UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SISTEMA DE BÚSQUEDA MEDIANTE EJEMPLO PARA DETECCIÓN DE PLAGIO EN IMÁGENES PARA LA PLATAFORMA DOCODE

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

SIMÓN PEDRO SEPÚLVEDA RUDOLPH

PROFESOR GUÍA:
JUAN VELÁSQUEZ SILVA

MIEMBROS DE LA COMISIÓN:
GASPAR PIZARRO VENEGAS
ANDRES CABA RUTTE

SANTIAGO DE CHILE
2018

RESUMEN DE LA MEMORIA PARA OPTAR AL
TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: SIMON PEDRO SEPULVEDA RUDOLPH
FECHA: SEPTIEMBRE 2018
PROFESOR GUÍA: JUAN VELASQUEZ SILVA

SISTEMA DE BÚSQUEDA MEDIANTE EJEMPLO PARA DETECCIÓN DE PLAGIO EN IMÁGENES PARA LA PLATAFORMA DOCODE

El presente trabajo muestra el diseño de un sistema buscador de plagio en imágenes para el proyecto DOCODE que es una plataforma web que busca solución a la problemática de plagio académico y que facilita el análisis de originalidad de documentos. Esta herramienta pertenece al centro de investigación y docencia de la Universidad de Chile *Web Intelligence Centre* (WIC).

El desarrollo de este trabajo está enfocado a la búsqueda de imágenes por medio del sistema *Content-based image retrieval* (CBIR) dando énfasis en los resultados de medidas de desempeño, tiempo de ejecución y búsqueda en grandes bases de datos en función de los distintos métodos, como por ejemplo textura que usa herramientas de análisis de señales aplicadas para casos bi-dimensionales. Estos métodos permiten una descripción computacional de imágenes, con el objetivo de ser utilizados para un sistema de reconocimiento de patrones de la rama de la inteligencia computacional como lo es CBIR.

El sistema de detección de plagio se somete a distintas exigencias para probar su óptimo funcionamiento. Para esto se describe la metodología y se desarrolla un procedimiento para simular situaciones de plagio en imágenes, donde además también se mide la eficiencia de la descripción y búsqueda.

Los resultados son presentados en orden cronológico, para luego ser comentados y analizados en discusión de resultados. En esta sección se presentan las ideas principales durante el proceso de toma de decisiones para el diseño del sistema CBIR, justificando básicamente el proceso de selección de características.

Agradecimientos

A mis padres y hermanos, que si bien decidí no contarles en que estaba trabajando para esta memoria, siempre mostraron atención y preocupación por mi.

A todos los integrantes del Web Intelligence Centre, especialmente a los que estuvieron siempre cerca durante este trabajo y los buenísimos partidos de taca-taca después de los almuerzos.

Tabla de contenido

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

La era donde vivimos se caracteriza por la enorme facilidad de compartir información por medio del Internet. Este medio entrega espacios abiertos donde cualquier persona es capaz de compartir contenidos como texto, videos, audio y, en particular, lo que respecta a esta investigación, imágenes. Agreguemos a este fenómeno la invención de las cámaras digitales y escáner, que trajo con él la libertad al ser humano plasmar eventos de su entorno, y con ello a la mayor cantidad de fotografías digitales compartidas y subidas a la red [13][14].

Existe una enorme cantidad de recursos de conocimiento almacenados en la web, por ejemplo *Wikipedia*, incluso algunos de estos espacios pueden ser consultados de forma gratuita y de libre modificación tanto de recursos de texto como multimedia, esta gran facilidad de recursos que existe en nuestros tiempos hace a las personas crear información a partir de muchas fuentes disponibles, esto induce, en algunos casos, mostrar material de investigación ajena acusando de ser propios o sin dar crédito al verdadero autor del material utilizado.

Bajo este contexto, usualmente se comete, más por desconocimiento que por maldad, la acción de plagio la cual es definida por la real academia española como: *copiar en lo sustancial obras ajenas, dándolas como propias* [15]. El plagio ha llegado a tales extremos que ha sido observado en el ámbito académico o peor aún cuando se comete plagio con plena lucidez y conocimiento del hecho [16].

1.1. Motivación

El plagio es un acto reprochable que debe ser detectado y sancionado para evitar su extensión descontrolada de esta mala práctica, a raíz de estos hechos, han surgido sistemas capaces de automatizar la detección de plagio con el fin de resguardar el trabajo de autores en diversas áreas donde se desarrollan sus investigaciones.

Este trabajo se desarrolla bajo el ambiente de desarrollo de la plataforma de detec-

ción de plagio DOCODE, que constantemente está ampliando sus características, entre ellas la detección de plagio en imágenes. Algunos sistemas similares a DOCODE son: *Unicheck* (`https://unicheck.com/`), *The Plagiarism Checker* (`https://smallseotools.com/plagiarism-checker/`), *Turnitin* (`https://www.turnitin.com`), *Grammarly* (`https://www.grammarly.com/`). De estos sistemas de búsqueda de plagio contienen buscadores de imágenes como es el caso de *The Plagiarism Checker* que usa su sistema *Reverse Image Search* para encotrar plagio en imagenes, sin embargo este último sistema no está integrado al buscador de plagio en textos.

La búsqueda de plagio en imágenes es abordada usando *content based image retrieval* (CBIR) o Consulta de imágenes mediante ejemplo. CBIR es una técnica que permite el manejo de obtención de imágenes en una colección grande de base de datos.

Puesto que la cantidad de imágenes a buscar es enorme, nos motiva a encontrar una técnica de búsqueda de imagen por medio de reconocimiento de patrones, básicamente el sistema debe ser capaz de caracterizar las imágenes extrayendo características que sean únicas para describir el contenido de una imagen, además deben cumplir con ser simples de extracción para no malgastar recursos y de pocos datos para ahorrar espacio y tiempo de búsqueda. Por estas razones la técnica CBIR es elegida comúnmente para la realización de búsqueda de plagio de imágenes en una gran colección de estas [17, 18, 19].

1.2. Objetivo General

Diseñar un sistema de búsqueda de imágenes mediante ejemplo por medio de extracción de características que describan imágenes de forma única y la búsqueda de estas para retornar un set de imágenes con medida de similitud de plagio al usuario. Finalmente el sistema debe estar apto para su integración a la plataforma DOCODE que estará encargado de detectar, a partir de documentos entregados por el cliente, imágenes correspondientes a plagios de publicaciones o documentos con propiedad intelectual.

1.2.1. Objetivos específicos

- Investigar del estado del arte en búsqueda de imágenes.
- Diseñar e implementar un sistema CBIR para la detección de tipos de imágenes pertenecientes a base de datos de imágenes de propósito general.
- Determinar las transformaciones que corresponde a plagio en imágenes y realizar un ajuste fino al sistema para la detección de imágenes en la base de datos que cumplan con las transformaciones definidas como plagio.
- Diseñar un detector de citas en imágenes para omitir detecciones de plagio.

1.3. Antecedentes

La realización de este trabajo esta bajo el contexto de la actualización y ampliación de funcionalidades del proyecto DOCODE el cual se realiza en el Web Intelligence Centre de la Universidad de Chile.

1.3.1. Web Intelligence Centre

El Web Intelligence Centre, en adelante WIC, es un centro de investigación, servicios y docencia perteneciente a la Universidad de Chile cuya principal herramienta es el *Web Intelligence* que abarca diversas disciplinas de la inteligencia computacional y tecnología de la información enfocadas en la extracción de conocimiento a partir de datos en la web.

Misión, visión y objetivos del WIC son los siguientes [20]:

Misión

Crear tecnologías de información usando data science para apoyar la toma de decisiones en organizaciones que se interesan en innovar. El WIC cree que esta disciplina puede generar un gran impacto en la sociedad y eso los apasiona.

Visión

Para el año 2022 quiere ser un centro autosustentable de data science enfocado en salud con proyectos transferidos en 3 regiones/centros de salud pública y privada.

Objetivos

- Publicar en las principales revistas, conferencias y editoriales relacionadas con Web Intelligence.
- Proveer a servicio profesional, excelente y rápido para todos nuestros clientes.
- Dictar cursos de orientación práctica acerca de las Tecnologías de Información y su aplicación en los negocios.

1.3.2. DOCODE

DOCODE es un proyecto que nace como una solución a la problemática del plagio académico. Es un software creado para facilitar el proceso de análisis de originalidad de documentos digitales, acelerando su revisión y simplificando el trabajo de quienes tengan que revisar documentos. Existe la modalidad de comparación contra la web y comparación entre documentos [20]

DOCODE en su página web destaca las siguientes características:[21]

Contraste web
Valida la originalidad de tus documentos contra todas las páginas web que se encuentran indexadas a Google.

Contraste repositorio
Valida la originalidad de tus documentos contra un repositorio personal de archivos.

Informe de originalidad
Revisa las fuentes de plagio de tus documentos analizados, a través de un informe de originalidad intuitivo.

Integración vía API
Integra en tu flujo de trabajo al DOCODE y detecta la originalidad de tu contenido de manera natural y fluida.

1.4. Contribuciones de la memoria

Esta memoria contribuye a la expansión de características ofrecidas por la plataforma de búsqueda de plagio DOCODE, dando a este sistema la capacidad de incorporar una búsqueda de plagio en las imágenes en los documentos solicitados para revisión de sus clientes.

Durante el desarrollo de este trabajo se tuvo la oportunidad de escribir una publicación sobre uno de los métodos de búsqueda estudiados. El paper de estudio fue aceptado para la conferencia *Web Intelligence International Conference 2018* en calidad de *Regular Paper*.

1.5. Estructura

A continuación se presenta la estructura de esta memoria de título.

Marco conceptual: En este capítulo se explica todo lo necesario que debe manejar el lector para tener una mejor comprensión del resto de los capítulos. Conceptos como CBIR, extracción de características, base de datos, filtros, transformadas, etc..., serán abordados en este capítulo.

Metodología de evaluación: En este capitulo se describirán los métodos a utilizar para las pruebas del sistema de búsqueda CBIR como también la creación de un ambiente simulado para evaluar detecciones de plagio y funcionalidades del programa final.

Resultados: En este capítulo se presentarán los resultados obtenidos como por ejemplo las medidas de desempeño, tiempos de ejecución y ejemplos de resultados retornados como

imágenes serán expuestos.

Discusión de resultados: En este capítulo se explicarán las decisiones tomadas, a partir de los resultados de la sección anterior, para entender mejor la evolución del sistema durante las pruebas y funcionamiento del sistema final.

Conclusiones: En este capítulo se expondrá lo que se logró en esta memoria de título, como también los trabajos futuros que dejó.

Capítulo 2

Marco conceptual

En este capítulo se describen los conceptos utilizados para la implementación del sistema de búsqueda. El capítulo parte describiendo que es una característica y el espacio donde viven, luego introduce el concepto de CBIR, su funcionamiento y los distintos tipos de bases de datos de imágenes, el capítulo continúa mencionando las medidas de desempeño y la realización de las búsquedas para finalmente terminar describiendo los distintos métodos de extracción de características que se evalúan.

2.1. Espacio de características

El censado de muestras del mundo "real" requiere ser tratado de manera especial para obtener una correcta interpretación o visualización para una máquina, esta interpretación sugiere la creación de un espacio con multidimensional donde se desea que las características extraídas formen clusters, en caso contrario se dice que la características extraídas no son 'buenas'. La cualidad de formar clusters es fundamental para el retorno de clases en la etapa de clasificación, sin embargo la búsqueda de plagio en imágenes no posee clases (o la clase es solo una imagen), por lo que imposibilita la formación o un etiquetado para clusters en el espacio.

2.2. Content Based Image Retrieval

La consulta de imágenes mediante ejemplo (CBIR) es un sistema para la recuperación de imágenes digitales en una gran base de datos [22]. CBIR nace por la necesidad de búsqueda de imágenes digitales en un contexto donde de la cantidad de estas creció rápidamente en las últimas décadas, el sistema clásico de búsqueda basado en descripción del contenido por medio texto y palabras claves no fue suficiente para satisfacer las necesidades de búsqueda, por lo que fue necesario implementar un sistema basado en el contenido de la imagen de

consulta, entregando al sistema una imagen de consulta como único recurso [23]. CBIR es considerado el diseño de sistema de búsqueda de imágenes más eficiente donde el costo y el tiempo se reducen en gran medida. [24]

2.2.1. Funcionamiento de CBIR

El procedimiento para diseñar un sistema CBIR de búsqueda de imágenes, descrito en el diagrama de flujo de la figura 2.1 parte con la extracción de características de la base de datos de imágenes para formar sus vectores descriptores que son guardados para futuras consultas, estas características varían según el sistema a diseñar según requerimientos como tiempo de procesamiento, tipos de imágenes de la base de datos.

La búsqueda empieza cuando el sistema recibe una imagen de consulta donde las características de esta imagen son extraídas de igual forma a las de la base de datos, luego se procede a comparar la similitud de la imagen de consulta con la base de datos por medio de medidas de distancia entre sus vectores de características, el resultado final corresponde a la imagen cuya distancia de su vector de característica en la base de datos sea el mínimo con respecto a la imagen de consulta, además el sistema puede entregar un ranking con una serie de imágenes correspondientes a imágenes que presentan mínimas distancias con la imagen de consulta para ser retornadas según orden de relevancia en la búsqueda.

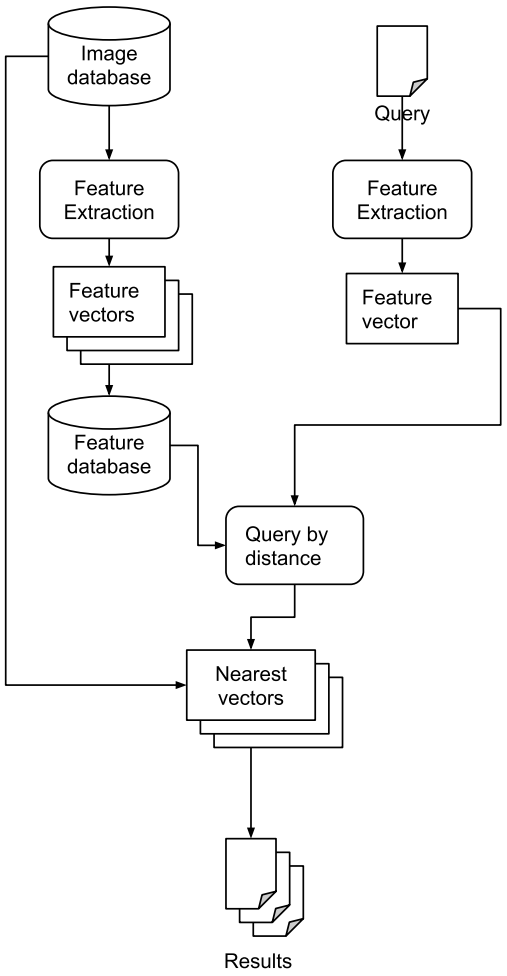


Figura 2.1: Diagrama de flujo de sistema CBIR. [1]

2.2.2. Base de datos

La estrategia para abordar el diseño de un sistema CBIR se ve influenciado fuertemente, si por ejemplo, la base de datos corresponde a imágenes muy similares, homogéneas, los cuales son generalmente más fáciles de tratar que una base de imágenes heterogénea [25]. La razón de esta diferencia es por la presencia de algoritmos especializados y el conocimiento experto para extraer características de mayor utilidad para las imágenes homogéneas que para las heterogéneas.

Los tipos de imágenes que pueden albergar la base de datos de un sistema CBIR pueden tener las siguientes condiciones nombradas en el listado a continuación [26].

Tamaño de la imagen: También razón entre ancho y altura.

Profundidad de color: Escala de grises, 8-bit, 16-bit, etc...

Condiciones de captura: Iluminación, distancia entre objeto y cámara, tipo de cámara,....

(los cuales pueden ser conocido o no)

número de objetos en la imagen: También tema tratado en la imagen.

Conocimiento acerca del tipo de objeto en la imagen: Por ejemplo saber con anterioridad si las imágenes contienen perros, gatos o caballos.

Origen: Natural o sintético.

Formato de archivo

Fondo: Conocimiento si los objetos de la imágenes están o no frente a un fondo controlado.

Esto hace que las bases de datos de imágenes se puedan clasificar como sigue:

muy específica

Corresponden a imágenes cuyas condiciones de toma son muy bien conocidas, estas tomas son logradas con la misma máquina bajo condiciones controlada de iluminación, ángulo, resolución, etc. Un ejemplo de este conjunto son las imágenes de exámenes médicas como las resonancias magnéticas. fig. 2.2

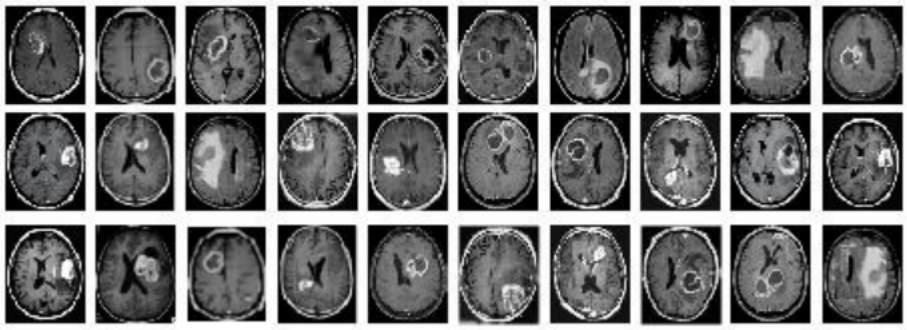


Figura 2.2: Imágenes de ejemplo de MRI de la base de datos *T2-weighted MRI database*. Fuente: [2].

específica

Son colecciones de imágenes que pueden pertenecer a varias clases, pero el proceso de obtención no es del todo controlado, por ejemplo se pueden encontrar imágenes con distinto color de fondo, objetos de interés captados a diferentes ángulos o en distintas posiciones de la imagen. fig. 2.3

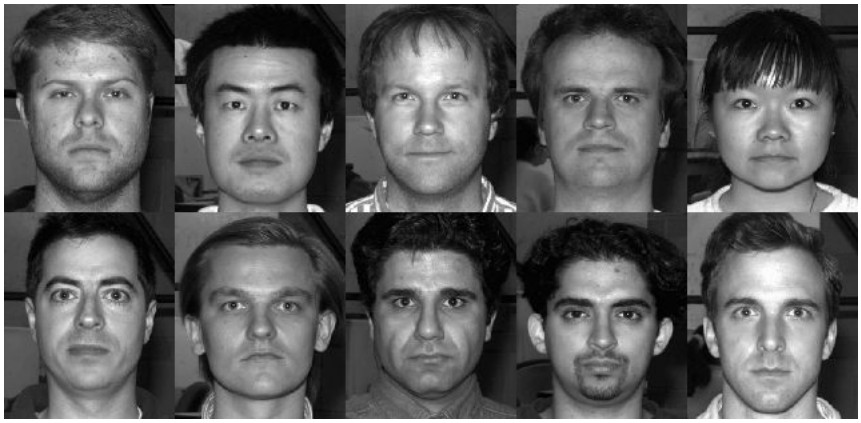


Figura 2.3: Ejemplos de imágenes de rostros. Base de datos. Fuente: [3].

Monotemático
Son imágenes donde solo se puede reconocer un tipo de objeto de interés.

General o heterogéneo
Cuando nada es sabido acerca de la colección de imágenes en la base de datos. Imágenes extraídas de internet corresponden a esta categoría.

2.2.3. Medidas de desempeño para CBIR

Se puede describir el desempeño de un sistema CBIR para un set de imágenes I, que recibe como consulta una imagen q, el cual tiene en la base de datos del sistema CBIR un conjunto $S \subset I$, también se define el conjunto de imágenes $R \subset I$ como el set total de imágenes recuperadas o retornadas por el sistema luego de una consulta [27].

Dada estas definiciones, la mejor manera de medir el sistema CBIR es por medio de las medidas utilizadas en el área de búsqueda y recuperación de información, *precision* y *recall* son definidos en las ecuaciones 2.1 y 2.2.

Precision Se define como la fracción de imágenes relevantes retornadas de la búsqueda dentro del set de imágenes retornadas por el sistema.

$$Precision(q) = \frac{|S \cap R|}{|R|} \tag{2.1}$$

Recall Se define como la fracción de imágenes relevantes retornadas de la búsqueda dentro del set de imágenes relevantes de la base de datos.

$$Recall(q) = \frac{|S \cap R|}{|S|} \tag{2.2}$$

Los sistemas CBIR pueden ser ajustados para retornar una cantidad de imágenes $R \subset I$

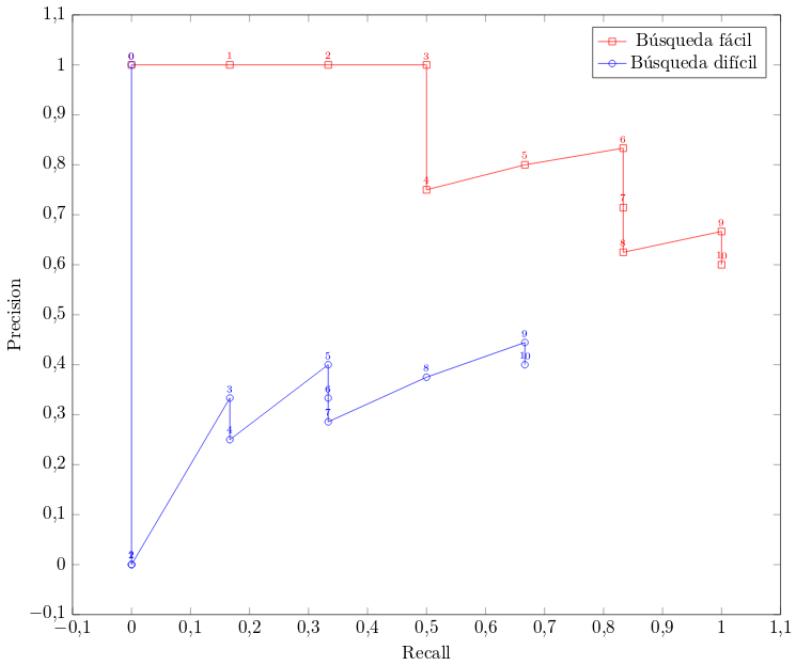


Figura 2.4: Gráfico de precision-recall para dos búsquedas hipotéticas. Los números sobre cada punto del gráfico muestra la cantidad de imágenes retornadas por el sistema. Asumimos precisión igual a 1 cuando el sistema retorna 0 imágenes. [1]

arbitraria por consulta, esta cualidad hace de las medidas de desempeño, precision y recall, tener una dependencia según este parámetro. Es Así como para obtener una medida de precision máxima, solo basta evaluar el sistema para una cantidad mínima de imágenes recuperadas, en particular una imagen $R = 1$, en cambio recall puede ser ajustado para obtener el valor máximo definiendo la cantidad de imágenes retornadas igual a la cantidad de imágenes en la base de datos, es decir $R = |I|$.

