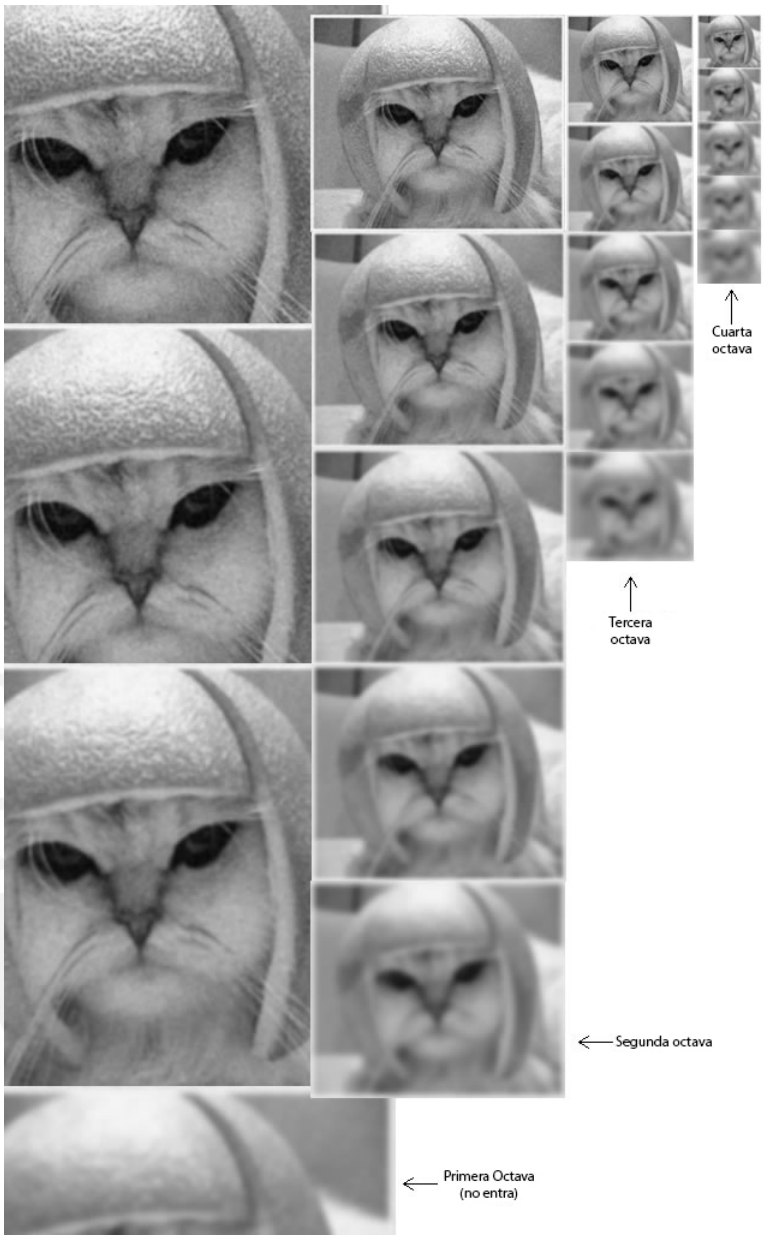


Figura 3.3: Octavas obtenidas. Adaptado de [1]

- Se localizan esquinas y bordes en la imagen (que son buenos para encontrar puntos de interés).

El problema de este método es que el cálculo de las derivadas es computacionalmente intensivo, por lo que el algoritmo SIFT utiliza una forma más sencilla para conseguir un resultado similar.

La estrategia que se utiliza es la siguiente: Para generar rápidamente imágenes de Laplaciano de Gauss (imágenes a las que se les calculó la segunda derivada después de difuminarlas) usamos los espacios de escala generados anteriormente, calculando la diferencia entre 2 escalas consecutivas (Diferencia de Gausianas/*Difference of Gaussians - DoG*). Estas diferencias son aproximaciones casi equivalentes al Laplaciano de Gauss, con las cuales podemos ahorrar trabajo y tiempo del proceso.

Gráficamente este procedimiento se expresa como se ve en la figura 3.4.

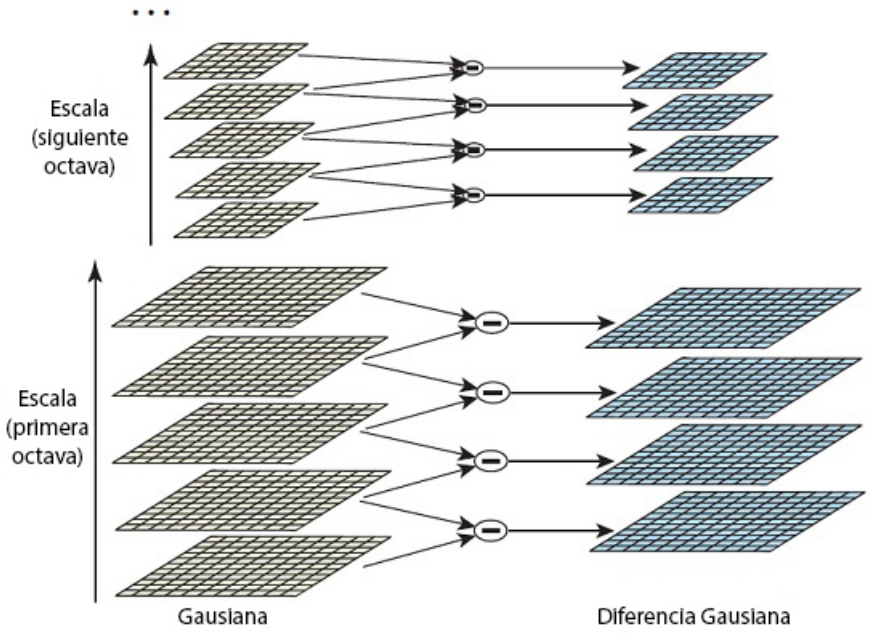


Figura 3.4: Representación gráfica de la estrategia de diferencia de gausianas. Adaptado de [1]

Además, estas diferencias son de escala invariable, por lo que no es necesario aplicarles más procesos para para volverlas invariables.

En nuestro ejemplo, las diferencias se verían como se muestran en las figuras 3.5 y 3.6. Ambas figuras fueron editadas agregando una columna más en la que se muestran las imágenes resultantes de la sustracción con algunos filtros. Estos filtros permiten ver más claramente los puntos más resaltantes de la imagen original, es decir, se muestran las partes de la imagen en donde se podrían encontrar puntos de interés. En nuestro ejemplo, algunas de estas partes son:

- Los bordes de los ojos
- El borde inferior del casco
- Los bordes de la nariz, etc.

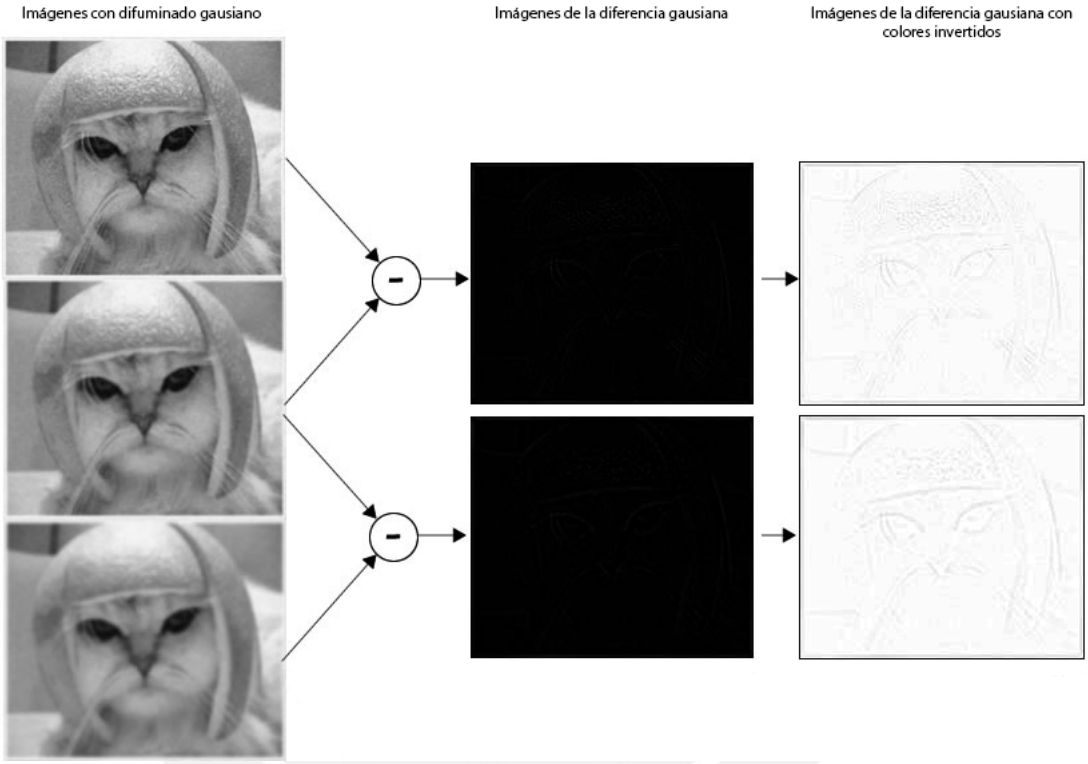


Figura 3.5: Imágenes obtenidas a partir de la aplicación de la diferencia de gausianas Parte 1. Adaptado de [1].

3.1.2.3. Encontrando puntos de interés (*keypoints*)

El buscar los puntos de interés es un proceso que se divide en dos partes:

- Localizar las máximas/mínimas en las imágenes generadas de las diferencias entre 2 escalas (imágenes DoG)
- Encontrar las máxima/mínima de los subpixeles

Localizar máximas/mínimas en imagenes DoG

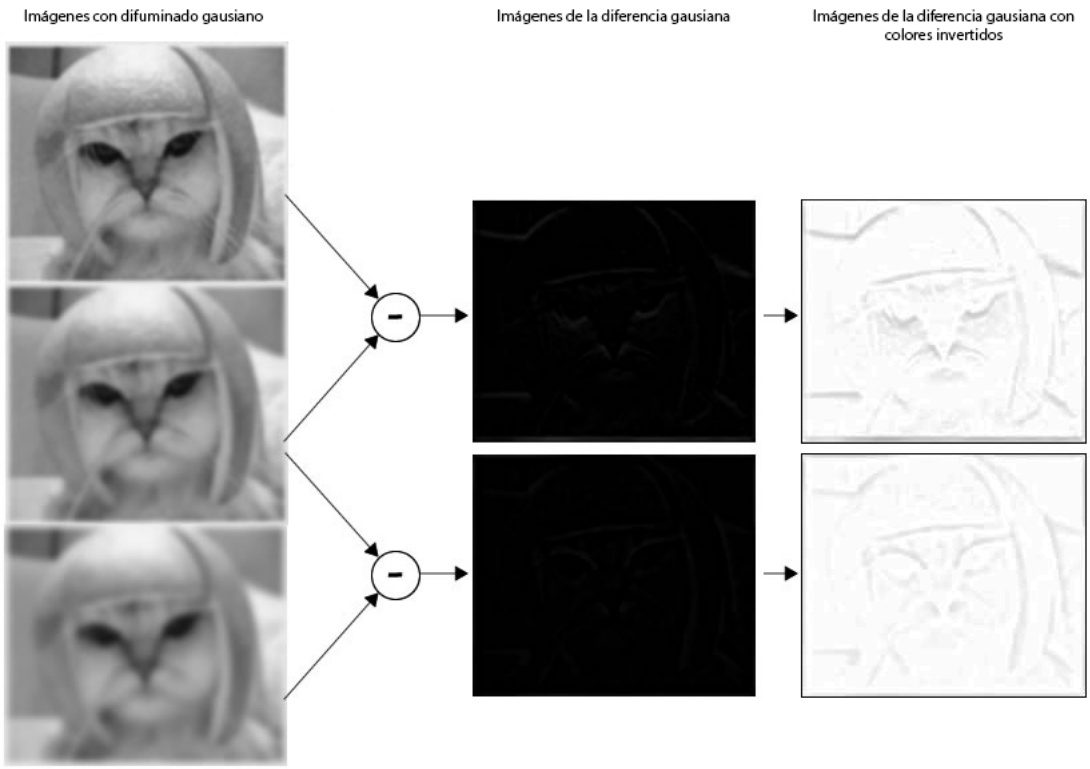


Figura 3.6: Imágenes obtenidas a partir de la aplicación de la diferencia de gausianas Parte 2. Adaptado de [1].

Este paso es sencillo puesto que se itera por cada pixel, verificándolo contra cada uno de sus vecinos. La verificación se debe realizar tomando tanto los pixeles de la imagen actual (imagen DoG que se esta evaluando en ese momento) como los pixeles de las imágenes adyacentes a esta, de la siguiente forma:

La 'x' representa al pixel que se esta evaluando en ese momento y los puntos verdes marcan lo que serían los vecinos de 'x'. Por lo tanto, en este caso, se realizan 26 verificaciones. Si el punto 'x' es el mayor o menor de entre sus vecinos, entonces este punto sera marcado como un "punto de interés".

Nótese que por la naturaleza de la verificación, los puntos de interés no se encontrarán en las imágenes DoG de los extremos, pues no hay suficientes vecinos para realizarla. Por lo tanto no se realiza la verificación en estas imágenes.

Terminado el proceso de verificación, los puntos marcados son las aproximadas máximas y mínimas. Se dice que son "aproximados" porque una máxima/mínima ca-

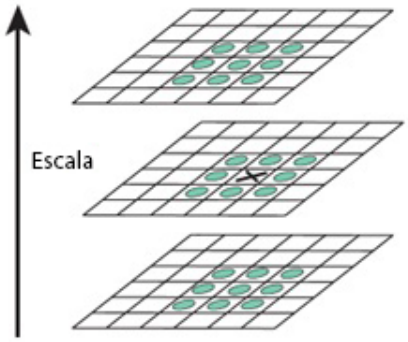


Figura 3.7: Verificación en imágenes DoG. Asumiendo que las 3 imágenes DoG son las primeras, la imagen evaluada en este caso seria la 2da. Adaptado de [1].

si nunca se encuentran exactamente en un pixel sino que se encuentra dentro del pixel. Pero no es posible acceder dentro de un pixel, por lo que se debe obtener la localización del ”subpixel” matemanticamente.

Encontrar las máxima/mínima de los subpixeles

Utilizando la información del pixel, se generan valores de subpixeles. Esto se logra gracias a la expansión Taylor[II] de la imagen al rededor del punto de interés aproximado

3.1.2.4. Deshacerse de malos keypoints

En el paso anterior se producieron una gran cantidad de puntos de interés, algunos de los cuales se encuentran en un borde o no tienen suficiente contraste. En ambos casos, estos puntos no son útiles para la comparación, por lo que son descartados.