Una manera de ver estas dos medida de desempeño en conjunto es construyendo una gráfico precision-recall (PR) donde queda evidenciado como el sistema responde a medida que se solicitan más resultados por búsqueda, la figura 2.4 muestra un ejemplo de este gráfico.

Podemos observar del gráfico dos líneas correspondientes a dos búsquedas hipotéticas llamadas fácil (rojo) y difícil (azul), donde búsqueda fáciles corresponden a imágenes con menor cantidad de transformaciones para simular plagio con respecto a las imágenes de búsqueda difíciles, las búsquedas fáciles obtendrán usualmente un precision mayor a valores iguales de recall.

Los gráficos precision-recall puede contener mayor cantidad de información de la necesaria para evaluar un sistema CBIR rápidamente, es por esto que además existen los siguientes métodos para poder entender correctamente la información entregada por las medidas de precision y recall a partir de la evaluación en puntos claves.

- P(N): Precision después de N imágenes retornadas.

- R(N): Recall después de N imágenes retornadas.
- R(P^*): Recall justo después que precision cae por debajo del umbral P^*.

La evaluación de un sistema CBIR para detección de plagio conlleva a que cada imagen de consulta solo tenga una imagen relevante en la base de datos, esto se produce debido a que la imagen original donde salen los plagios es única. Esta definición fuerza al sistema a tener como valor de imágenes relevantes para cada búsqueda igual a $S = 1$.

La medida de recall solo tendrá valores binarios: 0 cuando el sistema CBIR no retorna la imagen relevante y 1 en caso contrario, sin embargo el promedio de estos valores a los largo de toda la base de datos de imágenes de consulta de plagio simulado obtiene valores para recall útiles interpretados como probabilidad de detección.

En cambio para precision, este hecho produce un problema más grave, dando como resultados valores binarios: 0 y $1/R$, los cuales si se promedian, retornan los mismos valores de recall ponderado por $1/R$.

2.3. Búsqueda eficiente en espacio de características

Para cada consulta, el sistema debe realizar una búsqueda en el espacio de características conformado por un gran de puntos o elementos que es igual al número de imágenes registradas en la base de datos del sistema para la detección de plagio. La distancia de la imagen de consulta a cada uno de los puntos de la base de datos resulta ser una tarea que abarca un tiempo de ejecución altísimo si se realiza de forma bruta.

Para optimizar este problema se presenta el algoritmo *Ball Tree* que ayuda a consultar los resultados del método de clasificación supervisada KNN entrenado de forma rápida y eficiente.

2.3.1. Ball Tree

Ball tree es una estructura de datos que particiona el espacio multidimensional para aumentar el rendimiento de las búsquedas, generalmente se utiliza para optimizar las búsquedas en KNN. Esta estructura de datos obtiene su nombre por el hecho que divide el espacio el hiperesferas o bolas que almacenan puntos del espacio en conjuntos disjuntos, en ellas los lo puntos distan del centro o pivote en forma constante según una distancia pre-establecida con anterioridad su construcción.

El algoritmo de construcción se encarga de organizar las hiperesferas o bolas en estructura de árbol binario, dando así a cada nodo dos hijos que almacenan puntos disjuntos. La construcción del árbol usa la estrategia *top-down*, este parte definiendo el nodo raíz como

la hiperesfera que contiene todos los puntos del espacio, para luego empezar a ramificar a conjuntos más pequeños.

La velocidad de la búsqueda depende fuertemente de la cantidad de número de nodos del árbol, un buen número a elegir corresponde a la desigualdad 2.3. [28]

$$n^\circ \ nodos \leq n^\circ \ puntos \leq 2 * n^\circ \ nodos \tag{2.3}$$

La construcción de Ball Tree tiene una complejidad computacional de $\mathcal{O}(CR \log R)$ donde R es el número de puntos y C es el costo de calcular la distancia punto a punto, finalmente la complejidad computacional de realizar la búsqueda es de $\mathcal{O}(d \log R)$ donde d denota la dimensionalidad de cada uno de los puntos, sin embargo el tiempo de ejecución empeora a medida que la dimensionalidad crece más allá de las 30 dimensiones [28].

2.4. Procesamiento de imágenes

En esta sección se explicará la propiedad de espacio de color y texturas de imágenes, junto con técnicas que permiten aprovechar estas propiedades para describir imágenes de manera única y acercar un poco a la comprensión de descriptores.

2.4.1. Espacio de colores

Gran parte del mecanismo de percepción de imágenes realizada por el ser humano utiliza los colores, puesto que el color es una interpretación del cerebro para diferenciar las distintas frecuencias de la luz que es reflejada por los objetos, esta interpretación queda bastante abierta para los sistemas computacionales que debieron adaptar distintas bases para lograr la similitud de colores de la percepción humana. Los dos espacios principales de color utilizados para extracción de características son: RGB y HSV.[29]

Para mostrar el uso de un espacio de color, se tiene como ejemplo el espacio RGB, el cual se define generalmente como un cubo unitario donde cada una de sus tres dimensiones representan los colores rojo, verde y azul. El computador reserva 8 bits para cada componente para este espacio, por lo que permite crear un total de $255^3 = 16{,}7$ millones de representaciones para colores.[29]

Para crear un sistema capaz de buscar contenido en imágenes es útil tener como ejemplo a seguir la percepción humana del color. Al contrario del computador, donde el manejo del color generalmente es hecho de el espacio RGB, tenemos que le resulta sencillo a la percepción humana diferenciar entornos que presenten las mezclas de colores aditivos producto del solapamiento de luz, por ejemplo para un sistema computacional el amarillo puro se define como la mezcla de rojo y verde con valores 255 para ambos y un valor nulo para azul, sin

embargo, el ser humano no puede, a través de sus sentidos, obtener de forma natural el hecho de que el color amarillo proviene de la mezcla de dos colores, es decir, que la búsqueda de un color amarillo no se ve afectada por otros colores como el rojo y azul como es el caso de imágenes RGB.

El espacio de color HSV, figura 2.5, tiene la característica que en sus bases la variación de sus valores es distinguible para percepción humana [30], por ejemplo el valor de matiz (Hue) hace un barrido por los colores desde rojo hasta violeta, Saturación (Saturation) denota la púreza del color seleccionado en matiz e intensidad (Value) que selecciona el grado de iluminación de color, el valor nulo de esta último siempre da como resultado el negro.

El ser humano ha sido capaz de adaptar la percepción de colores en situaciones donde la fuente de luz no es suficiente para revelar el verdadero color de los objetos, por esta razón que el valor de intensidad (Value) en el espacio HSV es menos sensible a la variación para la visión humana.

La transformación entre espacios colores RGB a HSV queda definida por la fórmula 2.4 aplicada por cada pixel de la imagen

$$H = \begin{cases} No\ definido & \text{si } MAX = MIN \\ 60° \cdot \frac{G-B}{MAX-MIN} + 360° & \text{si } MAX = R \ y \ G < B \\ 60° \cdot \frac{B-R}{MAX-MIN} + 120° & \text{si } MAX = G \\ 60° \cdot \frac{B-R}{MAX-MIN} + 240° & \text{si } MAX = B \end{cases} \tag{2.4}$$

$$S = \begin{cases} 0 & \text{si } MAX = 0 \\ 1 - \frac{MIN}{MAX} & \text{En caso contrario} \end{cases}$$

$$V = MAX$$

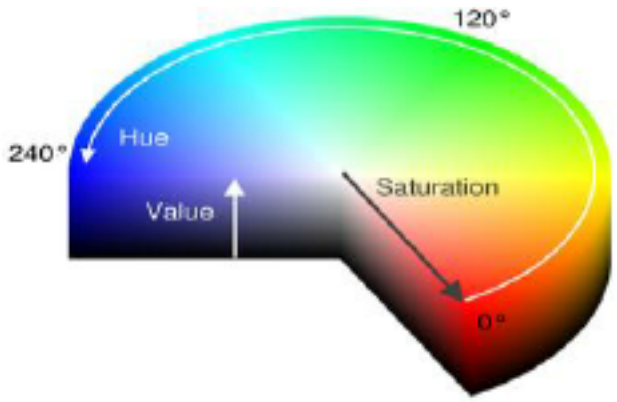
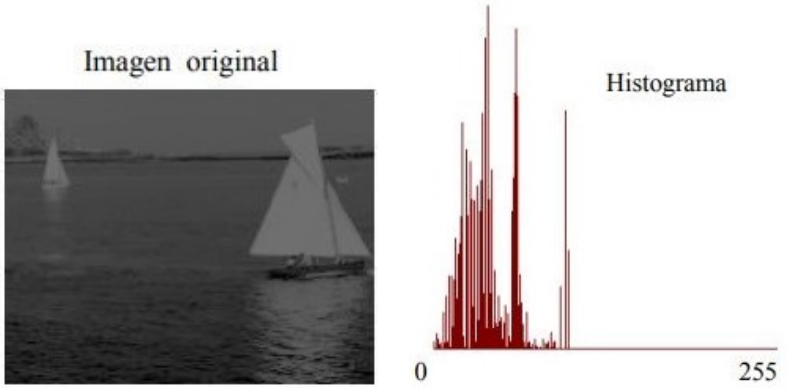


Figura 2.5: Espacio de color HSV [4].

2.4.2. Ecualización de histogramas

La ecualización de histogramas es una técnica que ajusta los valores de intensidad de pixeles de tal manera para obtener un mejor aprovechamiento del rango de valores de estos, como resultado final se obtiene un mejoramiento del contraste de una imagen a color o monocromática.

Esta técnica utiliza los histogramas de frecuencia acumulada, puesto que la forma del histograma permite obtener la información del contraste de la imagen, por ejemplo en la figura 2.6 se observa que una imagen de bajo contraste genera un histograma de frecuencias acumuladas estrecha [5].



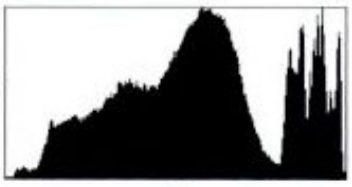
(a) Imagen original. (b) Histograma de la imagen (a).

Figura 2.6: Imagen de bajo constaste con su respectivo histograma. [5].

La técnica de ecualización se encarga de modificar los valores de pixeles para obtener una histograma de frecuencias acumuladas de forma que se ajuste a una función lineal para asegurar que cada valor de pixel tiene la misma tasa de ocurrencia en la imagen. La figura 2.7 muestra un resumen de ecualización de imágenes.

(a) Imagen original sin ecualización.

(b) Resultado de imagen (a) con ecualización de histogramas aplicada.



(c) Histograma de niveles de grises de (c).

(d) Histograma de niveles de grises de (b) después de la ecualización.

(e) Histograma acumulado antes de la ecualización.

(f) Histograma acumulado después de la ecualización.

Figura 2.7: Detalles del proceso de ecualización de histogramas. [6]

2.4.3. Transformación log-polar

La transformación log-polar de una imagen convierte el dominio habitual de una imagen en coordenadas cartesianas a coordenadas logarítmicas polares, la ecuación 2.5 es usada para un primer paso que consiste en la transformación a coordenadas polares donde $p(a, r)$ es la imagen en coordenadas polares y $f(i, j)$ es la imagen original de tamaño $N \times N$.

$$p(a,r) = f\left(\left\lfloor \frac{N}{2} \right\rfloor + \left\lfloor r\ cos\left(\frac{2\pi a}{S}\right)\right\rfloor, \left\lfloor \frac{N}{2} \right\rfloor - \left\lfloor r\ sen\left(\frac{2\pi a}{S}\right)\right\rfloor\right) \tag{2.5}$$

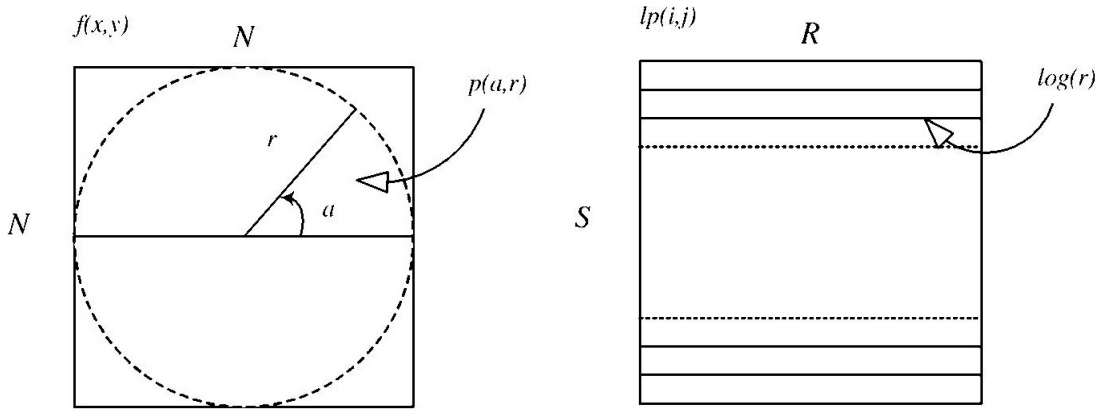
Donde $a \in \{0, ..., S-1\}$ y $r \in \{0, ..., \left\lfloor \frac{N}{2} - 1 \right\rfloor\}$

La segunda etapa consiste en aplicar logaritmo a la componente radial de la imagen $p(a, r)$ para construir finalmente la imagen log-polar $lp(i, j)$ de tamaño $S \times R$ según la ecuación 2.6.

$$lp(i,j) = p\left(i, \left\lfloor \frac{log_2(j+2)}{log_2(R+2)} \right\rfloor \cdot \left\lfloor \frac{N}{2} \right\rfloor\right) \tag{2.6}$$

La zonas radiales (muestreadas por el parámetro S), que emergen desde el centro de la imagen, se ajustan para posicionarse a lo largo de las filas en la imagen log-polar como se puede apreciar en la figura 2.8, por lo tanto las rotaciones son representadas en este espacio con cambios circulares en las filas de la imagen log-polar [31, 32]. Un ejemplo de esta

transformacion se puede apreciar en la figura 2.9.



(a) Imagen con valores de pixeles representados en coordenadas polares.

(b) Representación de valores de pixeles en coordenada cartesianas tras transformación log-polar.

Figura 2.8: Esquema explicativo de transformación log-polar.[7].

(a) Imagen rotada.

(b) Transformación log-polar de la imagen (a).

Figura 2.9: Ejemplo de una imagen rotada al aplicarle transformación log-polar.

2.4.4. Descriptores

A continuación se presentan distintos algoritmos de extracción de características a imágenes. La extracción requiere métodos cuyos retornos sean capaces de representar de forma concisa y única las imágenes para formar los vectores de características, almacenarlos y realizar las consultas en base a esta información.

2.4.4.1. Histogramas de colores

Los histogramas de colores se utilizan para obtener información de los colores y usarlos directamente como características para la clasificación, estos consisten básicamente en un histograma por cada componente del espacio de color. Los histogramas guardan la frecuencia de ocurrencia de un determinado valor de pixel en la imagen, estos valores de pixeles son guardados en los histogramas según su pertenencia a un conjunto de valores para reducir la dimensionalidad de esta característica y simplificar la búsqueda en la base de datos.

Los histogramas, generalmente son calculados utilizando frecuencia relativa de sus valores de pixeles, la ecuación número 2.7 muestra como se obtienen este histograma. [5] Ejemplos de estos histogramas se muestran en las figuras 2.6 y 2.7.

$$H(k) = \frac{n^\circ \textit{ de pixeles de intensidad } k}{N \cdot M} \tag{2.7}$$

La búsqueda de histogramas se realiza generalmente con distancias estándar como distancia euclideana, ecuación 2.8, o métricas diseñadas especialmente para este tipo de problemas como: correlación, chi-cuadrado, manhattan o distancia de Bhattacharyya.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (a_i(x) - a_i(y))^2} \tag{2.8}$$

2.4.4.2. Local binary pattern

Local binary pattern o LBP es un descriptor de texturas usado por su invarianza al cambio de niveles de iluminación, y por ende también a niveles de escalas de grises, sin embargo como todo descriptor de texturas, tiene problemas con las rotaciones [33].

Para obtener las características LBP de una imagen, primero esta se convierte a escala de grises. Para cada pixel de la imagen, se encuentran sus P pixeles vecinos a un radio de distancia R pixeles, definimos el vector $(g_c, g_0, g_1, ...g_{P-1})$ como los niveles de grises de los pixeles g_0 como el central y los $g_0, g_1, ...g_{P-1}$ el de sus vecinos.

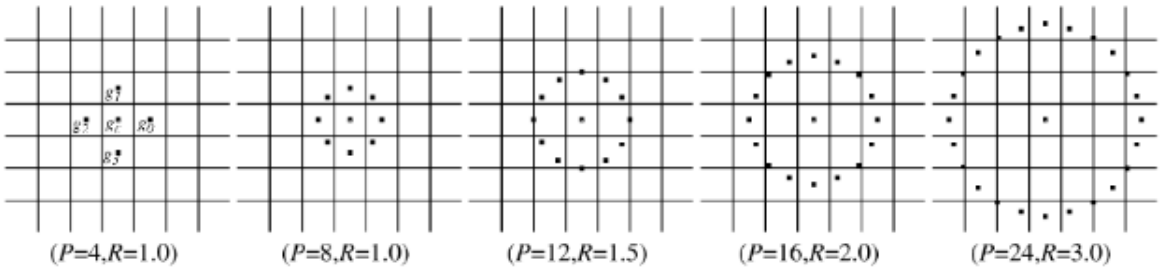


Figura 2.10: Pixeles vecinos LBP para distintas configuraciones de parámetros P y R.[8]

Para encontrar los pixeles vecinos dado los valores P y R, se utiliza la fórmula de la ecuación 2.9 que generan vecinos cercanos de forma radial como muestra la figura 2.10.

$$g_p = \left(-R \cdot sen\left(\frac{2\pi p}{P}\right), R \cdot cos\left(\frac{2\pi p}{P}\right)\right) \ \forall p \in (0, 1, ..., P-1) \tag{2.9}$$

Donde se considera la posición del pixel central g_c como $(0, 0)$.

Luego se compara el valor de cada pixel vecino con el pixel central usando el vector de guarda las diferencias de la ecuación 2.10

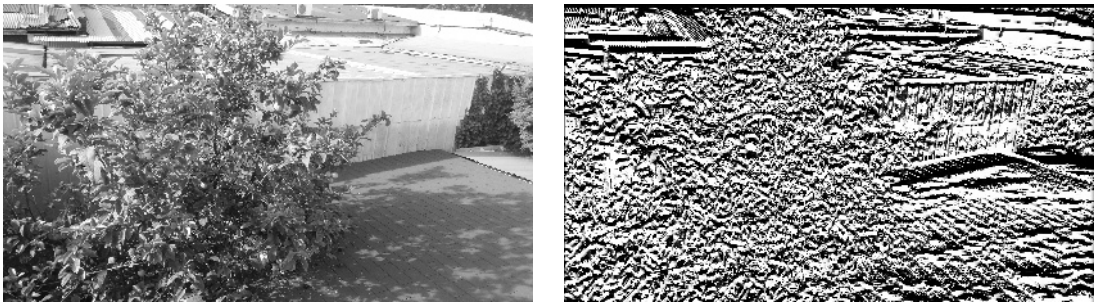
$$(g_c - g_0, g_c - g_1, g_c - g_2, ..., g_c - g_{P-1}) \tag{2.10}$$

Este vector es convertido a un número binario, la transformación de la ecuación 2.11 es usada para representar solamente si los niveles de grises de los vecinos es mayor o menor al pixel central.

$$s(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ \\ 0 & \text{si } x < 0 \end{cases} \tag{2.11}$$

Luego de aplicar la transformación de la figura 2.11 a cada componte del vector 2.10 se obtiene un numero binario de largo P, y en particular si $P = 8$, el número binario generado se puede convertir a decimal, como muestra en la ecuación 2.12, para una representación visual a escala de grises si se reemplaza el valor del pixel central por el valor obtenido. La figura 2.11 muestra la transformación LBP aplicada a una imagen.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{2.12}$$



(a) Imagen original en escala de grises. (b) Transformación LBP de la imagen (a).

Figura 2.11: Ejemplo de una imagen en escala de grises al aplicarle transformación LBP.