Descartando puntos con bajo contraste

Si la magnitud de la intensidad, en el pixel actual de la imagen DoG (que esta siendo verificada para encontrar máxima/mínima) es menor que cierto valor, entonces es descartado.

Descartando bordes

[II]La expansión Taylor

Se calculan 2 gradientes en el punto de interés, una perpendicular a la otra. En base a la imagen al rededor del punto de interés, existen 3 posibilidades. La imagen al rededor puede ser:

Una región plana.- En cuyo caso ambas gradientes son pequeñas.

Un borde.- En este caso una gradiente sera pequeña y la otra grande.

Una esquina.- Aquí ambas gradientes serán grandes.

Las esquinas son buenos puntos de interés, por lo que si ambas gradientes son suficientemente grandes el punto de interés se conserva, en otro caso se descarta.

Matematicamente esto se logra gracias a la Matriz Hessiana[III]. Con esta matriz se puede verificar fácilmente si un punto es un borde o no.

3.1.2.5. Asignar una orientación a los puntos de interés

En este paso, se procede a asignarle una orientación al punto de interés para que no varíe ante una rotación (rotación invariable).

La idea general es recolectar las orientaciones y magnitudes de las gradientes al rededor de cada punto de interés. Luego se deducirá la orientación (u orientaciones) mas destacada en esa región y esta se asignará al punto de interés. Cualquier cálculo posterior se realizará considerando esta orientación, lo cual asegura la orientación invariable.

Para poder hacer esta recolección, primero se identifica el punto de interés sobre el cual se trabajará y se define la región dentro de la cual se realizará la recolección. Esto se muestra en las imágenes de la figura 3.8.

Nota: El tamaño de la ”región de recolección” depende de la escala. A una mayor escala le corresponderá una región mayor.

Luego de definir la región, se calculan las magnitudes y orientaciones dentro de esta, como se muestra en la figura 3.9.

[III]La Matriz Hessiana o Hessiano es una matriz cuadrada de las segundas derivadas parciales de una función. [23]

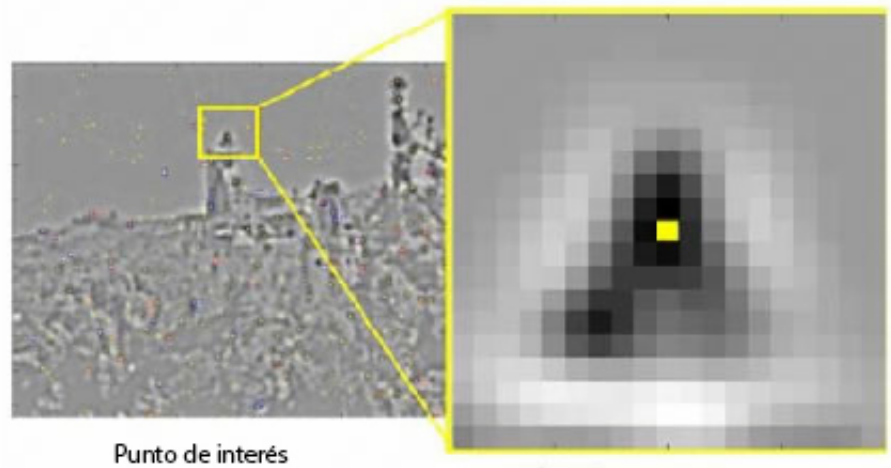


Figura 3.8: Punto de interés actual. Adaptado de [1].

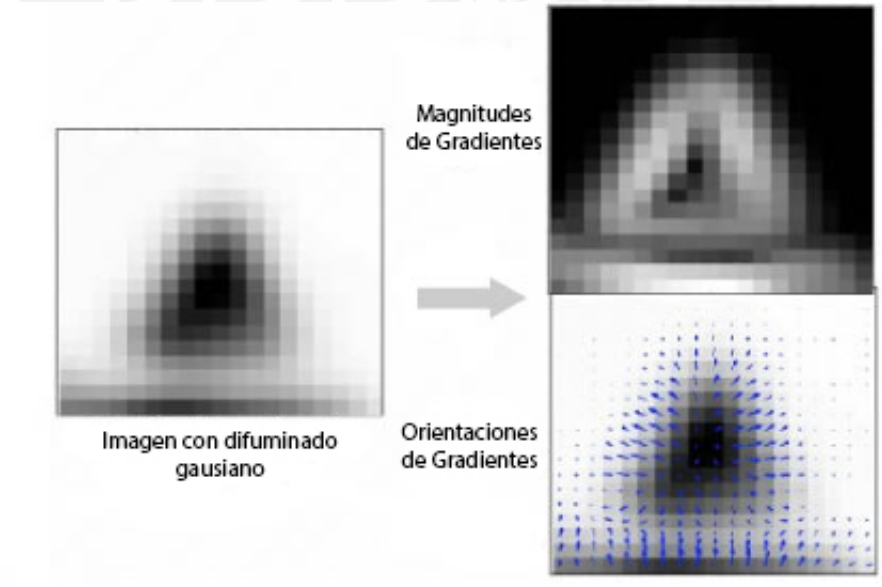


Figura 3.9: Magnitud y orientación de gradiente. Adaptado de [1].

Estas magnitudes y orientaciones se calculan utilizando la fórmula de la figura 3.10.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Figura 3.10: Fórmula para cálculo de magnitud y orientaciones. [1]

Las magnitudes y orientaciones son calculadas para todos los pixeles al rededor del punto de interés. En base a esto se crea un histograma.

En este histograma, los 360 grados de orientación se reparten en contenedores de 10 grados cada uno, resultando en un total de 36 contenedores distribuidos de la siguiente manera:

contenedor 1: 0° - 9°

contenedor 2: 10° - 19°

contenedor 3: 20° - 29°

contenedor 4: 30° - 39°

⋮

contenedor 36: 350° - 359°

De esta manera, la orientación y magnitud de cada pixel de la región es almacenada dentro de estos contenedores de la siguiente forma: dependiendo de la orientación es que se asigna un determinado contenedor. Por ejemplo, si la orientación de el pixel actual es de 18.759 grados, entonces esta debe ir en el contenedor de 10 a 19 grados. La ”cantidad” que es agregada a este contenedor será proporcional a la magnitud de dicho pixel.

Una ves se ha realizado este proceso para todos los pixeles de la región, el histograma mostrará un pico en alguno de los rangos.

En la imagen 3.11, se puede observar que el pico en el histograma esta en el rango de 20 a 29 grados. Por lo tanto el punto de interés tendrá asignada una orientación de 3 (pues este rango es del contenedor 3).

Además, cualquier pico que supere el 80 % será convertido en un nuevo puntos de interés. Es decir, se creará un nuevo punto de interés, que tendrá la misma ubicación y escala que el original, pero su orientación será la del otro pico.

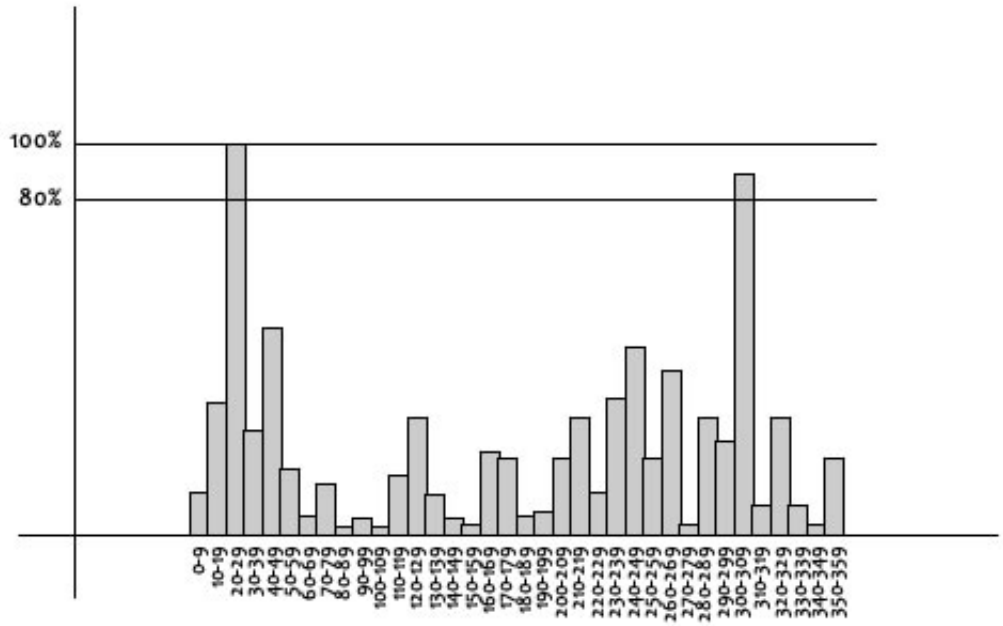


Figura 3.11: Histograma de orientaciones. En este caso se producirán 2 puntos de interés pues hay un punto que sobrepasa el 80 %. [1]

3.1.2.6. Generar características SIFT

En este último paso, lo que se hace es crear una huella digital (fingerprint) para cada punto de interés, para así poder identificarlo.

Para poder hacer esto se crea un marco de 16x16 pixeles al rededor del punto, el cual a su ves se divide en 16 pequeños marcos de 4x4. Gráficamente es como se muestra en la figura 3.12.

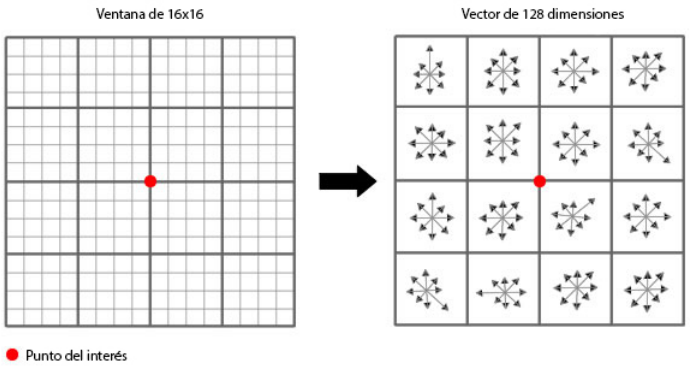


Figura 3.12: Ventana de 16x16 pixeles divididas en 16 ventanas de 4x4. Adaptado de [1]

Dentro de cada marco de 4x4 se calculan las magnitudes y orientaciones de los pixeles, las cuales se pondrán en un histograma de 8 contenedores, como se muestra en la figura 3.13.

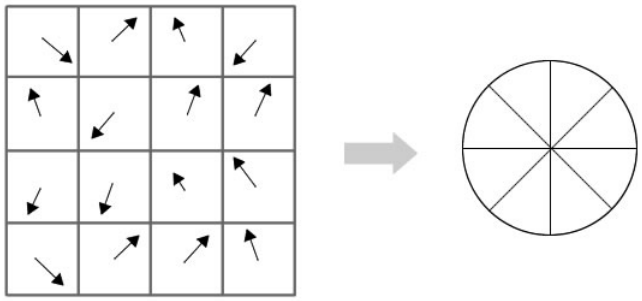


Figura 3.13: Ventana de 4x4 pixeles y conjunto de 8 contenedores. [1]

De la misma forma que en el paso anterior, las orientaciones se ponen en los contenedores que corresponden a sus grados. Sin embargo, la cantidad añadida a cada contenedor no dependerá solo de la magnitud del gradiente, sino que también de la distancia que hay al punto de interés. Por tanto, las gradiente que estén más alejadas del punto de interés añadirán un valor menor al contenedor. Para poder calcular la cantidad a agregar en los contenedores, se utiliza una ”función de ponderación gaussiana”, la cual genera un gradiente que se multiplica con la magnitud de las orienta-

ciones y con lo cual se obtiene una ponderación. Graficamente se expresa como en la figura 3.14.

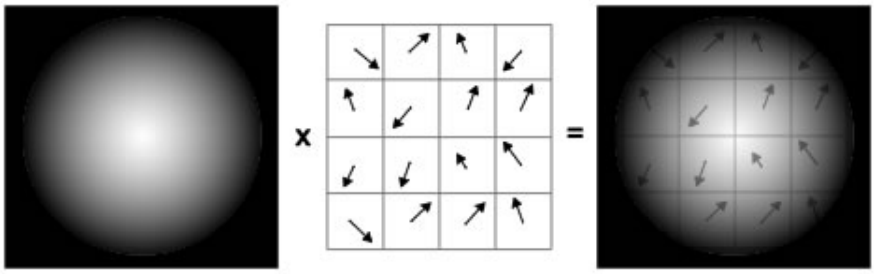


Figura 3.14: Ponderación Gaussiana. [1]

Se realiza este proceso para los 16 pixeles (dentro del marco de 4x4) obteniendo un histograma con 8 contenedores. Esto se repite para los 15 marcos restantes de 4x4, terminando con una cantidad de 16 histogramas, es decir, 128 valores (16 ventanas x 8 contenedores). Estos valores luego se normalizan y se reúnen conformando un vector denominado el ”vector característico”. El punto de interés será identificado gracias a este vector.

El ”vector característico” permitirá identificar semejanzas entre distintas imágenes permitiendo no solo afirmar si estas son idénticas o no, puesto que solo se compararán dichos vectores y no la imagen completa.

Nota: En las imágenes se puede apreciar que el punto de interés no se encuentra dentro de un pixel en específico, sino que esta ”entre pixeles”. Por tanto, el marco de 16x16 toma orientaciones y magnitudes que estan entre los pixeles. Es por esto que se debe interpolar la imagen para poder obtener estos valores entre pixeles.

Resolviendo problemas

El vector característico presenta algunas dependencias, las cuales deben de ser substraídas para poder finalizar el proceso.

Dependencia de rotación.- El vector utiliza la orientación de las gradientes, por lo que si se rota la imagen, todos los valores cambiaran. Para evitar esto, lo que se debe de hacer es substraer la rotación del punto de interés a cada una de las orientaciones, de manera que la orientación de cada gradiente es relativa a la orientación del punto

de interés.

Dependencia de iluminación.- Si se retiran los números grandes, se puede lograr la independencia de iluminación, de manera que cualquier valor que sea mayor a 0.2 (valor fijado por el algoritmo) será cambiado a 0.2. Luego se vuelve a normalizar el vector y este ya será independiente de la iluminación.

3.1.3. Archivo .KEY

Este tipo de archivo es el que se obtiene como resultado luego de utilizar el algoritmo SIFT sobre una imagen. Este archivo almacena los puntos claves de la imagen sobre la cual se corrió el algoritmo además de otros datos más de dicha imagen.