Para finalizar, existe un método que otorga a LBP cierta invarianza a la rotación, este método consiste en alterar el orden del vector 2.10 según cambios circulares. Por lo tanto si la función $ROR(x, i)$ rota circularmente un vector x en i espacios, el método de invariante a rotaciones de LBP^{ri} queda definido como muestra la ecuación 2.13.

$$LBP^{ri}_{P,R} = \text{mín}\{ROR(LBP_{P,R}, i)\} \mid i = 0, 1, ..., P-1 \tag{2.13}$$

2.4.4.3. Firma de imagen

La firma de imagen o *image signature* corresponden a métodos de extracción de características usando básicamente heurísticas para determinar elementos propios de una imagen para ser utilizados como descriptores o vectores de características. Para las pruebas, se implementará el método de *image signature* propuesto por *Wong, H. C., Bern, M., Goldberg, D* [9].

Este tipo de descriptor fue diseñado para la detección de imágenes duplicadas (fig. 2.12) y para poseer robustez a transformaciones de re-dimensionamiento, re-escaneado y compresión de imágenes pero no para manejar bien los recortes y rotaciones dentro de grandes bases de datos.

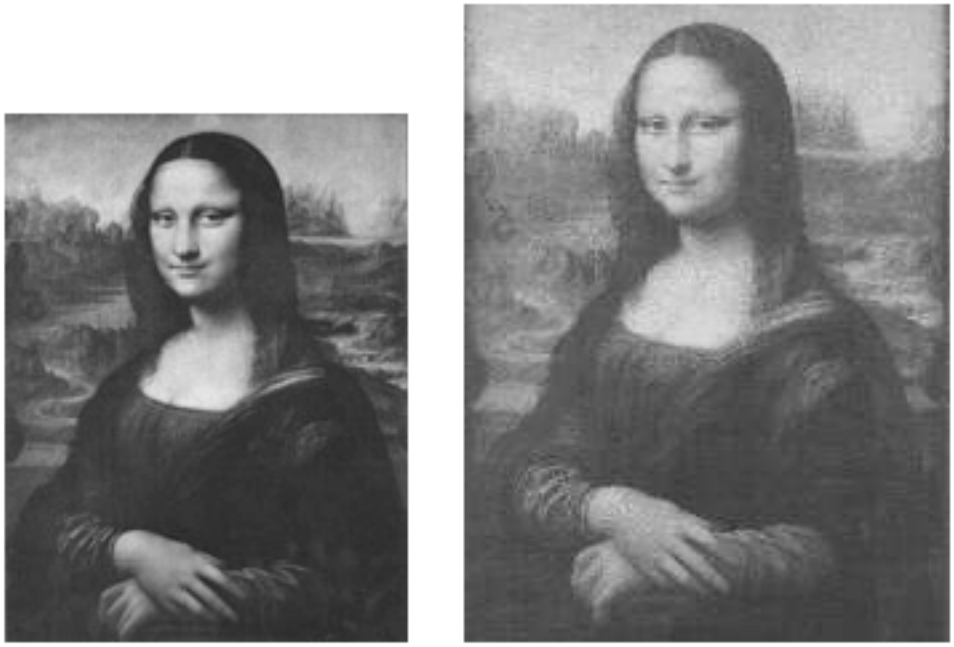


Figura 2.12: Ejemplo de imagen duplicada descargadas de internet donde se observa dos versiones de escala de grises producto de la distinta compresión JPEG. [9]

Para extraer las características de una imagen, primero se transforma a escala de grises si la imagen es a color, luego se realiza un proceso de selección de filas y columnas con mayor información para cortar la imagen producto de recorte de la imagen por medio de la eliminación de filas y columnas.

La selección de columnas parte computando la suma de las diferencias de pixeles en cada una de las columnas de la imagen, luego se eliminan aquellas columnas que contengan al más del 5 % del total de la suma, de igual manera se realiza la acción descrita anteriormente para la selección de filas, la imagen recortada final de divide en cuadrillas de 9×9.

A cada cuadrilla se rellena con el nivel de gris dominante, como se muestra en la figura 2.13, el cual se obtiene seleccionando una región cuadrada de $P \times P$, donde P es calculado según la ecuación 2.14, esta región cuadrada es centrada en la mitad de cada cuadrilla y es en esta región donde se toma como nivel de gris dominante al promedio de grises dentro de $P \times P$.

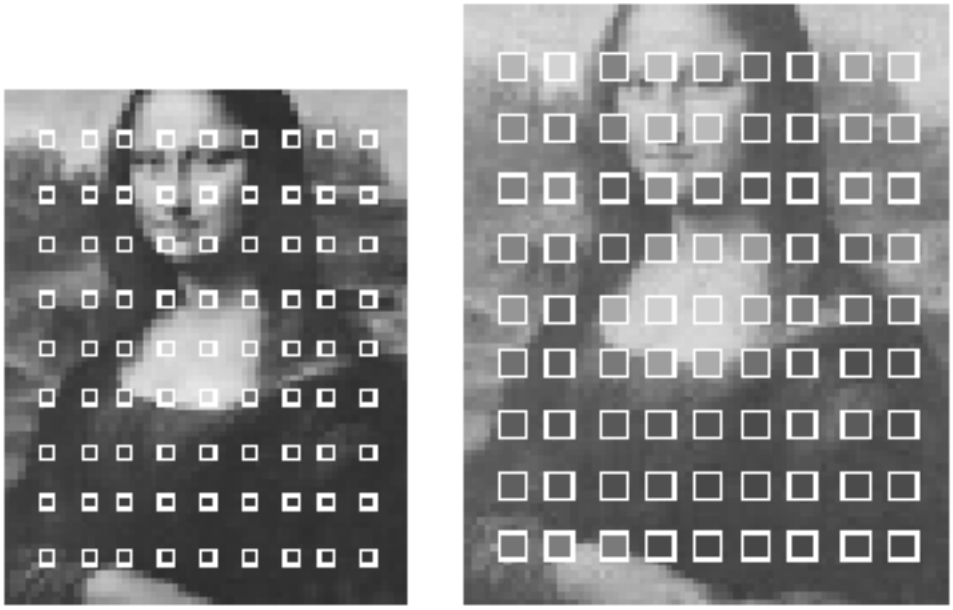


Figura 2.13: Cuadrillas de 3×3 y 5×5 con su respectivo nivel de gris dominante. (en la imagen se muestran las cuadrillas de menor tamaño para reconocer la imagen de fondo).[9]

$$P = \text{máx}\left\{2, \lfloor 0{,}5 + \text{mín}\left\{n, m\right\}/20 \rfloor\right\} \tag{2.14}$$

En este punto se da por concluído la etapa de pre-procesamiento de la imagen, solo basta obtener el vector de características, para ello se recorren todas cuadrillas y se comparan los niveles de grises dominantes con las 8 cuadrillas vecinas con la seleccionada. Para describir las comparaciones, se adopta el siguiente código de 5 símbolos: -2,-1,0,1 y 2, los cuales codifican los siguientes significados: más oscuro, oscuro, igual, iluminado y más iluminado. Finalmente las cuadrillas que no poseen 8 vecinos, se rellenan su ausencia con valores 0.

De esta forma se obtienen vectores de largo 648, que facilitan la búsqueda por la base de datos por no ser de tener gran dimensionalidad.

2.4.4.4. Momentos de color

Corresponde al tipo de extracción de características de color, este método hace referencia a que la distribución de color se puede modelar como una distribución de densidad de probabilidad, donde las características corresponden generalmente a los tres primeros momentos de la distribución de probabilidad [34], como por ejemplo la media (2.18), desviación estándar (2.16), Skewness (2.17) para ajustes gaussianos.

$$E_i = \sum_{j=1}^{N} \frac{1}{N} p_{ij} \tag{2.15}$$

$$\sigma_i = \sqrt{\frac{1}{N} \sum_{j=1}^{N} (p_{ij} - E_i)^2} \tag{2.16}$$

$$s_i = \sqrt[3]{\frac{1}{N} \sum_{j=1}^{N} (p_{ij} - E_i)^3} \tag{2.17}$$

Donde N es el número de pixeles y p_{ij} es el valor del pixel del índice *i,j* de la imagen.

2.4.4.5. Correlograma de color

La correlación de color se usa como una corrección para los fallos de presenta los histogramas de colores, en este caso , la correlación de color es capaz de tomar en cuenta la probabilidad de encontrar un color a una distancia fijada a priori a partir de un pixel de prueba. La formulación es la siguiente: sea una imagen I y sin perder generalidad, de tamaño cuadrado de nxn y con una cantidad m de colores denotados por $c_1, c_2, ..., c_m$. Para u pixel p con posición(x, y) se define el operador I que entrega el color del pixel P y el conjunto $I_c = \{p|I(p) = c\}$, es decir todos los pixeles p que tienen color c. El conjunto de pixeles corresponde a un espacio con una métrica con normas bien definidas, para calcular la distancia entre dos pixeles $p_1 = (x_1, y_1)$ e $p_2(x_2, y_2)$ se usa la norma L_∞, es decir $|p_1 - p_2|_\infty = max\{|x_1 - x_2|, |y_1 - y_2|\}$.

El correlograma de I según una distancia d fijada a priori se calcula como sigue: Para $i, j \in 1, 2, ..., m$

$$\gamma_{i,j}^{(d)}(I) = \mathbb{P}_{p_1 \in I_{c_i}, p_2 \in I}[p_2 \in I_{c_j} || p_1 - p_2| = d] \tag{2.18}$$

Donde la probabilidad se obtiene del histograma de color normalizado[35].

2.4.4.6. Matriz de co-ocurrencia de niveles de grises

La matriz de co-ocurrencia de niveles corresponde a un descriptor de texturas que guarda la frecuencia de aparición de todas las combinaciones de dos valores de pixeles, de referencia y vecino. Esta matriz aproxima aproxima la probabilidad de distribución conjunta de un par de pixeles llamados $par(g, g')$ [36].

Dado un vector de desplazamiento $\tau = (\Delta_x, \Delta_y)$, se construye cada elemento de la matriz

de co-ocurrencia ($A_\tau(g, g')$) según la ecuación 2.19.

$$A_\tau(g, g') = \{par(g, g')|I(x, y) = g \wedge I(x + \Delta_x, y + y\Delta_y) = g'\} \tag{2.19}$$

Donde $par(g, g')$ corresponde a los dos valores de niveles de gris

2.4.4.7. Filtrado de imágenes

Las máscaras o kernel se utilizan para realizar filtrado en imágenes, la técnica empleada corresponde a una operación de convolución discreta en dos dimensiones, ecuación 2.20, donde los elementos participantes son los datos otorgados por los pixeles de la imagen y los valores de una matriz llamada máscara o kernel del filtro.

$$f[x, y] * g[x, y] = \sum_{n=1}^{N} \sum_{m=1}^{M} f(\tau_1, \tau_2) g(x - \tau_1, y - \tau_2) d\tau d\tau \tag{2.20}$$

La ecuación 2.20 muestra la aplicación de convolución para una imagen de dimensión $N \times M$ y $g[x, y]$ son los valores del filtro aplicado a la imagen $f[x, y]$.

Al igual que la convolución de una dimensión, existe una parte móvil. La máscara recorre cada pixel de la imagen y realiza una multiplicación con sus vecinos cubiertos por el tamaño de máscara, estos resultados son sumandos y son parte de la imagen filtrada como aparece en la figura 2.14.

Las imágenes se pueden interpretar como señales de dos dimensiones y por ende se entiende que poseen componentes de frecuencias. La operación de convolución de mascaras, anteriormente descrita, corresponde a realizar una aplicación de filtro en señales digitales.

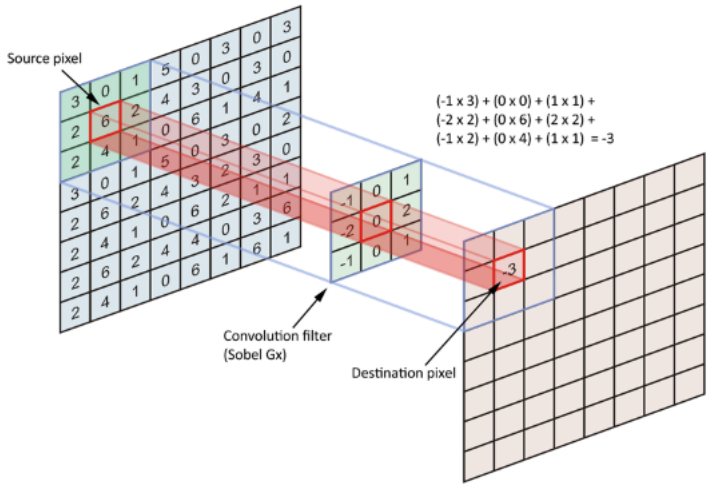


Figura 2.14: Aplicación de convolución discreta de dos dimensiones sobre un punto de una imagen.[10]

Un ejemplo de la aplicación de filtrado sobre una imagen se muestra en la figura 2.15, donde el kernel utilizado corresponde a la ecuación 2.21.

$$Kernel = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.21}$$

(a) Imagen original. (b) Imagen filtrada.

Figura 2.15: Ejemplo de filtrado de imagen.

2.4.4.8. Transformada de Fourier

La transformada de Fourier se utiliza generalmente en análisis y procesamiento de señales para cambiar el dominio de la señal desde el dominio temporal al dominio frecuencial. Básí-

camente la transformada de Fourier utiliza una familia ortonormal de funciones sinusoidales generadas a partir de una base exponencial compleja capaz de barrer todas las frecuencias, ecuación 2.22, a partir de esta base, la transformada encuentra la correlación que existe en cada elemento de la base (frecuencias) con la función objetivo, esta función o señal a transformar se somete a una proyección con cada función de la base ortonormal que otorga finalmente el coeficiente que vive en el cambio de dominio. La ecuación 2.23 muestra la transformada de Fourier continua en una dimensión.

$$\{e^{-j2\pi fx}\}_f \tag{2.22}$$

$$\hat{f} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(x) e^{-j2\pi fx} dx \tag{2.23}$$

La aplicación que tiene la transformada de Fourier es muy variada, desde telecomunicaciones hasta procesamiento de imágenes y aplicaciones en la medicina

La figura 2.16 muestra la transformada de Fourier para un pulso unitario, este resultado es ampliamente utilizado en telecomunicaciones donde recibe el nombre de función Sampling.

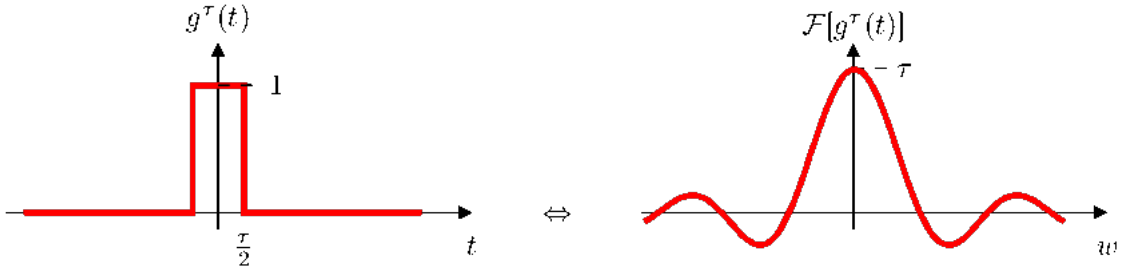


Figura 2.16: Transformada de Fourier de un pulso unitario.

Esta transformada se puede extender a su forma discreta para casos de funciones de dos dimensiones, como es el caso de las imágenes y lo que concierne al trabajo a realizar. La ecuación 2.24 muestra la transformada de Fourier para imágenes.

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2\pi i \left[\frac{ux}{M} + \frac{vy}{N}\right]} \tag{2.24}$$

Donde M y N son la dimensiones en píxeles de altura y ancho de la imagen, también se puede apreciar las componentes de ortogonalización de cada frecuencia $\frac{x}{M}$ y $\frac{y}{N}$ que corresponden a las frecuencias discretas para formar la exponencial compleja y pueden ser indexadas de forma discreta según el tamaño de la imagen. La figura 2.17 muestra la imagen de un círculo. Bajo el punto de vista de análisis de señales de una dimensión, si uno mira la imagen

desde el centro, se puede notar que el círculo genera pulsos unitarios en todas sus direcciones, por lo tanto su transformada de la figura 2.18 corresponde a funciones Sampling, esta función es continua en el espacio frecuencias, por lo tanto está conformada por infinitas funciones y dicho de otra manera, se necesitan infinitas funciones en combinación lineal para formar un pulso unitario.

Figura 2.17: Imagen de ejemplo antes de aplicar transformada de Fourier.

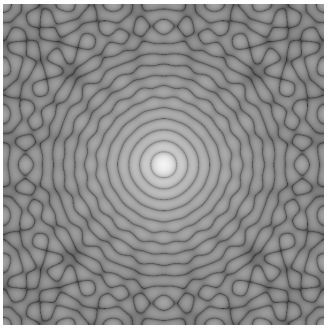


Figura 2.18: Imagen de ejemplo después de aplicar transformada de Fourier.

Al aplicar un filtro, se puede observar como las frecuencias de la imagen original son afectadas según la naturaleza del filtro, por ejemplo la imagen de la figura 2.19 muestra la transformada de Fourier en la figura 2.17 al aplicarle un filtro Gaussiano, el cual corresponde a un filtro pasa bajo.

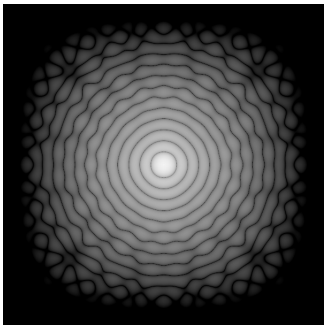


Figura 2.19: Filtrado de frecuencias altas en espectro de Fourier de imagen de prueba.

Un problema de la transformada de Fourier es la pérdida de características de frecuencia con evolución o existencia temporal, debido a que la transformada fue concebida para señales estacionarias. Esta característica se presenta en imágenes que corresponden básicamente a señales discretas en dos dimensiones [37].

2.4.4.9. Transformada Wavelet

La transformada Wavelet cambia el dominio temporal de una señal a dominio frecuencial utilizando como funciones base con características de poder generar distintas traslaciones (tiempo) y distintas dilataciones (o cambios de escala) por medio de una función Wavelet madre generadora de bases, a diferencia de Fourier que usaban todo el dominio del tiempo en sus cálculos. El cambio de escala en las bases permite obtener los detalles de una frecuencia específica y su lapso de tiempo de ocurrencia queda determinado por la traslación temporal de la base aplicada, por ejemplo al usar una escala más amplia, los suceso de frecuencia baja son detectados.

Para imágenes, la transformada Wavelet puede ser aplicada para casos discretos bidimensionales utilizando de forma recursiva filtros pasa-altos y pasa-bajos con cambios de resolución en la imagen, estos efectos son representados por el siguiente esquema de aplicación de filtros de la figura 2.20, algoritmo conocido como piramidal o de Mallat [38].

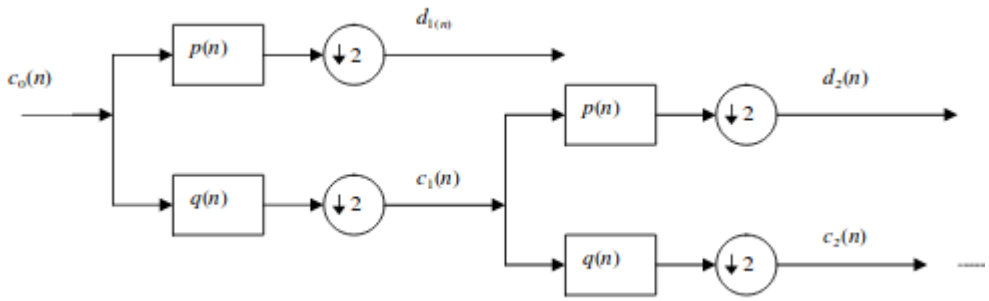


Figura 2.20: Algoritmo piramidal para descomposición Wavelet de imágenes [11].

Donde $C_0(n)$ corresponde a la imagen original, $p(n)$ y $q(n)$ filtros pasa-alto y pasa-bajo respectivamente, luego de cada filtrado la imagen es re-escalada a la mitad de su tamaño, para luego obtener los coeficientes o imagen $d_i(n)$, por otra lado, los coeficientes o imágenes $c_i(n)$ son iterados nuevamente por ambos filtros y cambiados de escala para obtener el nuevo coeficiente $d_{i+1}(n)$, las iteraciones terminan con el ajuste del valor 'i'que representa el nivel de profundidad de aplicación de la descomposición Wavelet o hasta donde los filtros no puedan actuar debido al pequeño tamaño de las imágenes de los niveles más profundos.

La figura 2.21 muestra como se forma una grilla de distinto muestreo o resolución correspondientes a la aplicaciones de un banco de filtros pasa-banda sobre la misma imagen de prueba, el conjunto da finalmente la transformada de Wavelet sobre una imagen.