Un archivo .key tiene la siguiente estructura:

Primera línea: 2 números, el primero es la cantidad de keypoints que hay en el archivo y el segundo es la cantidad de descriptores por cada keypoint (por lo general son 128)

Segunda línea: 4 números con información del primer keypoint; El primero es la posición en coordenadas x, el segundo la posición en coordenadas y, el tercero es la escala y el 4to la orientación.

Lineas 3 a 9 : Los descriptores del keypoint de la línea anterior, con 20 descriptores por línea (excepto en la última línea que hay menos descriptores)

Lineas 10 en adelante: Repiten el formato desde la 2da línea hasta terminarse todos los keypoints.

3.1.4. Keypoints y descriptores

Los Keypoints, como ya se mencionó, son los puntos de interés que se obtienen de una imagen. Estos se comparan contra los puntos de interés de otras imágenes para determinar si estas son similares o no.

Los Keypoints en este caso están conformados por los siguientes elementos:

- Posición en el eje X del subpixel
- Posición en el eje Y del subpixel
- Escala (en radianes de -PI a PI)

- Orientación (en radianes de -PI a PI)
- Vector característico (con 128 descriptores)

Los descriptores son cada uno de los números que están dentro del ”vector característico” del punto de interés. Con estos es con los que se trabaja para poder determinar si 2 imágenes son similares o no.

3.2. Aplicación del SIFT

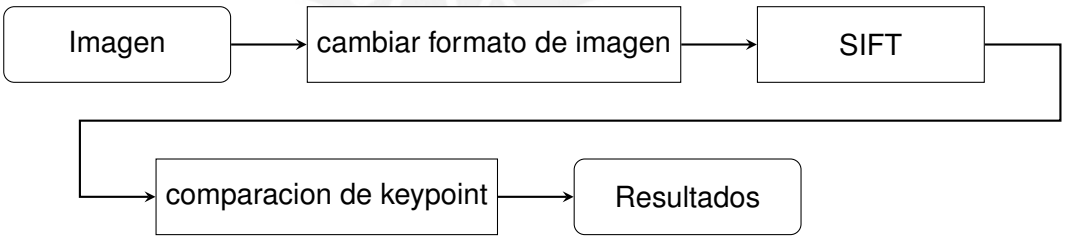
Dentro de nuestra solución el algoritmo SIFT es utilizado de la siguiente forma:

Primero se crea una lista con los nombres de las imágenes que se encuentran dentro de la carpeta a evaluar. A continuación, con dicha lista, se procede a convertir las imágenes a formato PGM (puesto que este es el único formato que recibe el algoritmo SIFT de Lowe) tomando siempre en cuenta las restricciones especificadas (tipos de archivo, tamaño, etc). Este proceso lo logramos utilizando las herramientas ”identify” y ”convert” de la librería de ImageMagick.

Una ves convertidas las imágenes, se procede al uso de SIFT para obtener los puntos clave de las imágenes. Estos puntos son almacenados por la misma herramienta en archivos con extensión .key.

Ya obtenidos los archivos .key se procede a leerlos para poder realizar la comparación entre imágenes.

El flujo finalmente quedará de la siguiente forma:



Capítulo 4

Comparación múltiple

En este capítulo se explicarán los cambios que se realizan al módulo obtenido en el capítulo anterior, para lograr la comparación de varias imágenes.

4.1. Selección de herramienta

Lo que se implemento en un inicio fue un ”modulo de comparación” el cual compara los archivos en modo ’todos contra todos’, es decir, toma una imagen y la compara contra todas las restantes, luego toma la siguiente imagen y la compara contra las otras (sin incluir a las que ya se compararon) y sigue así hasta terminar con todas las imágenes (ver Imagen 4.1).

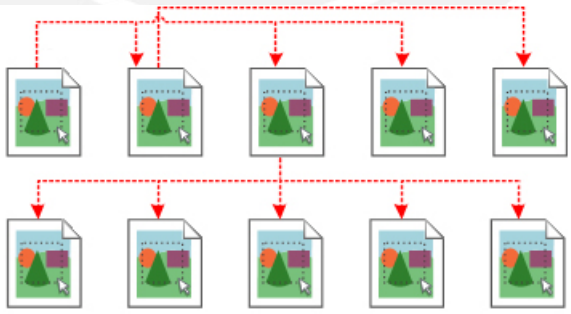


Figura 4.1: Representación gráfica de proceso de comparación [Elaboración propia]

Este método es algo ineficiente, puesto que el tiempo de comparación es alto al tener que evaluar cada uno de los puntos de interés con los otros, por tanto se

evaluaron otras opciones para realizar este trabajo.

Para mejorar los tiempos se consideraron 2 opciones:

Árboles de n-nodos.- Plantea crear una estructura tipo árbol, en la cual se alojarían los descriptores de las imágenes de manera que los que se parezcan se encuentren bajo un mismo nodo. Se descartó utilizar esta opción debido a que no ofrecía una mejora razonable en los tiempos.

Herramienta LSH.- (*Locality Sensitive Hashing*) Algoritmo que se suele utilizar para la resolución de la búsqueda de Vecino más Cercano, teniendo la ventaja de ser eficiente aun si la búsqueda se realiza sobre data muy extensa. [7]. Pertenece a una clase de algoritmos denominada “Randomized Algorithms”(Algoritmos aleatorizados) que no garantizan una respuesta exacta, pero proveen una garantía de alta probabilidad de que se obtendrá una respuesta correcta o cercana a la correcta [8].

En rasgos generales, lo que el algoritmo LSH hace es encriptar varias veces los items ingresados como entradas para luego realizar un proceso de “clusterizacion” organizando los items de manera que los más parecidos se encuentren en un mismo índice.

Por ejemplo, en nuestro caso, podría expresar de la siguiente forma:

Luego de aplicar el algoritmo SIFT, se tiene los puntos de interés de 4 imágenes: Imagen1, imagen2, imagen3 e imagen4 con 17, 15, 19 y 9 puntos respectivamente. Expresándolo gráficamente se vería como en la imagen de la figura 4.2. Cada punto de interés está identificado por las siglas “IxKy” donde “x” es el numero de la imagen e “y” es el número del keypoint.

Luego, al pasar los puntos de interés por el algoritmo LSH, estos serán organizados de manera que los puntos de interés similares se encuentren bajo un mismo índice. Gráficamente, la distribución se vería como en la figura 4.3. Como se puede apreciar, la tabla indica que los puntos de interés 1 y 2 de la Imagen 1, se parecen al punto 3 de la Imagen 2, a los puntos 4, 6 y 8 de la la Imagen 3 y al punto 5 de la Imagen 4.

Finalmente se decidió utilizar la segunda opción (LSH) puesto que al analizar la primera opción se observó que esta podría incurrir en altos consumos de memoria (en especial cuando se tuvieran muchos nodos). Además, en la documentación revisada

[14] [17] se recomienda utilizar este algoritmo.

4.2. Manejo de resultados

Una vez decidida la herramienta, se realizaron los cambios necesarios para poder obtener los resultados en un formato legible. En consecuencia, la herramienta devuelve un archivo en formato csv, el cual contiene una matriz de doble entrada en la cual se muestra el porcentaje de parecido entre todas las imágenes. Este porcentaje se obtiene de la siguiente forma:

Por cada índice de tabla hash creada por el LSH, se guarda una lista que contiene el nombre de una imagen y la cantidad de keypoints que tiene dicha imagen dentro de ese índice. Una vez llena la tabla hash, se hace un recorrido tomando una imagen de base y obteniendo la cantidad de puntos de interés que coincidan con los de las otras imágenes. Por ejemplo, tomamos de base la imagen “A” y la evaluamos contra la imagen “C”, lo que se hará es guardar y sumar los puntos de interés de los índices en los que ambas imágenes (“A” y “C”) tengan al menos un punto de interés. Esto nos dará la cantidad de puntos de interés de la imagen “A” que coinciden con los de la imagen “C”. Finalmente, obtenemos un porcentaje sumando ambas cantidades y dividiéndolas entre la suma de la cantidad total de puntos de interés de cada imagen. Una vez hecho esto por cada tabla hash, se toma la mediana de todos los valores y es ésta la que se toma como porcentaje de similitud.

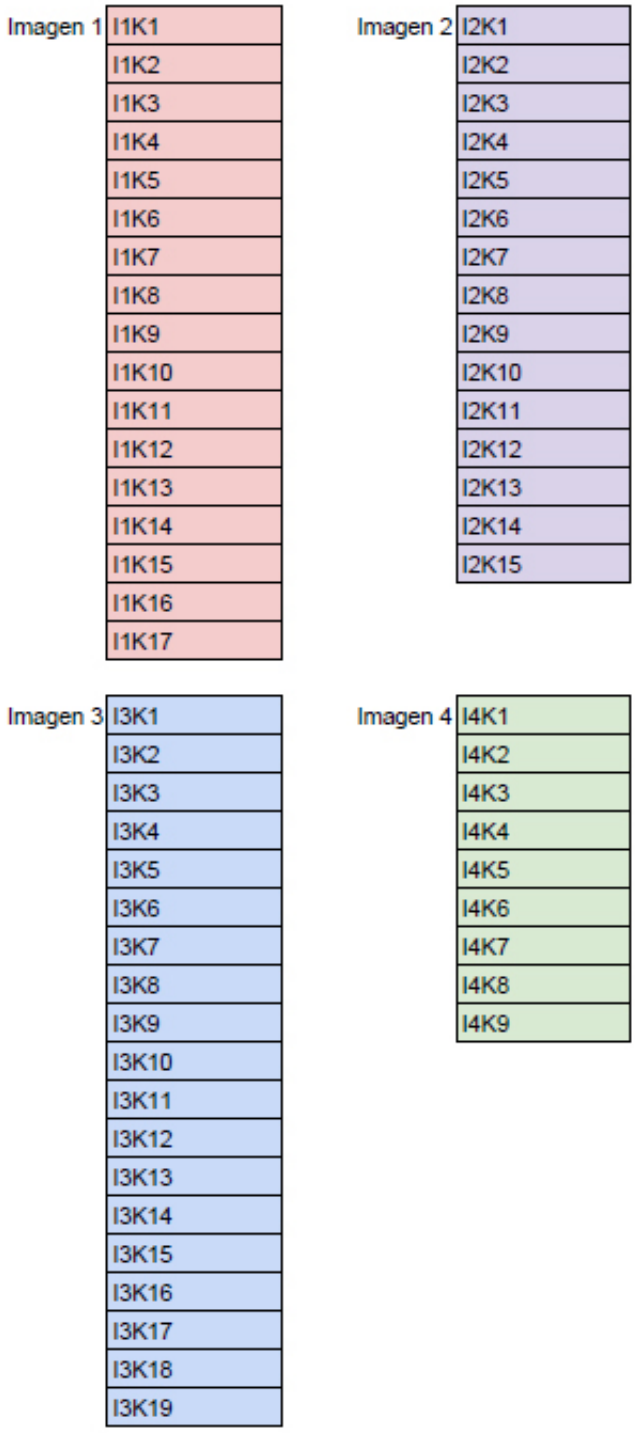


Figura 4.2: Ejemplo de puntos de interés de 4 imágenes [Elaboración propia]

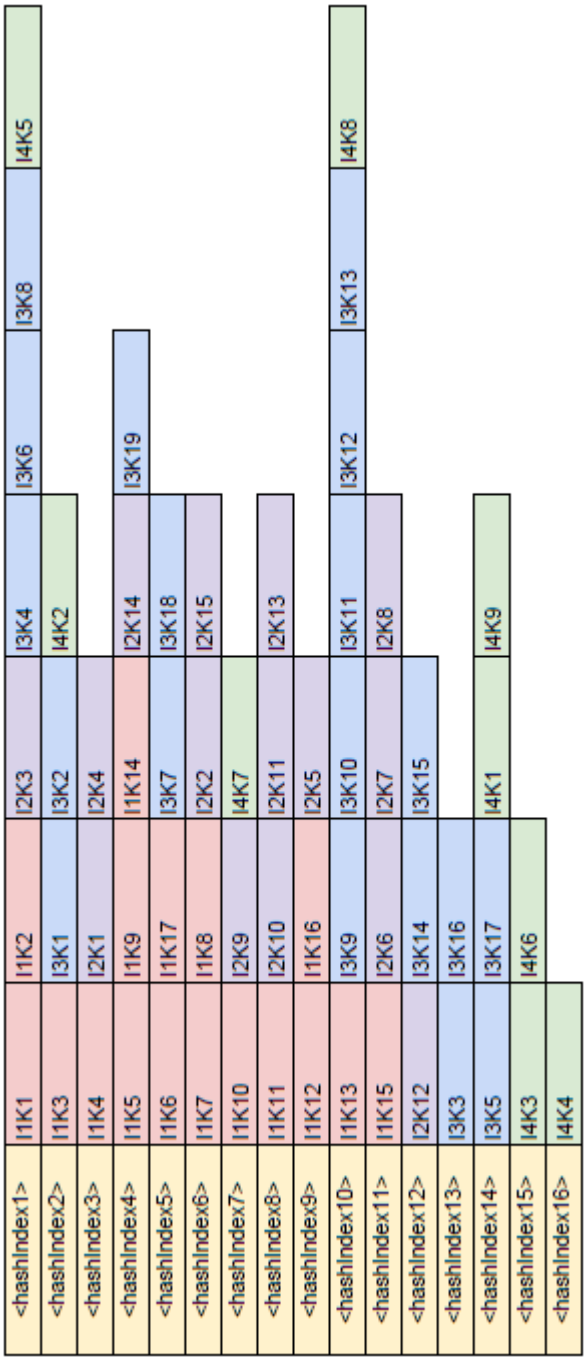


Figura 4.3: Ejemplo de distribución de los puntos de interés de la figura 4.2 dentro de una tabla hash [Elaboración propia]

Capítulo 5

Pruebas y validación

A continuación se describen las pruebas realizadas a la herramienta implementada para poder validar su funcionamiento.

5.1. Metodología

Para poder sustentar la precisión de la solución elaborada, se utilizó la curva Precision/Recall, pues este método permite definir que tan precisa es la obtención de datos por parte de la herramienta.

5.1.1. Curva Precision/Recall

A continuación se indican los conceptos de Precisión y Recall dentro del contexto de reconocimiento de patrones.

Precisión: Denota la proporción de datos que han sido obtenidos y clasificados como positivos y que realmente son “Positivos”. [24]

Recall: Es la proporción de “Positivos” que han sido obtenidos y clasificados como positivos. [24]

Estos conceptos pueden entenderse más claramente apreciando la figura 5.1.

Los “Positivos” a los que hacen referencia ambas definiciones, se refieren a los elementos que son verdaderamente relevantes dentro de un grupo de elementos.