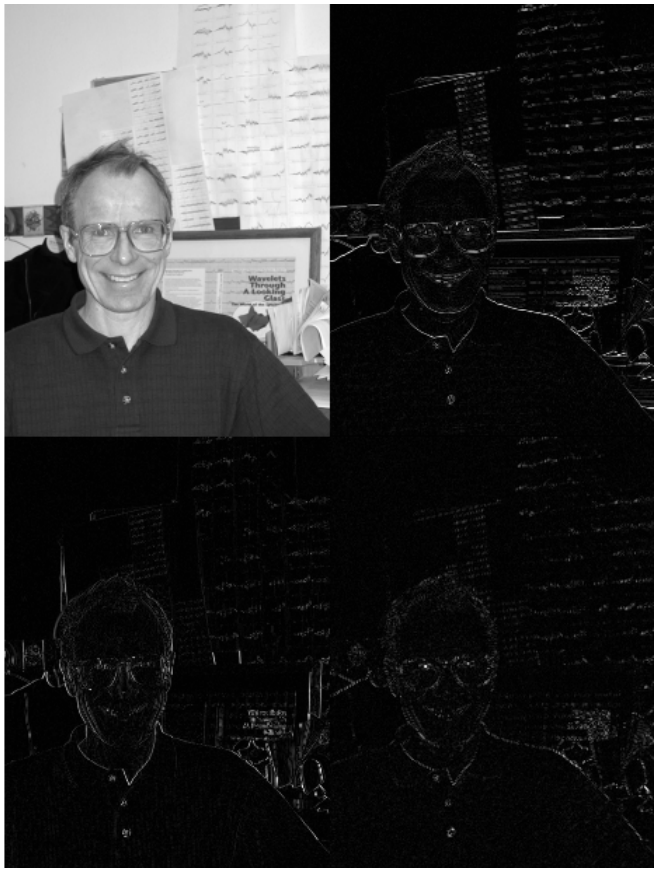


Figura 2.21: Transformada Wavelet de una imagen [12].

Existen variantes de esta transformación donde cambian las bases ortonomales [39].

2.4.4.10. Filtros Gabor

El filtro Gabor ha sido utilizado ampliamente en CBIR para la extracción de características del tipo texturas en imágenes [40], este filtro corresponde básicamente a la aplicación de bancos de filtros generados por la ecuación 2.25 que es la función Gabor de dos dimensiones.

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right) + 2\pi jWx\right] \tag{2.25}$$

Donde σ_x y σ_y son la desviaciones estándar de una distribución Gaussiana a lo largo de los ejes x e y. Los parámetros de diseño de los filtros son m y θ de las ecuaciones 2.26, 2.27 y 2.28. El primer parámetro m ajusta la escala del filtro y segundo θ ajusta la orientación del filtro.

$$g_{mn}(x,y) = a^{-m} g(x',y') \tag{2.26}$$

$$x' = a^{-m}(x\ cos(\theta) + y\ sen(\theta)) \tag{2.27}$$

$$y' = a^{-m}(-x\ sen(\theta) + y\ cos(\theta)) \tag{2.28}$$

Estos parámetros tiene como restricción:

- $a > 1$
- $\theta = \frac{n\pi}{K}$
- $n = 0, 1, ..., K - 1$
- $m = 0, 1, ..., S - 1$

Donde K y S son el número de orientaciones y escalas de filtros a generar.

Finalmente la aplicación de los filtros sobre la imagen es descrita por la ecuación 2.29 que corresponde a la operación convolución de un filtro con una imagen.

$$G_{mn}(x, y) = \sum_s \sum_t I(x - s, y - t) g^*_{mn}(s, t) \tag{2.29}$$

La aplicación de este filtro para un kernel de tamaño 31×31, longitud de onda 7 píxeles y 3 orientaciones, son presentadas en la figura 2.22.

(a) Imagen original. (b) Imagen filtrada. (c) Imagen filtrada.

(d) Imagen filtrada.

Figura 2.22: Aplicación de filtros Gabor.

2.4.5. Técnicas adicionales para la vectorización de características

Aparte de utilizar los coeficientes que arrojan los métodos que recurren a transformaciones de la imagen o elementos de matriz como GLCM, existen técnicas para continuar extrayendo información. Para los descriptores mencionados anteriormente, se mencionan las siguientes.

2.4.5.1. Transformada Wavelet y filtros Gabor

Las características para describir imágenes por medio de la transformada Wavelet o filtros Gabor corresponden a la extracción de componentes de media de la energía y desviación estándar de las ecuaciones 2.31 y 2.32 respectivamente, estos son aplicados sobre los coefi-

cientes Wavelet o Gabor resultantes de la imagen. Sin embargo, para algunas aplicaciones, los coeficiente Wavelet Gabor pueden ser utilizados para formar los vectores de características.

$$\mu = \sum_{i}^{n} |C_i| \tag{2.30}$$

$$Energia = \frac{1}{MN} \cdot \mu \tag{2.31}$$

$$Desviacion\ estandar = \frac{1}{MN} \cdot \sqrt{(Energia - \mu)^2} \tag{2.32}$$

2.4.5.2. GLCM

A partir del vector de características se extrae de la matriz: energía, contraste, correlación, disimilaridad y homogeneidad.

$$Energía = \sqrt{\sum_{i,j=0}^{N-1} P_{i,j}^2} \tag{2.33}$$

$$Contraste = \sum_{i,j=0}^{N-1} P_{i,j}(i-j)^2 \tag{2.34}$$

$$Correlación = \sum_{i,j=0}^{N-1} P_{i,j} \left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right] \tag{2.35}$$

$$Disimilaridad = \sum_{i,j=0}^{N-1} P_{i,j}|i-j| \tag{2.36}$$

$$Homogeneidad = \sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1+(i-j)^2} \tag{2.37}$$

Donde N es el nivel de gris máximo de la imagen, por defecto 255, $p_{i,j}$ es la componente (i, j) de la matriz de co-ocurrencia (GLCM), μ_i σ_i^2 media y varianza de filas de GLCM; y finalmente μ_j σ_j media y varianza de columnas de GLCM.

Capítulo 3

Metodología de evaluación

En este capítulo se narrarán los procedimientos que se usaron para evaluar el sistema CBIR para detección de plagio. El capítulo parte explicando como se obtuvo una base de datos inicial para los primeros funcionamientos reales del sistema en DOCODE y el procedimiento para realizar pruebas con una fracción de esta base de datos en conjunto con la creación del set de imágenes de consulta que corresponden a simulaciones de imágenes plagiadas.

Para finalizar el capítulo se describirá características que realiza el programa automáticamente al ingresar un documento en formato PDF en él.

3.1. Obtención de la base de datos

El juego de datos utilizado para el funcionamiento y pruebas del sistema consisten en imágenes extraídas de la aplicación libre y de código abierto *XowA* [41]. El proyecto permite, entre otras funcionalidades, explorar las páginas de Wikipedia español y contenido de *Wikimedia Commons* sin conexión a internet. El proyecto *XowA* permite extraer sus imágenes en un formato cuyo tamaño no es mayor al mostrado en las visualizaciones de sus artículos.

La cantidad de elementos de esta base de datos consta de 1.198.736 imágenes de diverso contenido y contexto. La figura 3.1 muestra un set de ejemplos de imágenes que se alberga.

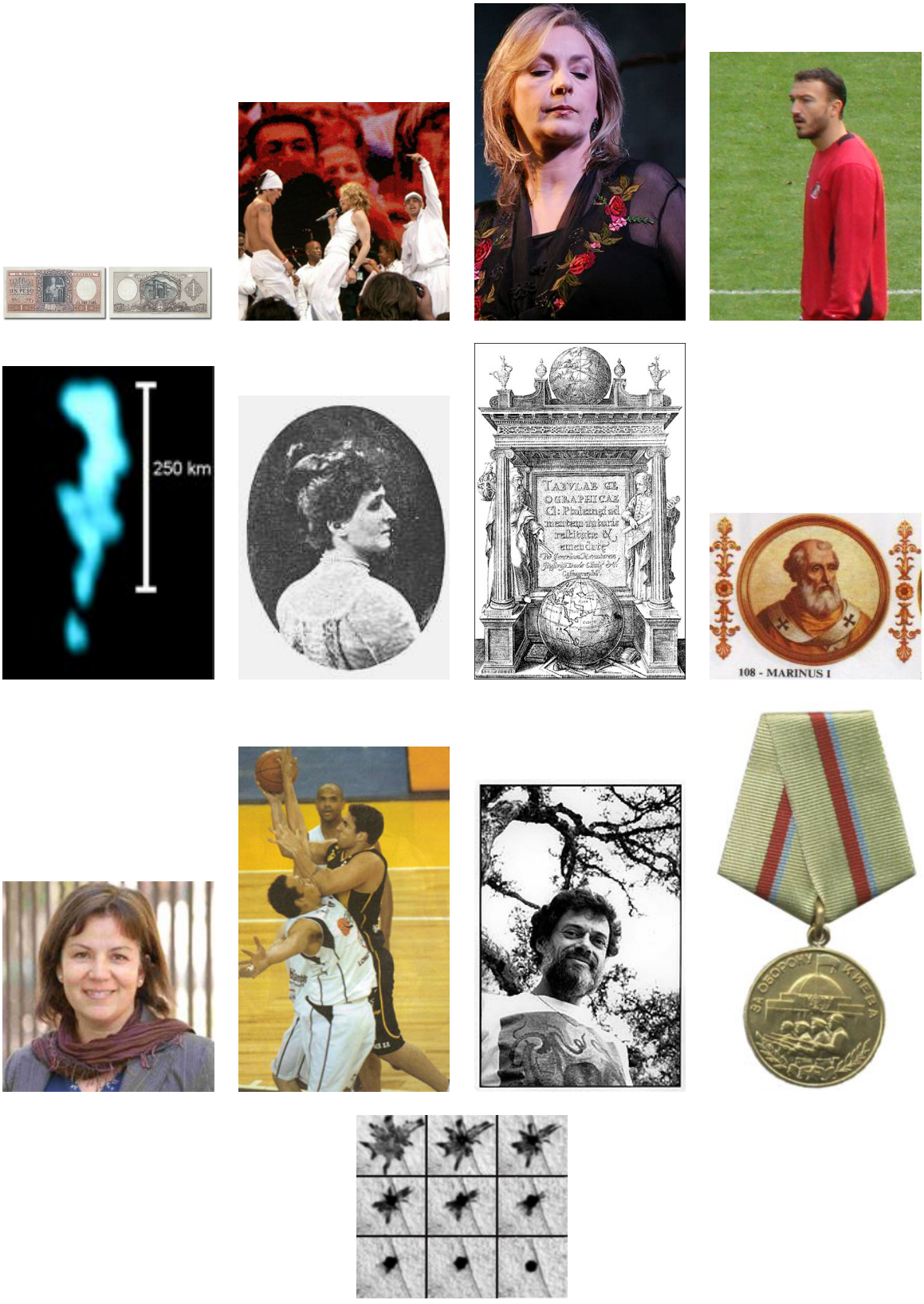


Figura 3.1: Ejemplo de imágenes de Wikipedia español.

3.2. Creación de imágenes de consulta

Para evaluar un sistema búsqueda de plagio se requiere crear un set de imagenes que simulen ser plagiadas, este set de consulta al sistema se generara a partir de adulteraciones de las imágenes originales de la base de datos. Las transformaciones a realizar corresponden a: recortes y rotaciones para aislar el área de interés, ajuste de color y brillo [18]. Estas transformaciones son posibles de incluir en un trabajo ajeno de su autor de origen con su correspondiente citado y con el contenido que deseaba mostrar la original de la imagen sin adulterar.

La creación de la base de datos por medio de CBIR utiliza generalmente la extracción de características de bajo nivel como: color, textura y forma. Estas características de bajo nivel corresponden a medidas capaces de describir el contenido de la imagen de forma general no así como lo hacen en cambio las características de alto nivel que son utilizadas para visión y reconocimiento de objetos tales como detección facial, huellas dactilares. Por último son inexistentes los sistemas que utilizan las características de alto nivel para realizar búsquedas en grandes bases de datos con imágenes de contenido general [23], por lo que se dará énfasis a los colores y texturas.

Para la creación de una imagen que simule ser plagiada, se considerará necesario someter la imagen: a una, todas o combinaciones de algunas de las transformaciones mencionadas en [42, 43, 9].

La descripción y proceso de obtención de cada una de las transformaciones realizadas a las imágenes son las siguientes:

Recorte
: A partir de cada lado de una imagen, se recortaron estos a usando como parámetro la extracción de una muestra de distribución gaussiana con media el largo a cortar y desviación un quinto de la media, el resultado final corresponde a una línea que es desplazada por el largo restante según una distribución uniforme.

Cambio de iluminación
: Se convierte la imagen a espacio de color HSV utilizando las ecuaciones 2.4, luego la componente V, que corresponde al nivel de iluminación, es sometido a un cambio en el valor de su píxel utilizando un ruido gaussiano de media 0 y desviación 100 para finalmente volver al espacio RGB.

Rotaciones
: Consiste en rotar la imagen usando como eje de rotación el centro de la imagen, esta transformación se realiza sin modificar el tamaño original de la imagen, por lo tanto genera regiones vacías.

Rotación con ajuste de márgenes
: Las rotaciones en imágenes son calculadas usando de la matriz de rotación definida en

la ecuación 3.1, sin embargo esta matriz rota con respecto al origen, es decir, la esquina superior izquierda, sin embargo gracias a transformaciones isométricas de cambio de origen es posible modificar el punto de rotación.

$$M = \begin{pmatrix} cos(\theta) & -sen(\theta) \\ sen(\theta) & cos(\theta) \end{pmatrix} \tag{3.1}$$

Luego de realizar rotación, la imagen es recortada para eliminar zonas negras producto de la salida de los margenes de la imagen original, este recorte se efectúa maximizando el área del rectángulo inscrito en la imagen rotada como se aprecia en la figura 3.2.

(a) Imagen original. (b) Imagen rotada con zonas negras. (c) Imagen final.

Figura 3.2: Secuencia de obtención de imágenes rotadas.

Ajuste de color

Se obtiene al variar el espacio de color RGB por cada canal de color de acuerdo a una distribución uniforme entre los valores -10 y 10 que es sumado al valor original del pixel.

La figura 3.3 resume, en un set imágenes, todas las transformaciones descritas anteriormente aplicadas sobre una imagen de prueba 3.3(a).

La generación del conjunto de imágenes de consulta consistió en crear plagio, como lo mencionado en el párrafo anterior, un total de 10 veces a cada imagen de la base de datos del sistema CBIR. De esta formal se evita no crear imágenes que, por efecto del azar de las transformaciones, sean más fáciles de detectar que otras.

(a) Imagen original.

(b) Recorte.

(c) Cambio de brillo.

(d) Rotación.

(e) Rotación y recorte de ajuste de borde.

(f) Ajuste de color.

(g) Imagen con todas las transformaciones aplicadas (sin (d)).

Figura 3.3: Ejemplo de transformaciones de imágenes plagiadas.

3.3. Evaluación de descriptores

Se usará para evaluar los descriptores la medida de desempeño para sistemas CBIR recall(10), para tener una medida igual a todos los resultados, además la extracción de 10 imágenes como resultado de búsqueda brinda un número razonable de resultados a revisar por el usuario. Los resultados de R(10) arrojados por cada descriptor corresponderán a bases de datos de consulta donde solo una transformación será efectuada.

Una vez teniendo todos los resultados para los descriptores, se escogerán los que presenten mejores resultados de desempeño en cada una de las transformaciones por individual, en caso que ningún descriptor sea capaz de retornar resultados satisfactorios para todos los casos evaluados se optará por estrategias de vectores de características híbridas compensar los puntos débiles.

3.4. Extracción de referencias usando *CERMINE*

La extracción de citas se realizará usando un sistema para la extracción estructurada de metadatos de código abierto llamado CERMINE [44], en particular los metadatos pertenecientes a la sección de referencias.

CERMINE fue diseñado para extracción de metadatos en artículos científicos en formato PDF, entre los metadatos que busca el sistema son: título, autor, journal, volumen, resumen, año de publicación, referencias, entre otros. Sin embargo CERMINE trabaja de forma separada los metadatos correspondientes a referencias y los demás, por lo que se espera que los documentos, de cualquier tipo, que contengan una sección de referencias, CERMINE sea capaz de extraerlos.

La implementación de este sistema de extracción de datos usa en la mayoría de sus pasos técnicas de *Machine learning* supervisadas y no supervisadas. En esta sección se limitará a explicar solo la extracción de referencias que corresponde a una técnica de *Machine learning* no supervisada.

El sistema determina la ubicación de la sección de referencias donde el texto restante se somete a un proceso de clasificación para dividir la zona referencias en referencias individuales. *clustering* usando el algoritmo de *KMeans* con distancia Euclideana y con número de *clústers* igual a 2 donde un clúster indica las líneas donde empiezan las referencias. Para obtener las características del texto, el algoritmo extrae por cada línea la siguientes información:

- Largo relativo de línea.
- Línea de indentación.
- Espacio entre la línea analizada y la anterior.
- Contenido de texto como por ejemplo si la línea empieza con un patrón de enumeración o si la línea previa termina con un punto.

CERMINE puede extraer metadatos a partir de referencias como: autor, título, página, año y más. De este resultado se buscan las referencias que podría contener un link o dirección web a una imagen citada en el documento .

3.5. Funcionamiento del programa buscador final

El punto de partida serán documentos en formato PDF, de estos se extraerán las imágenes por medio de dos técnicas: CERMINE y secuencias de *bytes* de comienzo y termino de archivos JPG. Las imágenes extraídas ingresarán al sistema de búsqueda extrayendo las características y encontrando su imagen de fuente a partir de una distancia mínima en el espacio de características formado por la base de datos de imágenes, si es que existe. Por

último, esta base de datos fue diseñada para seguir agregando nuevas imágenes con archivos que guardan sus características de manera desacoplada.

Para referencias que contengan una dirección *url* válida detectada, el sistema descargará las imágenes que provengan de aquella dirección, y se ingresarán temporalmente a la base de datos para realizar las búsquedas.

Finalmente, las imágenes encontradas contendrán una medida de similitud entregada por la distancia calculada en el espacio de características. El funcionamiento se presenta explicado a través de un diagrama de flujo de la figura 3.4.

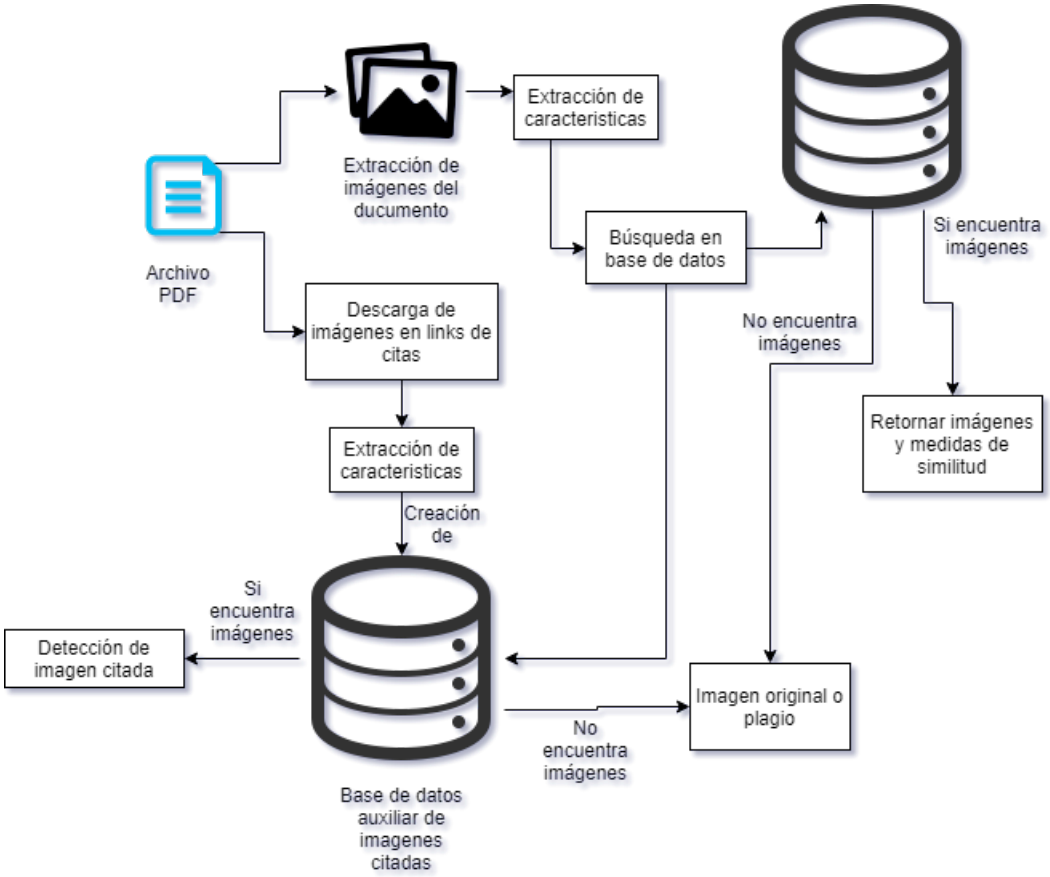


Figura 3.4: Diagrama de flujo del funcionamiento del sistema final de búsqueda.

Capítulo 4

Pruebas y resultados

En esta sección se mostrarán los resultados de diversas técnicas de extracción de características usando una base de datos en común, estas bases de datos consisten en sets generados automáticamente para cada transformación de la sección 3.2 con parámetros al azar, el resultado consiste en 10 imágenes transformadas para la simulación de plagio para cada una de las 250 imágenes iniciales de la base extraída de *Wikipedia* español (proyecto *XowA*), haciendo un total de 2500 imágenes para las pruebas, cada una de estas se configura para poseer una solo tipo de transformación para la evaluación. Los resultados serán evaluados usando recall(10) y presentados en tablas, así mismo, se presentarán tablas con los valores de tiempo de extracción de características para la base de datos inicial de 250 imágenes (creación de índices) y el tiempo de extracción de características con la búsqueda de cada una de las 2500 imágenes de consulta en la base de datos para cada caso de transformación a evaluar. Los datos de media y desviación estándar de tiempos fueron obtenidos usando sin considerar el tiempo en creación del índice.