Por ejemplo, Se tiene un conjunto de 50 elementos, dentro de los cuales los que son relevantes son 15. Se hace una búsqueda y se obtienen 30 elementos, dentro de

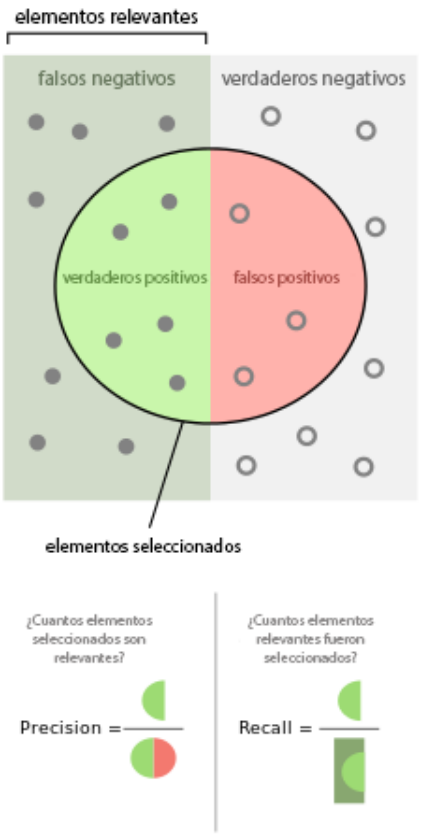


Figura 5.1: Representación gráfica de de Precision/Recall. La zona de la izquierda es la zona que contiene los elementos “Positivos”(relevant elements) de entre todos los datos y la zona de dentro del óvalo es la que contiene los datos obtenidos(selected elements). Adaptado de [25]

los cuales solo 10 son relevantes. En tal caso tendríamos:

Elementos “Positivos” : 15

Elementos “Positivos” obtenidos : 10

Elementos Obtenidos : 30

Por lo tanto, en este ejemplo:

Precisión = 10/30

Recall = 10/15

Normalmente, Precisión y Recall están inversamente relacionados, es decir, cuando uno se incrementa el otro decae. Por lo general se busca que ambos valores estén balanceados para que se puedan obtener datos de manera eficiente.

5.1.2. Dominio

Para poder obtener los datos necesarios para la elaboración de la curva, primero se utilizó un dominio que se compuso con imágenes recolectadas de [26] y de [27]. Sin embargo, al ser las imágenes de poca resolución (a pesar de ser de dimensiones no muy pequeñas) no permitían obtener muchos puntos de interés, y no proporcionaban la cantidad de información necesaria para que la herramienta pueda evaluarlas adecuadamente, por lo que los resultados obtenidos en base a este dominio fueron descartados. Posteriormente se utilizó el dominio obtenido de [28], de donde se obtuvieron imágenes con una mejor resolución que en el caso previo. Además, el sitio donde se encuentra dicho dominio ofrece cierta información del parecido de las imágenes, lo cual no se encontraba en otros datasets considerados, por lo que se pudo tomar dicha información como base para poder realizar la validación.

5.1.3. Pruebas

Luego de tener el dominio establecido, se realizaron pruebas con diferentes cantidades de imágenes, para así poder observar la variación de la precisión y el recall.

Se realizaron 5 pruebas con cantidades de 12, 24, 35, 46 y 52 imágenes. Se tomó como muestra máxima la cantidad de 52 imágenes debido a que la información de parecido que ofrece el sitio de donde se tomó el dominio no era suficiente (no había información de las relaciones entre todas las imágenes), y se quiso evitar ingresar mucha data no validada. El grupo de imágenes y los detalles de las pruebas se puede visualizar en los anexos.

5.1.4. Resultados

Luego de las pruebas se obtuvieron los resultados mostrados en la tabla 5.1.

Donde:

Clase 1: Corresponde a la clasificación de imágenes que son muy parecidas (65 % a más).

Clase 2: Corresponde a la clasificación de imágenes que son algo parecidas (entre 30 % y 65 %).

		CLASE 1	CLASE 2	CLASE 3
12	TRUE POSITIVES	3	0	2
	FALSE POSITIVES	46	15	0
	FALSE NEGATIVES	0	2	59
24	TRUE POSITIVES	0	8	22
	FALSE POSITIVES	19	227	0
	FALSE NEGATIVES	10	2	234
35	TRUE POSITIVES	0	21	111
	FALSE POSITIVES	0	463	0
	FALSE NEGATIVES	15	0	448
46	TRUE POSITIVES	0	27	44
	FALSE POSITIVES	122	837	0
	FALSE NEGATIVES	26	21	917
52	TRUE POSITIVES	1	51	47
	FALSE POSITIVES	101	1126	0
	FALSE NEGATIVES	34	16	1177

Cuadro 5.1: Tabla de resultados de evaluación de imágenes

Clase 3: Corresponde a la clasificación de imágenes que son poco parecidas (menor a 30 %).

En base a ellos se armó la tabla 5.2.

		12	24	35	46	52
CLASE 1	PRECISION	0.06122449	0	0	0	0.009803922
	RECALL	1	0	0	0	0.028571429
CLASE 2	PRECISION	0	0.034042553	0.04338843	0.03125	0.043330501
	RECALL	0	0.8	1	0.5625	0.76119403
CLASE 3	PRECISION	1	1	1	1	1
	RECALL	0.032786885	0.0859375	0.198568873	0.04578564	0.038398693

Cuadro 5.2: Tabla de resultados para Precision/Recall

En base a estos datos, se obtuvo el gráfico de la figura 5.2.

Se puede observar que la precisión para las clases 1 y 2 es muy baja, mientras que el recall es bastante alto. Sin embargo, para la clase 3 la precisión es bastante alta siendo el recall algo bajo.

Estos resultados pueden deberse a los siguientes:

El tipo de imágenes utilizadas.- Las imágenes que se utilizaron son fotos de diversas edificaciones, por lo que es muy posible que varios de los puntos de interés de 2 imágenes sean parecidos a pesar de que dichas imágenes sean de distintas estructuras.

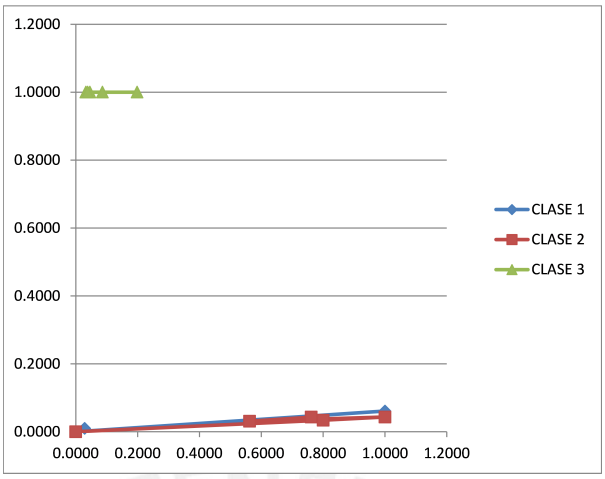


Figura 5.2: Curva Precision/Recall elaborada en base a los datos obtenidos

El tipo de evaluación que se utilizó.- Es posible que la evaluación que se utiliza para obtener los porcentajes no sea muy precisa y deba ser mejorada para poder abarcar la variedad casos que se presentan.

Capítulo 6

Conclusiones y Trabajos Futuros

A continuación expondremos las conclusiones a las que se llegaron tras la realización del proyecto y algunas recomendaciones sobre la solución planteada.

En base a lo investigado durante la elaboración de la tesis se ha podido observar que el tema tratado (procesamiento de archivos de imagen) tiene una gran amplitud.