El capítulo comienza probando las características: histogramas de color, transformada Wavelet, LBP, firma de imagen, GLCM, filtros Gabor. Luego se probarán características híbridas de las características anteriormente mencionadas con una base de datos de transformaciones de plagio simultáneas. El capitulo finaliza con algunas técnicas de pre-procesqamiento para la transformada Wavelet y sus pruebas en base de datos con transformaciones de plagio simultáneas.

4.1. Histogramas de color

Las siguientes pruebas consisten en la extracción de características de histogramas de colores usando el espacio de color HSV.

Como prueba preliminar, se evaluó el rendimiento para una base de datos de tamaño 3771 imágenes sin transformaciones, dando como resultado un recall igual 0.93. A partir de este

resultado se presenta la tabla 4.1 donde se detalla el recall resultante por fila al someter la base de datos a solo una transformación para simular plagio.

	Recall(10)
Rotación	0.777
Rotación + recorte	0.814
Recorte	0.815
Cambio de brillo	0.679
Ajuste de color	0.474

Tabla 4.1: Recall de histogramas de color para distintas transformaciones de imágenes

	Tiempo [s]
Índice	0,315
Rotación	4,719
Rotación + recorte	3,386
Recorte	3,788
Cambio de brillo	4,59
Ajuste de color	4,739
Media	4,2444
Desviación estándar	0,619

Tabla 4.2: Tiempo de ejecución para extracción de características de histogramas de color

4.2. Transformada discreta Wavelet 2D

La extracción de características se realiza dividiendo la imagen en regiones de 8×8, obteniendo así 64 sub imagenes y aumentar la dimensión del espacio de características formado por este método.

La aplicación de Wavelet 2D se realiza a cada una de estas 64 sub imágenes con un solo nivel de profundidad para obtener 4 imágenes correspondientes a la imagen de detalle y las 3 imágenes de aproximación, finalmente el vector de características se construye calculando la energía y desviación de estas 256 imágenes generadas. La tabla 4.3 muestra el recall obtenido al realizar búsquedas en una base de datos de 250 imágenes sujeto a distintos tipos de transformaciones.

	Recall(10)
Rotación	0.615
Rotación + recorte	0.479
Recorte	0.690
Cambio de brillo	0.801
Ajuste de color	1.000

Tabla 4.3: Recall para extracción de características usando Wavelet 2D con distintas transformaciones.

	Tiempo [s]
Índice	5,051
Rotación	49,728
Rotación + recorte	45,809
Recorte	47,872
Cambio de brillo	53,063
Ajuste de color	47,27
Media	48,7484
Desviación estándar	2,791

Tabla 4.4: Tiempo de ejecución para extracción de características de transformada Wavlet discreta 2D.

4.3. Local Binary Pattern

Esta prueba se realizó usando $LBP_{8,1}$, donde las características fueron directamente tomadas de los valores de la imagen LBP resultante redimensionadas a 50×50 pixeles. Resultados en tabla 4.5.

	Recall(10)
Rotación	0.678
Rotación + recorte	0.573
Recorte	0.739
Cambio de brillo	0.726
Ajuste de color	0.994

Tabla 4.5: Recall para características LBP usando la distancia euclideana para la búsqueda en base de datos.

	Tiempo [s]
Índice	0,618
Rotación	7,296
Rotación + recorte	6,646
Recorte	6,745
Cambio de brillo	7,347
Ajuste de color	7,297
Media	7,0662
Desviación estándar	0,341

Tabla 4.6: Tiempo de ejecución para extracción de características de Local binary Pattern.

La tabla 4.7 muestra una segunda prueba LBP que consistió en LBP *ROR* con extracción de características por cuadrillas de 4×4 de media y desviación estándar de cada una.

	Recall(10)
Rotación	0.571
Rotación + recorte	0.461
Recorte	0.653
Cambio de brillo	0.64
Ajuste de color	0.976

Tabla 4.7: Recall para características LBP *ROR* usando la distancia euclideana para la búsqueda en base de datos.

	Tiempo [s]
Índice	0,618
Rotación	7,396
Rotación + recorte	6,543
Recorte	6,794
Cambio de brillo	7,145
Ajuste de color	7,346
Media	7,0448
Desviación estándar	0,367

Tabla 4.8: Tiempo de ejecución para extracción de características de Local binary pattern ROR.

4.4. Firma de imagen

Las pruebas fueron realizadas utilizando dos tipos de distancias. Las tablas 4.9 y 4.10 muestran los resultados para distancias Manhattan y euclidiana respectivamente.

	Recall(10)
Rotación	0.751
Rotación + recorte	0.558
Recorte	0.742
Cambio de brillo	0.962
Ajuste de color	1.000

Tabla 4.9: Recall para extracción de características usando firma de imagen con distintas transformaciones y distancia Manahattan.

	Tiempo [s]
Índice	1,023
Rotación	11,475
Rotación + recorte	9,622
Recorte	9,97
Cambio de brillo	10,921
Ajuste de color	11,11
Media	10,6196
Desviación estándar	0,787

Tabla 4.10: Tiempo de ejecución para extracción de características de firma de imagen con distancia euclidiana.

usando distancia euclideana empeora los resultados de rotación y recortes

	Recall(10)
Rotación	0.660
Rotación + recorte	0.440
Recorte	0.630
Cambio de brillo	0.959
Ajuste de color	1.000

Tabla 4.11: Recall para firma de imagen usando la distancia euclideana para la búsqueda en base de datos.

	Tiempo [s]
Índice	0,985
Rotación	11,989
Rotación + recorte	9,765
Recorte	10,12
Cambio de brillo	11,371
Ajuste de color	11,222
Media	10,8934
Desviación estándar	0,923

Tabla 4.12: Tiempo de ejecución para extracción de características de firma de imagen con distancia Manhattan.

4.5. Matriz de co-ocurrencia de niveles de grises

La matriz generada para las pruebas de GLCM fue calculada usando como parámetros de diseño una distancia entre pixeles de 1 y 4 ángulos: $0°$, $45°$, $90°$ y $135°$ Los resultados de recall para una extracción de características usando la disimilaridad de la matriz GLCM se muestra en la tabla 4.13.

	Recall(10)
Rotación	0.377
Rotación + recorte	0.422
Recorte	0.794
Cambio de brillo	0.877
Ajuste de color	1.000

Tabla 4.13: Recall de GLCM usando disimilitud.

	Tiempo [s]
Índice	1,071
Rotación	11,876
Rotación + recorte	10,35
Recorte	10,852
Cambio de brillo	11,603
Ajuste de color	11,838
Media	11,3038
Desviación estándar	0,674

Tabla 4.14: Tiempo de ejecución para extracción de características de GLCM con extracción de disimilitud.

La tabla 4.15 muestra el recall usando correlación en GLCM para extracción de características.

	Recall(10)
Rotación	0.242
Rotación + recorte	0.474
Recorte	0.819
Cambio de brillo	0.799
Ajuste de color	1.000

Tabla 4.15: Recall de GLCM usando correlación como extracción de características.

	Tiempo [s]
Índice	1,61
Rotación	16,898
Rotación + recorte	15,293
Recorte	15,845
Cambio de brillo	16,597
Ajuste de color	16,847
Media	16,296
Desviación estándar	0,701

Tabla 4.16: Tiempo de ejecución para extracción de características de GLCM con extracción de correlación.

Finalmente se realizó una prueba donde se utilizaron ambas extracciones de características de la matriz GLCM presentadas en las tablas 4.13 y 4.15 concatenando el vector resultante.

	Recall(10)
Rotación	0.576
Rotación + recorte	0.614
Recorte	0.888
Cambio de brillo	0.773
Ajuste de color	1.000

Tabla 4.17: Recall de GLCM usando disimilaridad y correlación como extracción de características.

	Tiempo [s]
Índice	1,825
Rotación	19,102
Rotación + recorte	17,498
Recorte	18,099
Cambio de brillo	18,852
Ajuste de color	19,203
Media	18,5508
Desviación estándar	0,730

Tabla 4.18: Tiempo de ejecución para extracción de características de GLCM con extracción de disimilitud y correlación.

4.6. Filtros Gabor

Para las pruebas de filtros Gabor se configuró el sistema para generar filtros de tamaño 31×31, 4 orientaciones de $0°,\ 45°,\ 90°\ y\ 135°$ y 2 longitudes de onda de 7 y 11 pixeles. Los resultados con estos parámetros son presentados en la tabla 4.19.

	Recall(10)
Rotación	0.873
Rotación + recorte	0.812
Recorte	0.929
Cambio de brillo	0.859
Ajuste de color	0.972

Tabla 4.19: Recall para filtros Gabor.

	Tiempo [s]
Índice	4,91
Rotación	50,484
Rotación + recorte	33,047
Recorte	33,774
Cambio de brillo	50,985
Ajuste de color	54,129
Media	44,4838
Desviación estándar	10,208

Tabla 4.20: Tiempo de ejecución para extracción de características de Filtros Gabor de tamaño 31×31.

La tabla 4.21 muestra los resultados al expandir el tamaño de los filtros a 60×60.

	Recall(10)
Rotación	0.868
Rotación + recorte	0.793
Recorte	0.924
Cambio de brillo	0.857
Ajuste de color	0.978

Tabla 4.21: Recall de filtros Gabor para un tamaño de filtro expandido a 60×60.

	Tiempo [s]
Índice	6,65
Rotación	69,681
Rotación + recorte	52,134
Recorte	58,379
Cambio de brillo	74,207
Ajuste de color	74,059
Media	65,692
Desviación estándar	9,946

Tabla 4.22: Tiempo de ejecución para extracción de características de Fitros Gabor de tamaño 60×60.

Pruebas con aumento en el número de longitudes y/u orientaciones de onda son descartadas, puesto que generan una disminución en el rendimiento de la extracción de características, aumentando el tiempo de extracción de estas de 50 segundos a 2 minutos cuando se cambian los valores de longitudes de ondas de 2 a 5 y los valores de orientaciones de 4 a 10.

Una disminución en el número de orientaciones de 4 a 2, es decir 0 y 90 grados, produce un rendimiento descrito en la tabla 4.23.

	Recall(10)
Rotación	0.802
Rotación + recorte	0.779
Recorte	0.908
Cambio de brillo	0.850
Ajuste de color	0.969

Tabla 4.23: Recall de filtros Gabor para una disminución de orientaciones de 4 a 2.

	Tiempo [s]
Índice	2,578
Rotación	27,159
Rotación + recorte	17,498
Recorte	20,159
Cambio de brillo	26,91
Ajuste de color	27,109
Media	23,767
Desviación estándar	4,606

Tabla 4.24: Tiempo de ejecución para extracción de características de Filtros Gabor con orientaciones sumadas.

4.7. Características híbridas

Las siguientes pruebas consisten en el uso de características híbridas con el objetivo que crear un clasificador que sea capaz de encontrar imágenes sometidas a todas las transformaciones descritas, los descriptores usados para esta sección son los siguientes: histogramas de colores, filtros Gabor y GLCM. Cada uno de estos descriptores se utiliza de manera independiente hasta el momento donde se calculan la distancias a la base de datos, estas distancias son usadas para crear una combinación lineal entre ellas con coeficientes o pesos ajustables. Como primera prueba se evaluó el rendimiento de diferentes distancias para pesos iguales en los tres descriptores, la tabla 4.25 detalla los resultados usando recall(10) para la base de datos usada en la sección anterior.

Distancia	**R(10)**
Chebyshev	0.4608
Sokal-Michener	0.4872
Canberra	0.5864
Sørensen–Dice	0.4784
Euclideana	0.714
Bray-Curtis	0.6908
Manhattan	0.708
Jaccard	0.4784
Sokal-Sneath	0.4784
L_1	0.708
L_∞	0.4608

Tabla 4.25: Recall(10) para distintas distancias y pesos iguales para cada uno de los descriptores.

Se decide usar distancia euclideana para el resto de pruebas. Para buscar el valor adecuado de pesos que debe tener la combinación lineal de cada una de las distancias a la base de datos

por descriptor,en la figura 4.1 se realiza una búsqueda por fuerza bruta, donde se observan como cambian las medidas de recall para distintos valores de pesos.

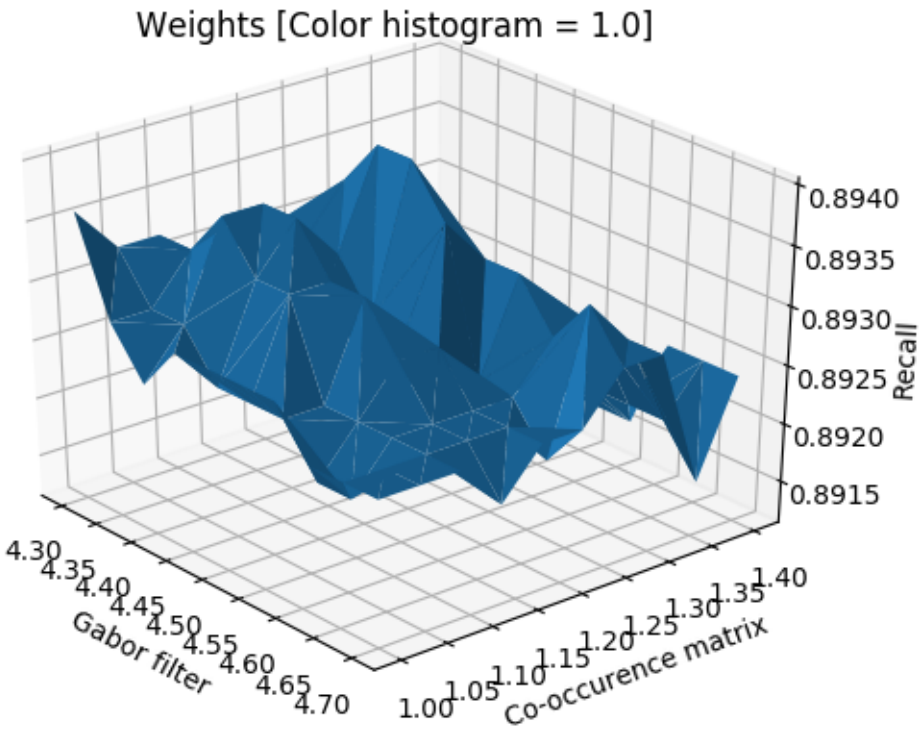


Figura 4.1: Recall 10 para diferentes combinaciones de pesos para características de color y textura, se mantuvo constante el valor del peso para las características de color para explorar solo combinaciones de pesos en texturas (Filtros Gabor y GLCM) en ejes x e y.

A partir de la figura 4.1 se concluye que los pesos óptimos para el buen funcionamiento del sistema corresponden a los valores mostrados en la tabla 4.26.

Características	**Pesos**
Histogramas H-S	1.00
Filtros Gabor	4.40
GLCM	1.20

Tabla 4.26: Pesos óptimos de la combinación lineal para cada característica.

Se presenta una prueba al sistema con los pesos configurados según la tabla 4.26, de modo de comparación, se mostrarán en la tabla 4.27 los resultados de recall(10) para cada una de las características por separado del sistema híbrido para un base de datos sometida

a transformaciones de rotación con recorte, recorte adicional y cambio de brillo de forma simultanea para las imágenes de consulta de simulación de plagio.

Característica	**Recall(10)**
histograma H-S	0.73
Filtros Gabor	0.71
GLCM	0.47
Combinación lineal	0.90

Tabla 4.27: Desempeño por cada método de extracción de característica para método de características híbridas sin ajuste de color.

Una prueba similar es realizada y presentada en la tabla 4.28 donde las imágenes de la base de datos fueron sometidas a las transformaciones de la prueba anterior pero con la transformación de ajuste de color adicional a estas y de forma simultanea para todas la imágenes generadas para consulta.

Característica	**Recall(10)**
histograma H-S	0.57
Filtros Gabor	0.71
GLCM	0.40
Combinación lineal	0.83

Tabla 4.28: Desempeño por cada método de extracción de característica para método de características híbridas con ajuste de color.

4.8. Pre-procesamiento de imágenes para transformada Wavelet

En esta sección se mostrarán los resultados obtenidos al aplicar los siguientes pre-procesamiento:

- Redimensionamiento a tamaño cuadrado de 60×60 pixeles.
- Implementación de mejoras para pre-procesamiento de imágenes descritas en las subsecciones: 4.8.1, 4.8.2, 4.8.3 y 4.9.
- División de las imágenes cuadradas en 4 sub-regiones de tamaño 30×30 pixeles.
- Extracción de características, por medio de coeficientes retornados de la transformada Wavelet 2D de cada sub-region.
- Concatenación de cada vector de características de sub-region para formar vector de características final.

La tabla 4.29 muestra los resultados obtenidos al aplicar los pasos descritos anteriormente sin considerar las mejoras del segundo punto.

	Recall(10)
Rotación	0.656
Rotación + recorte	0.624
Recorte	0.853
Cambio de brillo	0.967
Ajuste de color	1.000

Tabla 4.29: Resultados de recall para extracción de características con transformada Wavelet 2D y preprocesamiento de la imagen.

	Tiempo [s]
Índice	0,515
Rotación	5,316
Rotación + recorte	4,859
Recorte	5,258
Cambio de brillo	5,629
Ajuste de color	5,478
Media	5,308
Desviación estándar	0,290

Tabla 4.30: Tiempo de ejecución para extracción de características de transformada Wavelet 2D preprocesamiento de la imagen.

4.8.1. Ecualización de histogramas

	Recall(10)
Rotación	0.968
Rotación + recorte	0.256
Recorte	0.297
Cambio de brillo	0.898
Ajuste de color	1.000

Tabla 4.31: Resultados de recall para extracción de características con transformada Wavelet 2D y preprocesamiento de la imagen y ecualización de histogramas.

	Tiempo [s]
Índice	0,515
Rotación	5,597
Rotación + recorte	4,975
Recorte	4,945
Cambio de brillo	5,394
Ajuste de color	5,501
Media	5,2824
Desviación estándar	0,303

Tabla 4.32: Tiempo de ejecución para extracción de características de transformada Wavelet 2D con ecualización de histogramas.

4.8.2. Transformación Log-polar

	Recall(10)
Rotación	0.968
Rotación + recorte	0.256
Recorte	0.297
Cambio de brillo	0.898
Ajuste de color	1.000

Tabla 4.33: Resultados de recall para extracción de características con transformada Wavelet 2D y preprocesamiento de la imagen y transformación log-polar.

	Tiempo [s]
Índice	0,515
Rotación	5,466
Rotación + recorte	5,176
Recorte	5,286
Cambio de brillo	5,441
Ajuste de color	5,391
Media	5,352
Desviación estándar	0,120

Tabla 4.34: Tiempo de ejecución para extracción de características de transformada Wavelet 2D y transformación log-polar.

4.8.3. Transformación log-polar y ecualización de histogramas

	Recall(10)
Rotación	0.916
Rotación + recorte	0.674
Recorte	0.275
Cambio de brillo	0.969
Ajuste de color	0.981

Tabla 4.35: Resultados de recall para extracción de características con transformada Wavelet 2D y preprocesamiento de la imagen, transformación log-polar y ecualización de histogramas.

	Tiempo [s]
Índice	0,515
Rotación	5,481
Rotación + recorte	4,826
Recorte	5,408
Cambio de brillo	5,956
Ajuste de color	5,429
Media	5,420
Desviación estándar	0,401

Tabla 4.36: Tiempo de ejecución para extracción de características de transformada Wavelet 2D con ecualización de histogramas y transformación log-polar.

4.9. Combinación Wavelet 2D y Wavelet 2D con log-polar

Dado que las características extraídas por Wavelet 2D y Wavelet 2D con log-polar corresponden básicamente al mismo descriptor, no es necesario implementar un método de extracción de características híbridas como en la sección 4.7. Para combinar estas dos características, basta extraer 5 resultados de Wavelet 2D con log-polar y 10 resultados de Wavelet 2D sin log-polar, luego de ordenar los resultados según las distancias calculadas a la base de datos representan la misma distribución.

Para comprobar la correcta selección de características del sistema, se realizaron pruebas dondes las imágenes fueron sometidas a transformaciones simultáneas. La prueba nº1 consistió en transformar para cada imagen del set de pruebas 5 veces para rotaciones y 5 para recortes con el objetivo de probar si el sistema final es capaz de seleccionar Wavelet sin transformación log-polar para las imágenes con recortes y Wavelet con log-polar para imágenes con rotaciones.

La prueba 1 obtuvo un recall de 0.93. La prueba nº2 consideró la adición de las demás

transformaciones: brillo y color a todas las imágenes generadas, además se utilizó una nueva transformación: cambio de escala. Esta nueva transformación se aplicó nuevamente a todas las imágenes generadas anteriormente, finalmente este sistema obtuvo un recall de 0.87.

Se evaluó también el sistema para otras bases de datos con 196, 250 y 3771 imágenes que generan 1960, 2500 y 37710 imágenes para consulta. El recall del sistema para las transformaciones descritas en la prueba 1 fueron de 0.93, 0.92 y 0.77 respectivamente para cada base de datos. Y los resultados para una configuración idéntica a la segunda prueba fueron de 0.82, 0.87 y 0.66.

4.10. Adición de primera base de datos al sistema de búsqueda

Usando la base de datos de todas las 1.198.008 imágenes extraídas del proyecto *XoWa* en español, se realizaron pruebas de extracción de características y búsqueda para el sistema descrito en la sección 4.9, a partir de documentos en formato PDF, con la finalidad de comprobar el rendimiento en un escenario un poco más exigente y parecido al funcionamiento real que se espera que realice para la plataforma DOCODE.

La extracción de características de todo la base de datos se obtuvo en 38 minutos con 54 segundos, con respecto a la búsqueda de imágenes, se ingresó al programa un documento PDF con 51 imágenes de ejemplo (no todas las imágenes tenían copia original en la base de datos), de estas, el programa retorna las imágenes cuya distancia es menor a 0.1 (umbral ajustable) en el espacio de características.

Con estas condiciones se obtuvo la tabla 6.2 en la sección de anexos, esta tabla muestra el tiempo en retornar el set de imágenes que cumplen la distancia umbral del espacio de características para cada una de las 51 imágenes de ejemplo extraídas del documento PDF. El tiempo medio de segundos por iteraciones o imagen a buscar fue de 16,385 segundos con una desviación estándar de 1,228 segundos.

4.11. Resumen de resultados

Con el objetivo de comparar cada método de extracción de características de manera simple, se presentan las tablas 4.37, 4.38, 4.39 y 4.40 donde los resultados de recall(10) son coloreados para obtener una mejor compresión de las cualidades de cada método probado.

Transformaciones	histograma color	wavelet 2D	LBP	LBP ROR
Rotación	0,777	0,615	0,678	0,571
Rotación+recorte	0,814	0,479	0,573	0,461
Recorte	0,815	0,69	0,739	0,653
Cambio de Brillo	0,679	0,801	0,726	0,64
Ajuste de Color	0,474	1	0,994	0,976

Tabla 4.37: Resumen de resultados de recall. Parte 1.

Transformaciones	Firma manhattan	Firma euclidean	GLCM disimilaridad	GLCM correlación
Rotación	0,751	0,66	0,3768	0,242
Rotación+recorte	0,558	0,44	0,422	0,474
Recorte	0,742	0,63	0,794	0,819
Cambio de Brillo	0,962	0,959	0,877	0,799
Ajuste de Color	1	1	1	1

Tabla 4.38: Resumen de resultados de recall. Parte 2.

Transformaciones	GLCM disimilaridad + correlación	Gabor tamaño 31	Gabor tamaño 60	Gabor + orientaciones
Rotacion	0,576	0,873	0,868	0,802
Rotacion+recorte	0,614	0,812	0,793	0,779
Recorte	0,888	0,929	0,924	0,908
Cambio de Brillo	0,773	0,859	0,857	0,85
Ajuste de Color	1	0,972	0,978	0,969

Tabla 4.39: Resumen de resultados de recall. Parte 3.

Transformaciones	Wavelet 2D con preprocesamiento	wavelet 2D + ecualización de histogramas	wavelet 2D + log-polar	wavelet 2D + log-polar + ecualización de histograma
Rotacion	0,656	0,715	0,914	0,916
Rotacion+recorte	0,624	0,71	0,68	0,674
Recorte	0,853	0,902	0,266	0,275
Cambio de Brillo	0,967	0,993	0,923	0,969
Ajuste de Color	1	0,991	1	0,981

Tabla 4.40: Resumen de resultados de recall. Parte 4.

Descriptor	Segundos
histograma color	0,315
wavelet 2D	5,051
LBP	0,618
LBP ROR	0,618
Firma manhattan	1,023
Firma euclidean	0,985
GLCM disimilaridad	1,071
GLCM correlación	1,61
GLCM disimilaridad + correlación	1,825
Gabor tamaño 31	4,91
Gabor tamaño 60	6,65
Gabor + orientaciones	2,578
wavelet 2D + log-polar	4,851
wavelet 1D	0,716
Wavelet 1D + ecualización	0,715
Wavelet 1D + log-polar	0,766

Tabla 4.41: Tiempos de ejecución de cada descriptor para la generación de características de imágenes para la base de datos de 250 imágenes de prueba.

Los gráficos realizados a las tablas 4.41, 4.42 y 4.43 son presentados en anexos.

Descriptor	rotación [s]	Rotación+ recorte [s]	recorte [s]	brillo[s]
histograma color	4,719	3,386	3,788	4,59
wavelet 2D	49,728	45,809	47,872	53,063
LBP	7,296	6,646	6,745	7,347
LBP ROR	7,396	6,543	6,794	7,145
Firma manhattan	11,475	9,622	9,97	10,921
Firma euclidean	11,989	9,765	10,12	11,371
GLCM disimilaridad	11,876	10,35	10,852	11,603
GLCM correlación	16,898	15,293	15,845	16,597
GLCM disimilaridad + correlación	19,102	17,498	18,099	18,852
Gabor tamaño 31	50,484	33,047	33,774	50,985
Gabor tamaño 60	69,681	52,134	58,379	74,207
Gabor + orientaciones	27,159	17,498	20,159	26,91
wavelet 2D + log-polar	51,534	47,315	47,919	48,02
wavelet 1D	7,101	6,178	6,618	6,896
Wavelet 1D + ecualización	7,283	6,631	6,751	7,413
Wavelet 1D + log-polar	7,955	7,134	6,581	8,072

Tabla 4.42: Tiempos de ejecución de cada descriptor para la generación de características de las imágenes de consulta generadas a partir de las 5 transformaciones que conforman 2500 imágenes. Parte 1.

Descriptor	**color [s]**	**media [s]**	**Desviación estándar [s]**
histograma color	4,739	4,12075	0,63996842
wavelet 2D	47,27	49,118	3,07880507
LBP	7,297	7,0085	0,36426959
LBP ROR	7,346	6,9695	0,37656562
Firma manhattan	11,11	10,497	0,85237198
Firma euclidean	11,222	10,81125	1,04449043
GLCM disimilaridad	11,838	11,17025	0,69748327
GLCM correlación	16,847	16,15825	0,72721266
GLCM disimilaridad + correlación	19,203	18,38775	0,73045939
Gabor tamaño 31	54,129	42,0725	10,0085087
Gabor tamaño 60	74,059	63,60025	10,1359921
Gabor + orientaciones	27,109	22,9315	4,86175198
wavelet 2D + log-polar	51,537	48,697	1,91677733
wavelet 1D	6,981	6,69825	0,3993381
Wavelet 1D + ecualización	7,059	7,0195	0,38613426
Wavelet 1D + log-polar	7,585	7,4355	0,7061834

Tabla 4.43: Tiempos de ejecución de cada descriptor para la generación de características de las imágenes de consulta generadas a partir de las 5 transformaciones que conforman 2500 imágenes. Parte 2.

Capítulo 5

Discusión de resultados

En este capítulo se revisarán los resultados obtenidos de la sección anterior analizando las ventajas y desventajas de cada método de extracción de características y las decisiones tomadas durante la evolución del buscador final.

5.1. Características de color

El desempeño de la búsqueda de imágenes por medio de histogramas de color de la tabla 4.1, muestra resultados levemente satisfactorios para transformaciones de rotación con recorte y recortes puros, los cuales forman el conjunto de las únicas transformaciones que realizan un proceso de eliminación de información y que la mayoría de los métodos de extracción de características presentan resultados de recall menores a histogramas de color. La transformación de rotación retorna un desempeño más bajo que rotación con recorte por la presencia de zonas blancas en los lugares que la rotación de la imagen no cubre, estas zonas afectan a la forma del histograma, y por lo tanto a la búsqueda.

El bajo desempeño con respecto al cambio de brillo también es consecuencia del cambio de color, así como también la caída drástica de desempeño a menos de la mitad sujeto a imágenes con ajuste de color, hace de este descriptor una alternativa poco factible para la implementación final del buscador.

Finalmente este descriptor fue el único que no alcanzó recall(10) igual o cercano a 1.0 para imágenes de consulta sin transformaciones de simulación de plagio, este resultado es explicado por la existencia de imágenes en escala de grises como las presentadas en la figura 5.1.

Este descriptor presenta ventajas con respecto a las transformaciones de contienen recortes y el tiempo de extracción de características, es así que la adición de este descriptor al sistema de características híbridas ayuda a los descriptores de textura a remediar los problemas de

recorte y rotación sin perjudicar tiempo de ejecución durante creación de las imágenes del índice de la base de datos.

El set de imágenes de la figura 5.1 exhibe algunas de las imágenes en escala de grises no encontradas que presentó la búsqueda con histogramas de colores al no realizar ninguna transformación de imágenes en la base de datos.

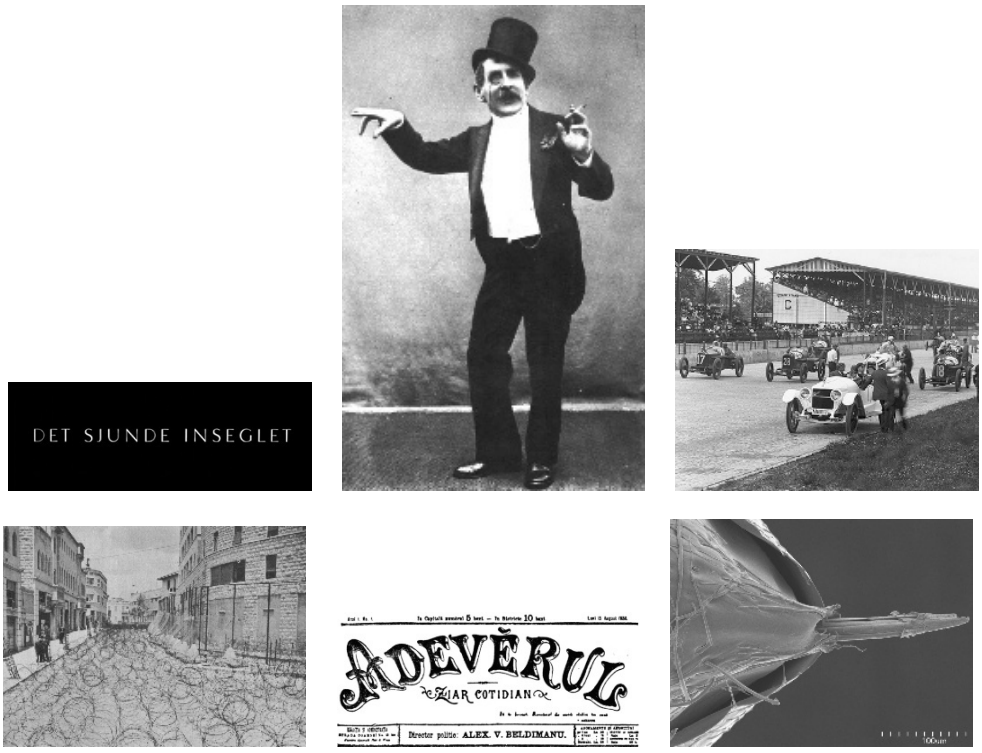


Figura 5.1: Imágenes en escala de grises que forman vectores de características similares en histogramas de colores.

5.2. Transformada Wavelet 2D

La primera implementación de Wavlet 2D se realizó extrayendo características correspondientes a parámetros estadísticos de los coeficientes resultantes de la transformación, la implementación de esta medida tuvo como objetivo la creación de vectores de características de dimensión constante con respecto al tamaño de la imagen y por la aplicación recurrente en la literatura en la búsqueda de CBIR.

La mayoría de las pruebas se pudieron haber hecho simplemente computando la transformada sobre imagen entera o en pequeñas regiones de ella, pero dado los resultados positivos se optó por dividir la imagen en regiones de 8×8, esto afecta directamente al rendimiento tanto como en extracción de características en un factor de 64 como en la búsqueda en la base de datos debido a la generación de un vector de dimensión 512.

5.3. LBP

Local binary pattern o LBP es ampliamente usado en CBIR por su invariabilidad al cambio de brillo, puesto que sus características dependen de la comparación del nivel del brillo entre los pixeles y sus vecinos de la imagen a analizar. Las pruebas señalan que este descriptor no es capaz de arrojar resultados satisfactorios para cambios en niveles de brillo, con lo que vemos que LBP no sirve para para ser implementado en el sistema.

La implementación de la mejora a rotaciones de LBP resultó ser contraproducente por le hecho que los resultados mostraron disminución en las transformaciones donde se supone que la mejora provoca cambios, estos son en las transformaciones sobre la imagen de rotaciones donde el recall en cada una decreció un 10 % aproximadamente con respecto a la aplicación de LBP directo de la prueba anterior.

Un aspecto positivo de este descriptor es el hecho que el tiempo de ejecución y el tamaño de vector de características propician las cualidades buscadas para la implementación del buscador final.

5.4. Firma de imagen

Firma de imagen es un descriptor diseñado especialmente para de búsqueda de imágenes exactamente idénticas, sin embargo sus resultados solo lograron un recall mayor al 90 % para las trasformaciones que no editaban las posiciones de los pixeles, es decir, cambios de brillo y color.

Finalmente el gran tamaño del vector de características, 648 dimensiones, otorga un gran retraso para búsqueda en base de datos grandes por medio del sistema *Ball Tree* que estipula en su documentación una baja en la efectividad de la búsqueda a partir de dimensiones de los vectores mayores a 30 elementos.

5.5. Grey Level Co-occurrence Matrix

Con respecto a GLCM, las ventajas de su uso fueron resultados positivos con respecto al recorte puro de la imagen, obteniendo resultados cercanos al 80 % descritos en las tablas 4.13 y 4.15, los cuales no eran vistos desde las pruebas con histogramas de colores. Además cuenta con número muy bajo de dimensiones para el vector de características para las optimizar las búsquedas realizadas en la base de datos final, las dimensiones de los vectores fueron: 4 para las pruebas realizadas extrayendo una sola característica de la matriz de co-ocurrencia de niveles de grises y 8 para la combinación de estas, por lo tanto se espera que la búsqueda en vectores en una base de datos grande sea manejado rápidamente por *Ball Tree*.

Con respecto a la combinación de disimilaridad con correlación en GLCM se evidenció que, si bien mejoró puntos como las rotaciones, estas no fueron suficientes como para optar usar este descriptor, sin embargo los recortes presentaron un recall superior a todos los métodos de extracción de características vistos hasta este punto.

5.6. Filtros Gabor

La implementación filtros Gabor es usualmente usado con un mayor número de parámetros, por lo menos con 4 valores de orientaciones y 5 de longitudes de onda para generar los filtros, lamentablemente al utilizar esta configuración se obtuvo tiempos de ejecución que superaban fácilmente el minuto para la base de datos de pruebas de 2500 imágenes, sin embargo al aplicar una reducción del número de longitudes de onda, los resultados no variaron drásticamente.

Básicamente, filtros Gabor obtuvo rendimientos aceptables para todo tipo de transformación en imágenes, tanto para casos de rotaciones como recortes donde supera todo tipo de descriptor estudiado hasta ahora.

Con respecto al cambio de tamaños en los filtros de 60×60 a 30×30, se observó que este cambio otorga aproximadamente alrededor de 20 segundos de reducción del tiempo de extracción de características según las tablas 4.20 y 4.22, lamentablemente, a pesar de los buenos resultados, esta mejora no bastó para ser determinante a la hora de establecer los descriptores finales por el pobre rendimiento de tiempo de procesamiento que requiere este descriptor.

La implementación de filtros capaces de extraer texturas por medio de simplificación del número de orientaciones, solo retornó como mejora una reducción del tiempo de ejecución con un pequeño descenso de recall para rotaciones, la cual sigue aún estando lejos para una implementación real.

5.7. Características Híbridas

La realización del sistema usando características híbridas consistió principalmente en encontrar una forma efectiva de búsqueda en los casos donde las imágenes presentaran transformaciones simultaneas, usar las ventajas o resultados de recall de ofrecen los filtros Gabor pero con una disminución del tiempo de extracción de características.

Si bien el sistema de características híbridas resultó funcionar satisfactoriamente alcanzando en recall de 90 % en la tabla 4.27, este solo fue posible descartando la transformación de color a las imágenes de prueba, esto es debido a la mal desempeño que presentan los histogramas de color frente a esta transformación donde su recall decrece a 57 % (tabla 4.28).

Desgraciadamente, el sistema híbrido no alcanza a ser utilizable para búsquedas en grandes bases por la ineficiencia en la extracción de características para filtros Gabor y el gran número de dimensionalidad que tiene que trabajar el sistema que básicamente es la suma de las características de los 3 descriptores.

5.8. Sistema Final de Búsqueda

Uno de los problemas que presentaban los sistemas de texturas como Wavelet, GLCM o filtros Gabor fue el bajo rendimiento frente a transformaciones de rotación de imágenes, sin embargo presentaban robustez al recorte, cuya transformación presentaba ser dominada por la extracción de texturas, finalmente los cambios de brillo y ajustes de color fueron abordados sin mayores problemas.

Con estos antecedentes, el sistema final de búsqueda presenta un sistema que utiliza todas las virtudes encontradas de la extracción de texturas en las imágenes evitando el bajo desempeño para rotaciones usando transformación log-polar. Este sistema demostró ser capaz de utilizar cualquiera de los métodos de extracción de características: Wavelet puro y Wavelet con transformación log-polar, además la dimensión del espacio de características es de 128, que comparado con los demás descriptores, es de tamaño medio, y por lo tanto, no dificulta la búsqueda.

El siguiente set de imágenes de la figura 5.2 muestra un pequeño grupo que el sistema no fue capaz de encontrar, es decir, son las imágenes pertenecientes al 0.13 del recall restante de la prueba nº2.

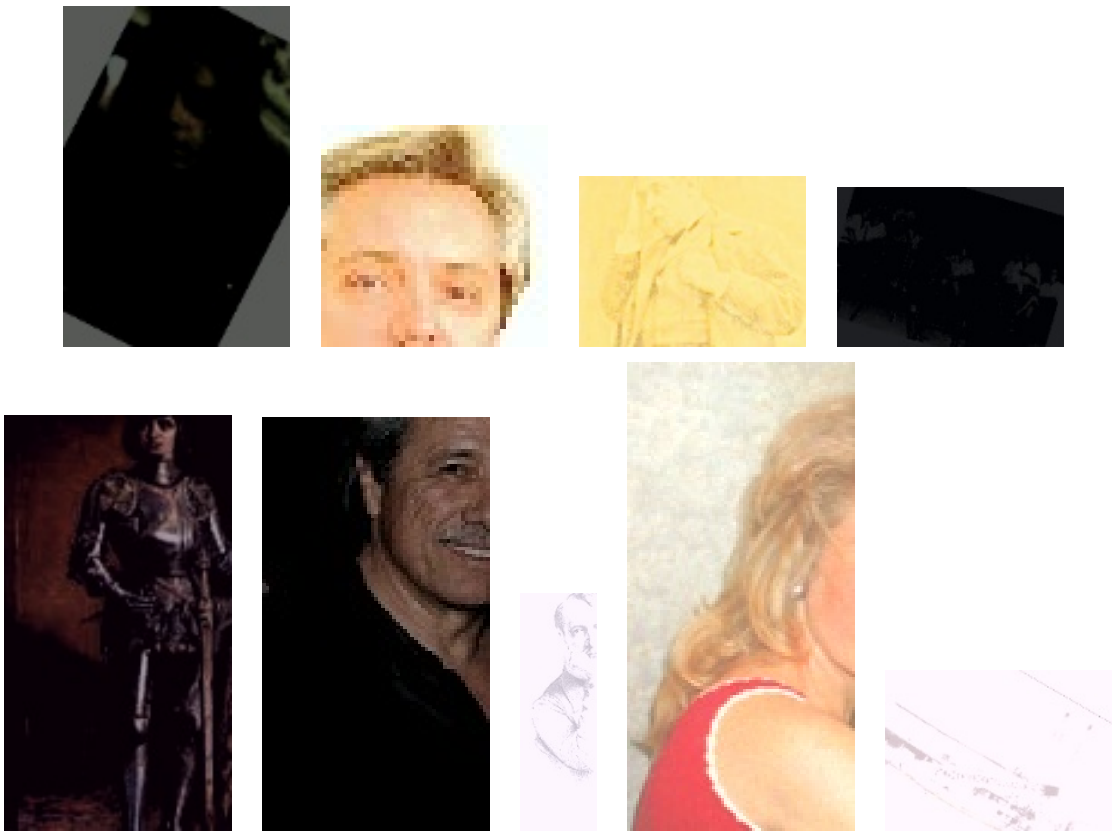


Figura 5.2: Imágenes no encontradas pertenecientes a la segunda prueba.