Si bien existen aplicativos que tratan de dar solución al problema de duplicación de archivos, estos son de licencia propietario. Y a pesar de eso no cumplen del todo las expectativas del usuario, ya que tiene deficiencias en la interfaz gráfica, exactitud de los resultados o tiempos de demora.

6.1. Conclusiones

- Se logró modelar una estructura de datos que permite que las imágenes de los formatos jpg, png, gif y bmp sean evaluadas por el algoritmo SIFT. Se desarrollo un módulo que permitiera la conversión de las imágenes al formato utilizado por el SIFT y además que les redujera el tamaño de ser necesario, puesto que el algoritmo SIFT no podía trabajar con imágenes muy grandes. Con esto se logró que cualquier imagen que sea de los formatos jpg, png, gif y bmp pueda ser comparada.

- Se implementó el módulo necesario para poder obtener los puntos de interés con el algoritmo SIFT, de manera que estos puedan ser leídos, pudiendo trabajar con ellos y así utilizarlos para poder realizar las comparaciones entre imágenes.

- Se adaptó el algoritmo LSH para poder organizar los puntos de interés obtenidos con el algoritmo SIFT y así sistematizar la operación de comparación, volviéndola más rápida y eficaz. Además, se hicieron las adaptaciones necesarias para que el algoritmo devuelva los resultados en un formato legible para las personas.
- Se unieron los módulos implementados para lograr que ambos algoritmos (SIFT y LSH) trabajen conjuntamente. De esta manera se obtuvo una herramienta que permite obtener un porcentaje de similitud entre imágenes que se encuentren en un directorio especificado.
- Se verificó el funcionamiento de la herramienta haciendo uso de la curva Precision/Recall, observando que los resultados, a pesar de no alcanzar la precisión deseada, si permiten establecer una distinción entre un grupo de imágenes.

6.2. Trabajos futuros

A continuación, algunas recomendaciones para trabajos futuros que se relacionen al tema.

- En el caso de manejo de imágenes, es recomendable establecer los formatos con los cuales se piensa trabajar desde un inicio, puesto que en la actualidad se tienen una gran variedad de formatos y no todos pueden manipularse del mismo modo.
- Para trabajar con código de otros autores (eg. algoritmo SIFT, herramienta ImageMagick, etc), es bueno contar con toda la documentación posible, y evaluar la codificación contra dicha documentación. Además, es recomendable tener el contacto de él o los autores, por si se presenta alguna duda.

Finalmente, creemos que sería posible hacer más precisa la solución presentada, agregando módulos que permitan evaluar las imágenes por su distribución de colores o que permitan un mejor desempeño frente a casos como los de desenfoque o falta de contraste. Esto no se pudo realizar en la presente solución pues salia del alcance propuesto.

Bibliografía

[1] U. Sinha. (2016, Febrero). [Online]. Available: http://www.aishack.in/

[2] RAE. (2016, Febrero) Pixel. [Online]. Available: http://lema.rae.es/dpd/?key=pixel

[3] J. Digital. (2016, Febrero) A versatile demosaicing algorithm for performing image zooming. [Online]. Available: http://www.jaguar.edu.co/z_aprendizaje/tutoriales/imagenDigital/

[4] J. S. Simonoff. Smoothing methods in statistics.

[5] J. W. S. Ltd. (2016, Febrero) Hypermedia image processing reference. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

[6] R. Hess, "An open-source siftlibrary," in *Proceedings of the International Conference on Multimedia*, ser. MM '10. New York, NY, USA: ACM, 2010, pp. 1493–1496. [Online]. Available: http://doi.acm.org/10.1145/1873951.1874256

[7] A. Andoni and P. Indyk. (2010, June) E2 lsh 0.1. [Online]. Available: http://www.mit.edu/~andoni/LSH/manual.pdf

[8] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 128–131, March 2008. [Online]. Available: http://www.slaney.org/malcolm/yahoo/Slaney2008-LSHTutorial.pdf

[9] L. Chen, C. Chen, and M. Lu, "A multiple-kernel fuzzy c-means algorithm for image segmentation," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 5, pp. 1263–1274, 2011.

[10] R. Chan and J. Ma, “A multiplicative iterative algorithm for box-constrained penalized likelihood image restoration,” *Image Processing, IEEE Transactions on*, vol. 21, no. 7, pp. 3168–3181, 2012.

[11] N. Agarwal, A. Kumar, J. Bhadviya, and A. Tiwari, “A switching based adaptive image interpolation algorithm,” in *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, 2012, pp. 981–984.

[12] C.-C. Lai and Y.-C. Chen, “A user-oriented image retrieval system based on interactive genetic algorithm,” *Instrumentation and Measurement, IEEE Transactions on*, vol. 60, no. 10, pp. 3318–3325, 2011.

[13] A. Hore, D. Ziou, and M. Auclair-Fortier, “A versatile demosaicing algorithm for performing image zooming,” in *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*, 2012, pp. 402–409.

[14] Y. Ke, R. Sukthankar, and L. Huston, “An efficient parts-based near-duplicate and sub-image retrieval system,” in *Proceedings of the 12th annual ACM international conference on Multimedia*, ser. MULTIMEDIA ’04. New York, NY, USA: ACM, 2004, pp. 869–876. [Online]. Available: http://doi.acm.org/10.1145/1027527.1027729

[15] R. A. Carrión Castagnola. (2011, Junio) Desarrollo de un algoritmo de clasificación de la huella dactilar para la policía nacional del perú. [Online]. Available: http://tesis.pucp.edu.pe/repositorio/handle/123456789/535

[16] S. Se, D. Lowe, and J. Little, “Vision-based mobile robot localization and mapping using scale-invariant features,” in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, 2001, pp. 2051–2058 vol.2.

[17] Y. Jing and S. Baluja, “Visualrank: Applying pagerank to large-scale image search,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1877–1890, Nov 2008.

[18] M. Brown and D. G. Lowe, "Recognising panoramas," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct 2003, pp. 1218–1225 vol.2.

[19] H. Koga, T. Ishibashi, and T. Watanabe, *Discovery Science: 7th International Conference, DS 2004, Padova, Italy, October 2-5, 2004. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, ch. Fast Hierarchical Clustering Algorithm Using Locality-Sensitive Hashing, pp. 114–128. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30214-8_9

[20] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. [Online]. Available: https://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf

[21] (2016, Febrero) Demo software: Sift keypoint detector. [Online]. Available: http://www.cs.ubc.ca/~lowe/keypoints/

[22] (2013, Septiembre) Binary release. [Online]. Available: http://www.imagemagick.org/script/binary-releases.php

[23] (2016, Febrero) World web math. [Online]. Available: http://web.mit.edu/wwmath/vectorc/minmax/hessian.html

[24] D. M. W. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation." [Online]. Available: https://www.researchgate.net/publication/228529307_Evaluation_From_Precision_Recall_and_F-Factor_to_ROC_Informedness_Markedness_Correlation

[25] Walber. Precision and recall. [Online]. Available: https://commons.wikimedia.org/wiki/File%3APrecisionrecall.svg

[26] R. F. L. Fei-Fei and P. Perona. (2004) Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. ieee. cvpr 2004, workshop on generative-model based vision. [Online]. Available: http://www.vision.caltech.edu/Image_Datasets/Caltech101/

[27] M.-E. Nilsback and A. Zisserman. 17 category flower dataset. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html

[28] R. A. James Philbin and A. Zisserman. The oxford buildings dataset. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/index.html