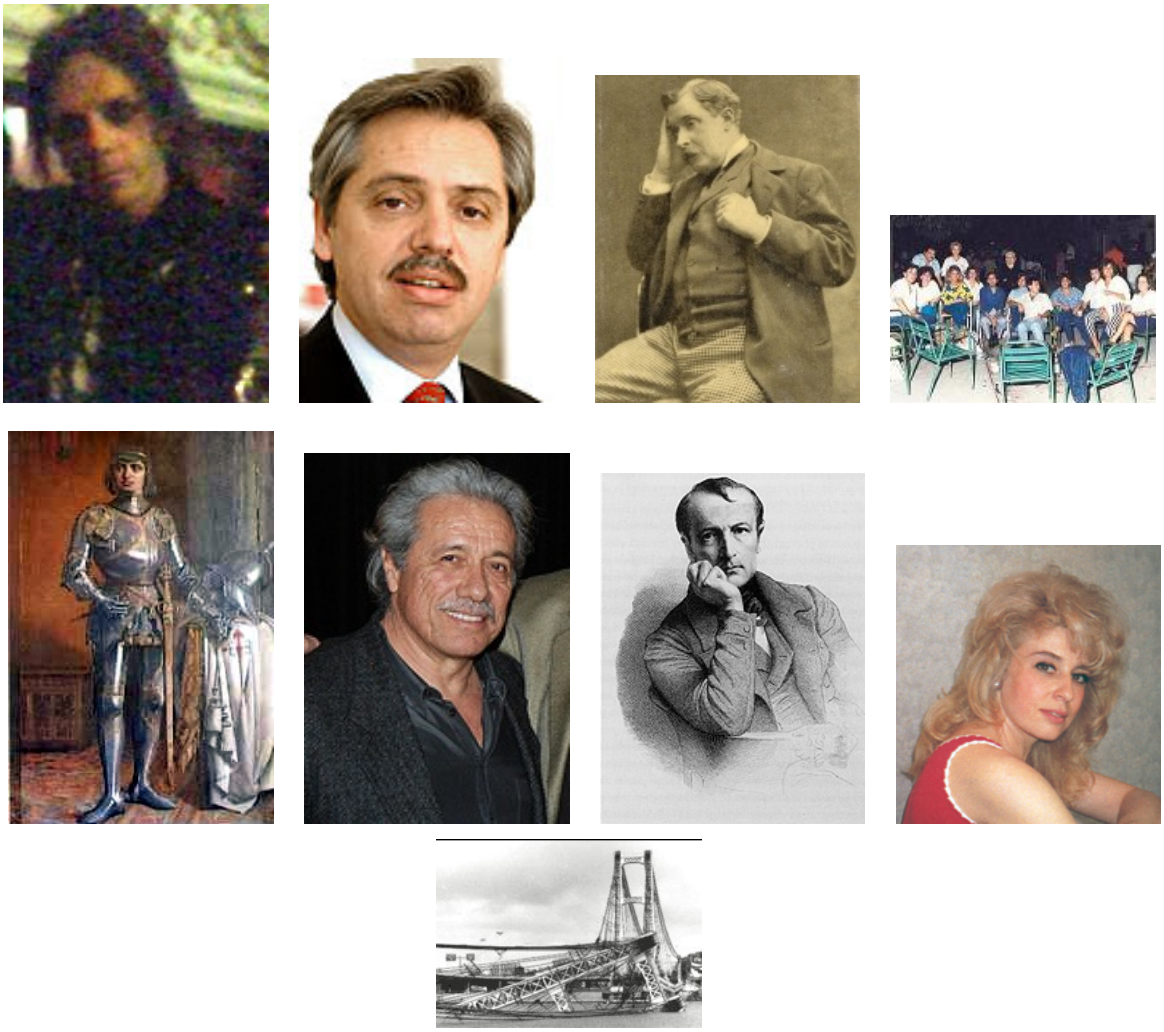


Figura 5.3: Imágenes originales de la figura 5.2.

5.9. Tiempo de ejecución

Para finalizar el capítulo, se presenta a continuación en la tabla 5.1 las dimensiones de los vectores de características generados para cada descriptor. La búsqueda en el espacio de caracteristicas realizadas pr el algoritmo de busqueda *Ball tree*

Descriptor	**Dimensión vector**
histograma color	512
wavelet 2D	512
LBP	32
LBP ROR	32
Firma manhattan	648
Firma euclidean	648
GLCM disimilaridad	4
GLCM correlación	4
GLCM disimilaridad + correlación	8
Gabor tamaño 31	64
Gabor tamaño 60	64
Gabor + orientaciones	32
wavelet 2D + log-polar	512
wavelet 1D	128
Wavelet 1D + ecualización	128
Wavelet 1D + log-polar	128

Tabla 5.1: Dimensión del vector de características para cada descritor.

Capítulo 6

Conclusiones

En esta memoria se estudió el problema de búsqueda de imágenes para la detección automática de imágenes plagiadas a través de ejemplo en documentos por medio del esquema CBIR para la plataforma DOCODE del *Web intelligence centre*.

La etapa correspondiente a la extracción de características fue estudiada a profundidad correspondiente la extracción de características, en ella se concluye que, a través de una amplia gama de métodos de extracción de características testeados y analizados, los descriptores que utilizan las texturas de las imágenes para extraer características, resultaron ser los mejores para desarrollar el sistema, entre ellos, los filtros Gabor y transformada Wavelet.

A pesar de los buenos resultados obtenidos con filtros Gabor, su implementación en el sistema final de búsqueda no es factible debido a lo poco optimizado que resulta su ejecución. Sin este problema, resultaría muy útil, puesto que a diferencia con el sistema final, este filtro presenta buen rendimiento al ejecutarlo en una versión "reducida", por lo que tiene un potencial aún mayor de lo mostrado en la sección 4.6 de resultados.

Los otros descriptores, aparte de Wavelet y filtros Gabor, no logran un desempeño suficiente en pruebas de transformaciones independientes de simulación de plagio, sin embargo con un pre-procesamiento adecuado como lo realizado con la extracción de texturas usando transformada Wavelet se pueda suplir algunas deficiencias.

El rendimiento del sistema se evaluó de forma exhaustiva para cada descriptor, a parir de las pruebas realizadas en la base de datos cuyo contenido eran todas las imágenes de *Wikipedia* en español del proyecto *XoWa*. Se concluye que el sistema final es capaz de gestionar, sin mayores problemas, una base de datos más grande, además el recurso creador del espacio de características permite una fácil ampliación por medio de la adición de otros volúmenes de imágenes.

6.1. Trabajo Futuro

A continuación se describirán los desarrollos que mejoran el trabajo de esta memoria.

- Indexar más imágenes para el sistema por medio del uso de un *Web Crawler*.
- Evaluar el sistema para un juego de imágenes mayor si se completa el punto anterior relacionado a la indexación por medio de un *Web Crawler*.
- Investigar el uso de *Deep Learning* para extracción de características. La disponibilidad de una GPU compatible con procesamiento paralelo CUDA para realizar los cómputos de la red y acelerar el proceso.
- Acelerar el proceso de filtado de imágenes por medio del uso de una GPU, así los filtros Gabor podrían llegar a ser factibles para una implementación real del sistema.
- Cambiar el buscador interno por medio de *Ball Tree* a uno externo como por ejemplo *elasticsearch*.
- Realizar pre-procesamiento a características no utilizadas en el sistema final, al igual como se hizo con Wavelet, para explorar otras opciones de implementación de búsqueda.

Publicación

El estudio de características híbridas para el sistema de búsqueda brindó la oportunidad de escribir una publicación que fue aceptada para la conferencia *Web Intelligence International Conference 2018* en calidad de *Regular Paper*. El trabajo se presenta en sección Anexos: Parte B.

Bibliografía

[1] J. V. S. Sepúlveda, G. Pizarro, "A plagiarism detection engine for images in docode," 2018.

[2] L. Anis, T. Ghada, S. Anis, and M. Abdellatif, "Optimal feature selection based on hybridization of msfla and gabor filters for enhanced mr brain image recognition using svm," *International Journal of Tomography & Simulation*, vol. 27, 2014.

[3] "Yale face database b." [Online]. Available: http://vision.ucsd.edu/~leekc/ExtYaleDatabase/Yale%20Face%20Database.htm

[4] K. Pradhan and G. Gohil, "Securing web communication using three layer image shielding," *ICDIT*, 2014.

[5] L. A. Fernández, D. Diaz, and R. Depaoli, "Optimización de la ecualización del histograma en el procesamiento de imágenes digitales," in *VII Workshop de Investigadores en Ciencias de la Computación*, 2005.

[6] W. Burger and M. Burge, *Digital Image Processing: An Algorithmic Introduction Using Java*, ser. Texts in Computer Science. Springer London, 2012. [Online]. Available: https://books.google.cl/books?id=jCEi9MVfxD8C

[7] C.-M. Pun and M.-C. Lee, "Log-polar wavelet energy signatures for rotation and scale invariant texture classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, pp. 590–603, 2003.

[8] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[9] H. C. Wong, M. Bern, and D. Goldberg, "An image signature for any kind of image," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1. IEEE, 2002, pp. I–I.

[10] A. Dertat. Applied deep learning - part 4: Convolutional neural networks. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[11] A. d. T. de Señales, "Introducción a la transformada wavelet," *Departamento de Señales y sistemas. Universidad de Navarra*, 2006.

[12] P. Jorgensen. (2006) Image decomposition using haar wavelet. [Online]. Available: http://homepage.divms.uiowa.edu/~jorgen/Haar.html

[13] "We now upload and share over 1.8 billion photos each day: Meeker Internet report- Technology News, Firstpost." [Online]. Available: https://www.firstpost.com/tech/news-analysis/now-upload-share-1-8-billionphotos-everyday-meeker-report-3652169.html

[14] K. G. Coffman and A. M. Odlyzko, "The size and growth rate of the Internet," p. 25.

[15] *Diccionario de la Real Academia EspaÃola.* Libr. de Parmantier, 1826. [Online]. Available: https://books.google.cl/books?id=OanYapiW-OIC

[16] C. A. Gómez, "Plagio y derechos de autor," *El Foro*, no. 10, pp. 59–67, 2009.

[17] P. Ovhal, "Detecting plagiarism in images," in *Information Processing (ICIP), 2015 International Conference on.* IEEE, 2015, pp. 85–89.

[18] S. Srivastava, P. Mukherjee, and B. Lall, "implag: Detecting image plagiarism using hierarchical near duplicate retrieval," in *India Conference (INDICON), 2015 Annual IEEE.* IEEE, 2015, pp. 1–6.

[19] P. M. Ovhal and B. Phulpagar, "Plagiarized image detection system based on cbir," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 4, no. 3, 2015.

[20] "Quienes Somos – Web Intelligence Centre." [Online]. Available: http://wic.uchile.cl/?page_id=14

[21] "Docode." [Online]. Available: http://docode.cl

[22] T. Dharani and I. L. Aroquiaraj, "A survey on content based image retrieval," in *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on.* IEEE, 2013, pp. 485–490.

[23] J. M. Patel and N. C. Gamit, "A review on feature extraction techniques in content based image retrieval," in *Wireless Communications, Signal Processing and Networking (WiSPNET), International Conference on.* IEEE, 2016, pp. 2259–2263.

[24] S. Mohamed, I. I. U. M. D. of Library, and I. Science, *Evaluation of Content Based Image Retrieval (CBIR) System Using Precision Measure.* Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, 2015. [Online]. Available: https://books.google.cl/books?id=rMWAAQAACAAJ

[25] Y. A. Aslandogan, C. Thier, C. T. Yu, J. Zou, and N. Rishe, "Using semantic contents

and wordnet in image retrieval," in *ACM SIGIR Forum*, vol. 31, no. SI. ACM, 1997, pp. 286–295.

[26] F. Banfi, "Content-based image retrieval using hand-drawn sketches and local features," Ph.D. dissertation, Université de Fribourg, 2000.

[27] H. Müller, W. Müller, D. M. Squire, S. Marchand-Maillet, and T. Pun, "Performance evaluation in content-based image retrieval: Overview and proposals," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 593–601, 2001.

[28] T. Liu, A. W. Moore, and A. Gray, "New algorithms for efficient high-dimensional nonparametric classification," *Journal of Machine Learning Research*, vol. 7, no. Jun, pp. 1135–1158, 2006.

[29] S. V. Sakhare and V. G. Nasre, "Design of feature extraction in content based image retrieval (cbir) using color and texture," *International Journal of Computer Science & Informatics*, vol. 1, no. II, 2011.

[30] S. Sural, G. Qian, and S. Pramanik, "Segmentation and histogram generation using the hsv color space for image retrieval," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 2. IEEE, 2002, pp. II–II.

[31] G. Wolberg and S. Zokai, "Robust image registration using log-polar transform," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 1. IEEE, 2000, pp. 493–496.

[32] T. H. Reiss, "Recognizing planar objects using invariant image features," *Lecture notes in computer science*, 1993.

[33] Z. Guo, L. Zhang, and D. Zhang, "Rotation invariant texture classification using lbp variance (lbpv) with global matching," *Pattern recognition*, vol. 43, no. 3, pp. 706–719, 2010.

[34] M. A. Stricker and M. Orengo, "Similarity of color images," in *Storage and Retrieval for Image and Video Databases III*, vol. 2420. International Society for Optics and Photonics, 1995, pp. 381–393.

[35] J. Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabi, "mage indexing using color correlograms, computer vision and pattern recognition, 1997," in *Proceedings, 1997 IEEE Computer Society Conference*, pp. 17–19.

[36] M. Presutti, "La matriz de co-ocurrencia en la clasificación multiespectral: tutorial para la enseñanza de medidas texturales en cursos de grado universitario," *4ª Jornada de Educação em Sensoriamento Remoto no Âmbito do Mercosul*, 2004.

[37] S. Kouro and R. Musalem, "Tutorial introductorio a la teoría de wavelet," *Chile: Universidad Técnica Federico Santa María*, 2002.

[38] S. Mallat, *A wavelet tour of signal processing*. Elsevier, 1999.

[39] J. Shukla and J. Vania, "A survey on cbir features extraction techniques," *International Journal Of Engineering And Computer Science*, vol. 3, no. 12, 2014.

[40] F. Long, H. Zhang, and D. D. Feng, "Fundamentals of content-based image retrieval," in *Multimedia Information Retrieval and Management*. Springer, 2003, pp. 1–26.

[41] "From xowa: the free, open-source, offline wiki application," http://xowa.org/, accessed: 2018-10-16.

[42] F. Kaliyadan *et al.*, "Image manipulation and image plagiarism–what's fine and what's not?" *Indian Journal of Dermatology, Venereology, and Leprology*, vol. 83, no. 5, p. 519, 2017.

[43] "Editorial policies, Journal of Cell Biology," http://jcb.rupress.org/editorial-policies#data-integrity.

[44] D. Tkaczyk, P. Szostek, M. Fedoryszak, P. J. Dendek, and Ł. Bolikowski, "Cermine: automatic extraction of structured metadata from scientific literature," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 18, no. 4, pp. 317–335, 2015.

Anexos

Parte A

Nº de imagen	Segundos/iteraciones
1	18.72
2	18.82
3	14.01
4	11.33
5	12.88
6	13.85
7	14.57
8	15.05
9	15.35
10	15.54
11	15.74
12	15.93
13	16.10
14	16.24
15	16.38
16	16.50
17	16.58
18	16.66
19	16.71
20	16.77
21	16.80
22	16.84
23	16.86
24	16.89
25	16.91

Tabla 6.1: Segundos por iteración para búsqueda en el programa final para cada imagen extraída de un documento en formato PDF. Parte 1

Nº de imagen	**Segundos/iteraciones**
26	16.94
27	16.96
28	16.97
29	16.99
30	17.02
31	17.03
32	17.04
33	17.06
34	17.07
35	17.08
36	17.09
37	17.11
38	17.13
39	16.78
40	16.41
41	16.46
42	16.50
43	16.54
44	16.57
45	16.61
46	16.64
47	16.67
48	16.70
49	16.73
50	16.75
51	16.78

Tabla 6.2: Segundos por iteración para búsqueda en el programa final para cada imagen extraída de un documento en formato PDF. Parte 2

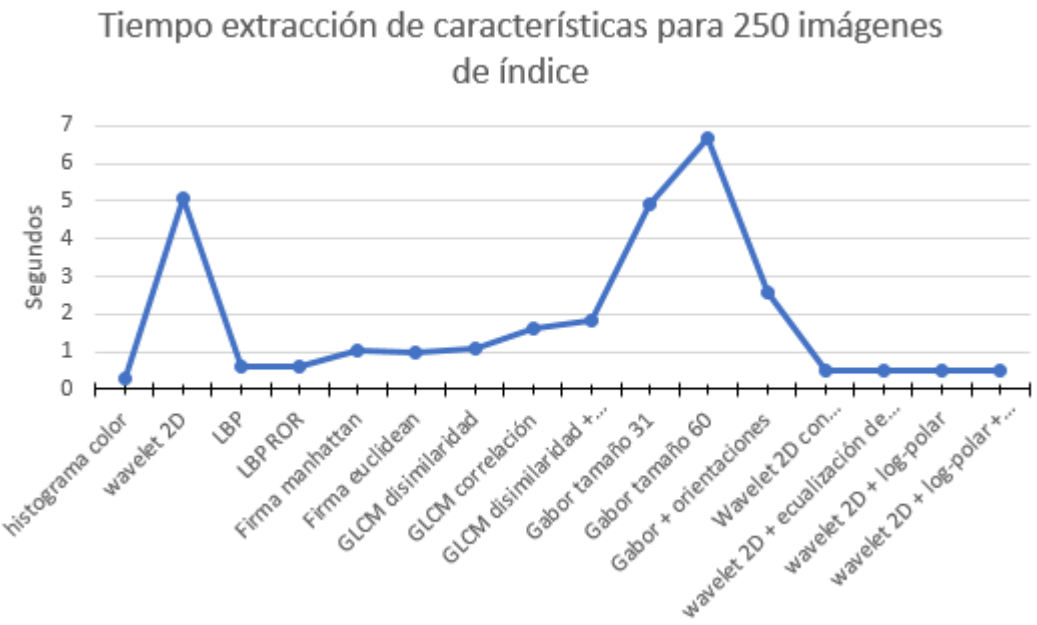


Figura 6.1: Tiempos de ejecución de cada descriptor para la generación de características de imágenes para la base de datos de 250 imágenes de prueba .

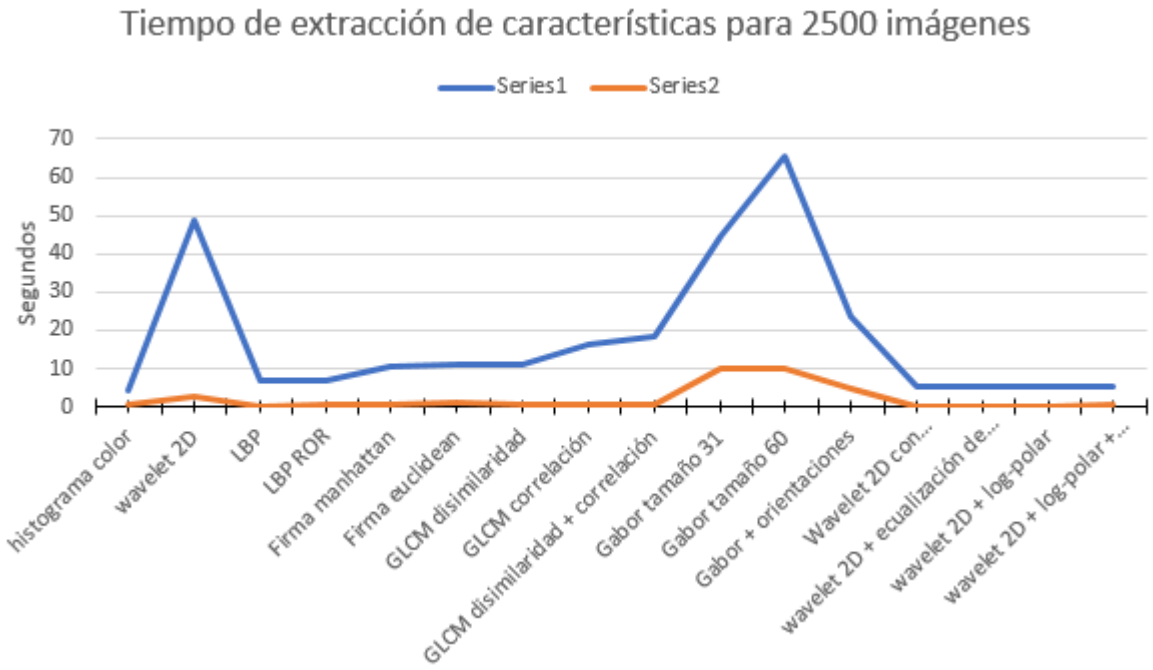


Figura 6.2: Tiempos de ejecución de cada descriptor para la generación de características de las imágenes de consulta generadas a partir de las 5 transformaciones que conforman 2500 imágenes.

Parte B

A plagiarism detection engine for images in Docode

1st Simón Sepúlveda
Electrical Engineering Department
University of Chile
Santiago, Chile
simon.sepulveda@ing.uchile.cl

2nd Gaspar Pizarro V.
Web Intelligence Centre
University of Chile
Santiago, Chile
gaspar.pizarro.v@wic.uchile.cl

3rd Juan D. Velásquez
Industrial Engineering Department
University of Chile
Santiago, Chile
jvelasqu@dii.uchile.cl

***Abstract*—Plagiarism is turning in someone else's work as your own. Though tools exist for checking the originality of texts, those do not work with images, which can be plagiarized as well. In previous works, we have developed the Docode plagiarism detection system, which works with text, but, as commercial requirements have evolved, it is necessary for it to be able to work with images. In this work we present a plagiarism detection engine for images, which works by fusing texture and color features in a weighted combination, so that it can work as a general-purpose engine. We ran experiments with the system, analyzing the improvement made by fusing color and texture features, and the impact of downsizing images on the performance of the system. We see that we reach a recall at 10 elements of 80% and the system allows an image compression of one half without considerable impact on its performance, and with this we can conclude we can build a plagiarism detection engine for images, able to handle general collections for its integration in the Docode system.**

***Index Terms*—Plagiarism detection, Content based image retrieval**

I. Introduction

According to [1], plagiarism is "turning in someone else's work as your own". This can happen in text-based work, but also in graphic work. Current mainstream tools for automatic plagiarism detection, like Turnitin[1], Quetext[2], Grammarly[3], or CopyLeaks[4], work mainly with text, but, to the best of our knowledge, they do not check the originality of images in the documents[5], while they can be plagiarized as well.

According to [2], ideas cannot be patented, but can be plagiarized, unless the idea is commited to a paper, where an idea can be plagiarized and its copyright can be infringed. With this in mind, we can see, text plagiarism do not lead always to copyright infringement, but in the case of graphic work, as it is a "commitment" of an idea, plagiarism can lead to copyright infringement, so content creators have to be aware of reuse of their images. This makes more patent the necessity of a way to do automatic plagiarism detection on images, as it can be used as a way to protect a creator's content.

In previous works, we have developed the Docode system [3], [4], a system for plagiarism detection which, as others, works only with text. As our system works with digital documents by extracting the text from them as the first step, we have access to the images within the documents, but we are currently not using them. Commercial requirements have presented the necessity of working with these images, that are currently wasted in the document analysis process. With a plagiarism engine for images in Docode, we could add value to it by analyzing more of the document than just its text.

Our objective is to add the functionality of image analysis to Docode. So, in this work, as a first step towards that objective, we present a plagiarism detection engine. When an image database is loaded, it can process suspect images, which may or may not be subject to modifications that, while making the result image not identical as the original, still counts as plagiarism. We test our engine with a set of images that satisfy accepted definitions of plagiarism, we show the benefits of feature fusion and the impact of downsizing the database in the system.

Our contribution in this paper is presenting an engine for plagiarism detection in images using different content-based features, analyzing their impact on the performance of the system and analyzing the impact of downsizing the database.

The rest of this document is as follows. Section II is about other work done in the same areas as this work, namely content-based image retrieval systems and their application to plagiarism detection. Section III shows the features of the system built. Section IV shows a draft of the integration of the presented system with the Docode system. Section V shows the experiments done with the system, namely the analysis of the weights of the features and the impact of images downsizing on the performance of the system. Section VI explains modifications that can be done to the system in order to improve it. Finally, section VII, show the conclusions from this work.

II. Related work

A. Content-based image retrieval

Content-based image retrieval (CBIR) systems are systems for image retrieval that rely on features that can be extracted from the images, as opposed to systems that use image metadata (text) to perform retrieval, the former being popular in the computer science community, whereas the latter being popular in the library science community [5]. Surveys of the subject can be found in [6], [7], [8], [9]. As described in

[1] http://turnitin.com/
[2] https://www.quetext.com/
[3] https://www.grammarly.com/
[4] https://copyleaks.com/
[5] Copyleaks works with images, but it is to perform Optical Character Recognition and extract the text from them.

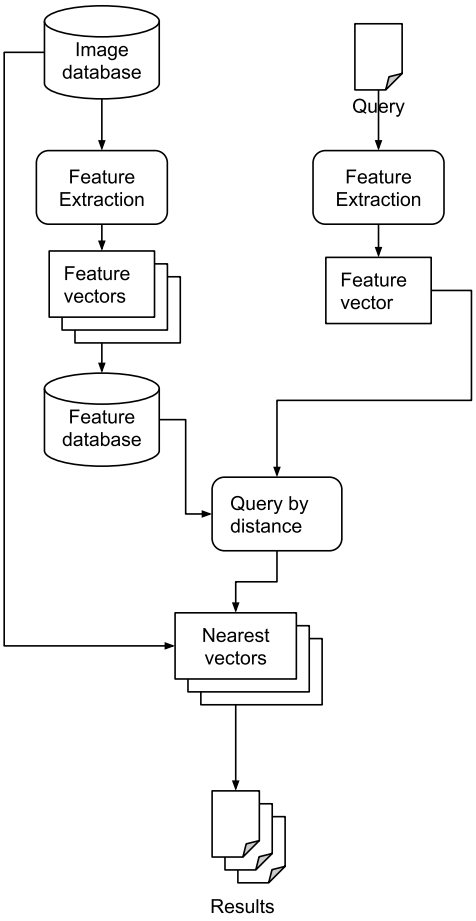


Fig. 1: Flowchart for content-based image retrieval systems

[7], a CBIR system works by extracting feature vectors from an image database, and storing them in another database (the *feature* database). When the user wants to find images related to a *query* image, the system extracts a feature vector from the query image in the same fashion as done with the image database, and does a distance query in the feature database, extracting the closest neighbors of the query vector. With identified vectors, the system can search in the image database and return a set of result images. Figure 1 shows a flowchart for the process.

Different CBIR systems have different algorithms for feature extraction from images and different distance measures for making distance queries.

1) Performance measures: We can describe a CBIR system as a system that has a set of images I, receives a query image q, which has, according to a ground truth definition, a set of relevant images $S \subset I$, and returns a set of images $R \subset I$. According to [10], though it is not the best, a useful way to measure the performance of a CBIR system is using the same measures as in the Information Retrieval area, precision and recall, defined as follows:

Precision It is the fraction of the retrieved images that are actually relevant to the query

$$Precision(q) = \frac{|S \cap R|}{|R|} \tag{1}$$

Recall It is the fraction of the relevant images that are actually retrieved

$$Recall(q) = \frac{|S \cap R|}{|S|} \tag{2}$$

A CBIR system generally can be adjusted to return any number of images as required. Thus, precision can be adjusted to be very high, making the system return as few images as possible, and recall can be adjusted to be very high, by making the system return a large result set (or maybe the whole image database), ensuring the relevant images will be in it. Because of that, precision and recall are usually shown in combination. A way to see this measures working together is to build a *precision-recall* (PR) graph, which shows how the two measures evolve as we make the system return more results. Figure 2 shows an example of a PR graph. As we can

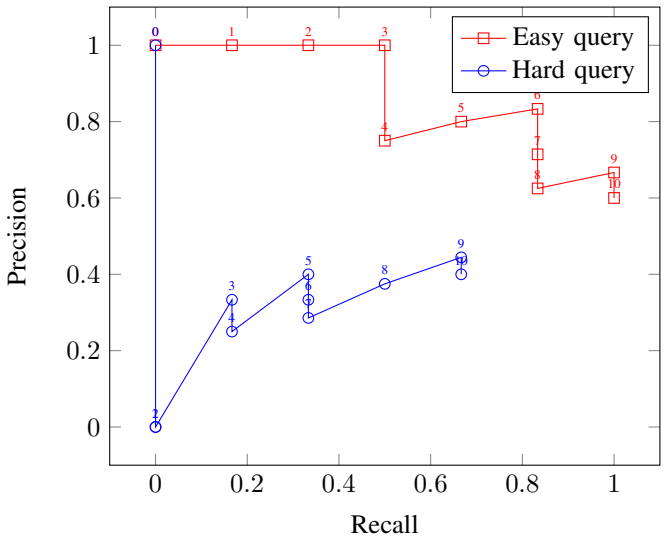


Fig. 2: Precision-recall graph for two hypothetical queries. The labels in the points show how many images are retrieved. We asume that precision is 1 when the system returns 0 images.

see in Figure 2, with an "easy" query, a CBIR system starts returning relevant items, thus having high precision at low recall, until at some point (when the recall is already high) it begins to return irrelevant images to the result, thus lowering its precision, whereas with a "hard" query, the system has to return many images (with a lot of irrelevant ones in the group) just to return the relevant ones, thus having low precision for

a high recall[6]. We can also see that at a certain number of images returned, the system has a better precision and recall with an easy query than with a hard query.

A PR graph can have too much information about the performance of the system, so a way to have a single-valued measure of the performance of a CBIR system is to select some key points in it. This way, we can have:

- $P(N)$: Precision after N elements have been retrieved.
- $R(N)$: Recall after N elements have been retrieved.
- $R(P^*)$: Recall when precision is just below P^*.

In the case of plagiarism detection, for each query there is only original image the query image is plagiarizing from. But also, since this system is intended to be used for document analysis with eventual user input, the system should be able to return more than one image, and eventually, in the best case, one of them is the right one, not necessarily the first one. Because of that, we prefer to evaluate the system using $R(10)$, the recall when retrieving 10 elements. This measure can be averaged across a whole query set in order to compare two CBIR systems.

B. CBIR-based plagiarism detection

Regarding the works in CBIR systems oriented to plagiarism detection, we can see the work is divided in two approaches: specialized and general-purpose engines. The first approach is oriented to build a plagiarism engine with focus in one class of images (like only flowcharts or only landscapes), whereas in the second approach the goal is to build an engine with an image database of images without any domain in mind.

Regarding specialized engines, we see that in [11] the authors use shape detection techniques to transform flowcharts into their base components (circles, rectangles, diamonds) and compare flowcharts by comparing the number of similar base components, using cosine distance to measure similarity. In [12], the authors build another flowchart-oriented plagiarism detection engines, and they propose alterations of input images that could be considered as plagiarism, like rotation and shearing of images. In [13] the authors build a plagiarism detection engine focused only in charts, that uses text-based and image-based features to compare charts that have the same information presented in different ways.

Regarding general-purpose engines, in [14], the authors use a Perceptual Hash-based spatial and frequency transformations to build a general-purpose image retrieval engine. In [15] the authors use Perceptual Hash and SIFT features and build a inverted index of a locally-sensitive hashing of the features for searching. In [16], the authors develop a plagiarism detection engine based on fuzzy logic to find plagiarized images, and can deal with cropped images.

As we can see from the works before, in specialized engines the focus is to try to obtain semantic, domain-based information about the images, whereas in general-purpose engines only generic features are obtained. The first approach can lead to engines that can deal with images that are not alike as images but that contain the same concepts (to which some would say they are the same), but can only deal with one domain of images, and the second approach can lead to engines that, while able to work with images from many domains, are less capable of working with extracting conceptual, high-level information about the images.

C. Contributions of this paper

In this paper, as said in Section I, we attempt to develop a general-purpose plagiarism detection engine for images using color and texture features. With the engine, we study the impact of the each set of features, in its performance and also the impact of image downsizing in the database, with the aim of having the smallest possible database, which should eventually integrate with Docode.

III. Proposed system

The system is developed to handle queries for a large number of different images from multiple domains. For this reason we propose a system that can extract features from color and texture in a non-expensive way. The features used here are of two types: Color-based and texture-based. Color features work well for plagiarized images [17], but there are documents that are not colored like as pure text images or poor-colored images like images with that are zoomed to an area or object of interest. For this problem we can also extract texture features in order to deal with that kind of images. With that taken into account, the features used in the system are:

H-S histogram The H-S histogram is the histogram of the different values of hue and saturation of each pixel in the image, in HSV space. We use the HSV color space because HSV is more similar to human color interpretation [18], [19]. We separate the H-S space in 16 bins for hue and 32 for saturation, which gives us 512 features.

Gabor filter Gabor filters are image filters used for texture analysis [20]. The impulse response for a Gabor filter, in this case, is defined as

$$G(x,y) = \exp\left(-\frac{x^2+\gamma y^2}{2\sigma^2}\right)\sin\{W[x\cos(\theta)+y\sin(\theta)]\} \tag{3}$$

Equation 3 shows that the filter is defined by a standard deviation σ, an aspect ratio γ rotation angle θ and a wavelength W. For our system, we use $\sigma = 4$, $\gamma = 0.5$, and we divide the image in 4 non-overlapping regions. For each one we apply a Gabor filter of size 31, rotated in $0°$, $45°$, $90°$ and $135°$, each one with a a wavelength of 7 and 11 pixels, and we extract the energy and the

[6]"Easy" and "hard" queries refer to how similar is a query document from the relevant database documents according to the extracted features. This is context dependent and apply to every information retrieval system. For instance, in a text-based system that extracts terms from the documents with only basic preprocessing, such as lowercasing every term, an easy query could be a verbatim copy of a document that is in the database, where one can expect the first result to be the exact document we want, whereas a hard query could be a document that is a copy of an element of the database but with all terms (as possible) replaced by a synonym, where we know the similarity between the query and the relevant documents is not much according to the extracted features (the terms), and thus we cannot expect the relevant item to be in the first results.

standard deviation of the coefficients for each filter, giving 64 features.

Gray-level co-ocurrence matrix (GLCM) The GLCM is used for texture analysis [21]. In an image I with l levels of color, the GLCM with an offset δx, δy is defined as an $l \times l$ matrix, where

$$c_{i,j} = \text{count}[I(x,y) = i \wedge I(x+\delta x, y+\delta y) = j)] \quad (4)$$

Equation 4 shows that the GLCM is defined by its step (δx and δy) and the number of levels of color l. In this case we convert the image to grayscale, look up in four directions at one pixel distance and two levels of color, and from these matrices (each one for each direction) we extract the dissimilarity and the homogeneity, as defined in Equations 5 and 6:

$$D = \sum_{i,j=0}^{l-1} P_{i,j}|i-j| \quad (5)$$

$$H = \sum_{i,j=0}^{l-1} \frac{P_{i,j}}{1+(i-j)^2} \quad (6)$$

where P is the GLCM and l are the number of levels of gray. This gives us 8 features.

Finally, as in [22], the distance between two vectors in the system is defined by a linear combination of the distance according to each one of the feature classes extracted. Figure 3 shows a diagram of the distance query in our system.

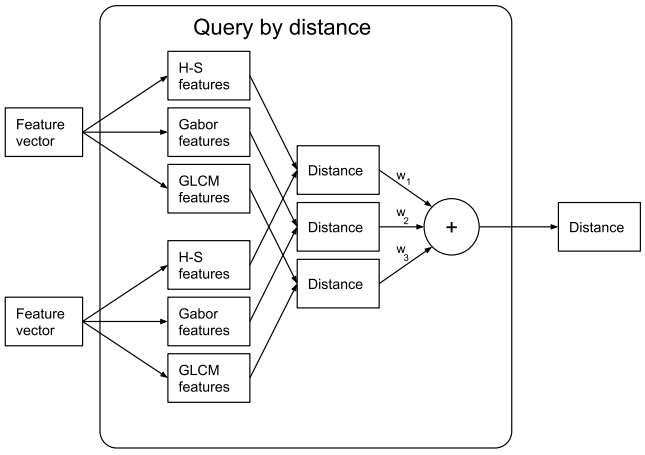


Fig. 3: Query by distance in our proposed system. The coefficients w_1, w_2 and w_3 have to be adjusted in order to maximize its performance.

IV. Integration in the Docode system

The Docode system works by taking a digital document and extracting its text for analysis, and the images can be extracted as well. In this case, parallel to the text extraction step, an image extraction step could be performed and use this system with the images extracted. As it is completely independent of the text analysis and the system works with a queue for each type of analysis, the system proposed in this paper would simply be another "worker" in Docode, as shown by Figure 4. The main problem with the integration, more than

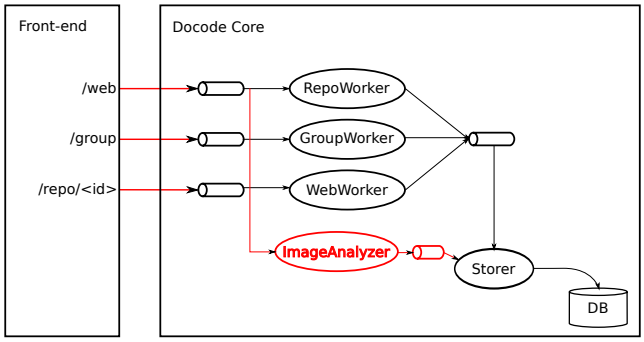


Fig. 4: Modifications to the architecture in [3]. The red `ImageAnalyzer` worker would extract the images of the documents and analyze them without impact on the other prcesses

the integration of the module in the Docode workflow, will be integrating the results of this system to the user interface in an understandable way.

V. Experiments

The system was built in Python, using the OpenCV library [23] and the scikit-image library [24]. All the experiments where run on a computer with a 4-core Intel Core i3 processor running at 3.9 GHz using 7.7 GB of RAM, running Ubuntu 16.04.

A. Original images

The dataset used in the experiments is a small subset of the images from Spanish-language Wikipedia, obtained by through the XowA project[7] from which around 250 images were randomly selected, creating a small dataset of 6.7 MB. These images, as they are from Wikipedia, we consider it as a general-domain database. These images are the "original" images, from which the feature database is built.

B. Transformations

Since this is an engine for detecting plagiarized images, we need to test it with plagiarized images, that is, images that are transformations of the original, intended to fool simple, pixel-based similarity measures. According to [25] and [26], images can be cropped, their brightness can be adjusted, their color balanced and rotated to achieve a better interpretation of content. With that in mind, we built a set of "plagiarized" images, which is a set of the images of the original dataset, after a combination of the following transformations.

Cropping Images are cropped by selecting a sub-rectangle of dimensions and position selected randomly inside the image. The rectangle is built by selecting a segment in each dimension of the images and taking the intersection of the bands defined by the segments. In each dimension,

[7] `http://xowa.org/`

the width of the segment follows a Gaussian distribution centered in the length of the dimension, and a variance 0.2 (segments that are larger than the dimension are discarded), and its offset is selected uniformly in the dimension, so that the segment fits in it.

Color The RGB components of the image are altered by adding an uniformly selected number in $[-5, 5]$ to each component.

Brightness The lightness of the image in HSV components is altered by adding a random number that follows a Gaussian distribution with mean zero an variance 50.

Rotation Images are rotated and the angle or rotation follows a Gaussian distribution of mean $30°$ and variance $10°$. In this case, images are cropped so that there are no empty spots after the rotation.

Figure 5 shows examples of these transformations. For the

(a) Original image (b) Cropped image (c) Brightness alterations

(d) Color alterations (e) Rotation (f) All transformations combined

Fig. 5: Transformations to the images.

experiments, we have an "easy" query set, which consists of images that have rotation and brightness alterations, and a "hard" query set, which consists of images that have all alterations combined, as in Figure 5f. The metric used in all of the experiments is the $R(10)$, as explained in section II-A1.

C. Dissimilarity Metrics

In order to find the best dissimilarity metric for the system, a test run was done, combining all the features with the same weight. Figure I shows the measurement of system's average recall for different dissimilarity metrics, at 10 returned images, using the easy query set. In this case, we simply sum the distances of each feature, color and texture.

Metric	R(10)
Chebyshev	0.4608
Sokal-Michener	0.4872
Canberra	0.5864
Sørensen–Dice	0.4784
Euclidean	0.714
Bray-Curtis	0.6908
Manhattan	0.708
Jaccard	0.4784
Sokal-Sneath	0.4784
L_1	0.708
L_∞	0.4608

TABLE I: Recall at 10 elements for different distance metrics, using a non-weighted sum of the feature distances.

In Table I we can see that the metric that achieves best precision and recall is the Euclidean distance, which is the one used in the following experiments.

D. Feature weights

With the Euclidean distance selected, we have to find the best linear combination of features for the system. To achieve this, we ran a grid search of the different weights for the features, as shown in Figure 6, using the easy query set.

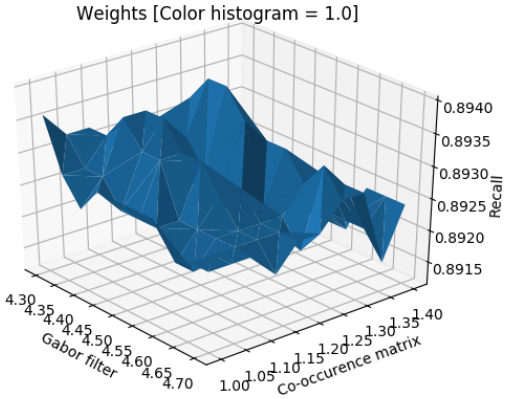


Fig. 6: $R(10)$ for different combinations of weights for texture and color features. In this graph, since what matters is the relative weight of the features, we fix the weight of the H-S histogram to 1, and we explore different combinations of weights for the GLCM and the Gabor features in the horizontal plane, and we plot the R(10) in the vertical axis.

With this, we achieve in the best case a R(10) of 0.9, with the coefficients in Table II.

Features	Weight
H-S histogram	1.00
Gabor filters	4.40
GLCM	1.20

TABLE II: Feature weights for the best linear combination of features.

Finally, with the selected weights for each feature, we ran the same test. The performances are shown in Table III, along with the performances of each feature by themselves.

Feature	R(10)
H-S histogram	0.73
Gabor filters	0.71
GLCM	0.47
Linear combination	0.90

TABLE III: Performance for the features.

E. Comparing query sets

With the calibrated system we can see how the system performs with the different query sets. Figure 7 shows the performance of the system with 10 randomly created easy and hard query sets.

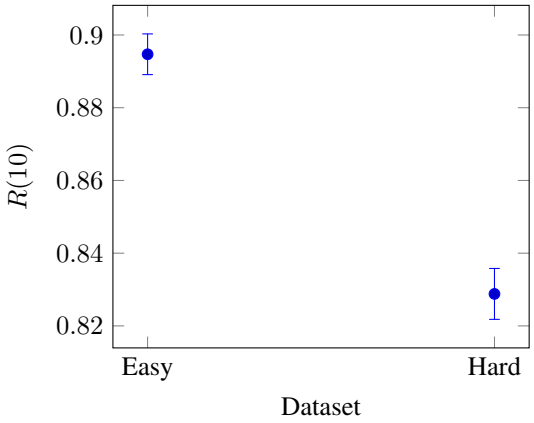


Fig. 7: R(10) of the system with the easy and the hard query set

F. Downsized images

The images in the database were downsized to a half, quarter and octave of their original size for testing the level of compression possible that makes the system performance to work without changes and discover the point where the system stops working properly and cannot handle queries. Figure 8 shows the performance of the system for different degrees of compression of the database.

G. Discussion

In Table III we can see that by themselves, the features give around 70% R(10). Color histogram present the best results, with 73% followed by Gabor filters who have a little descend in performance about 71%, while the GLCM by itself just has 47%. We can also see that the best combination achieves 90% R(10), and, as Table II shows, the feature with the most weight is the Gabor filter, followed by the GLCM and the color histogram. These results show that it is useful to do feature fusion, and that even though some features can be of poor performance, like the GLCM, a correct weight adjustment can make the system work better than with all the features by themselves.

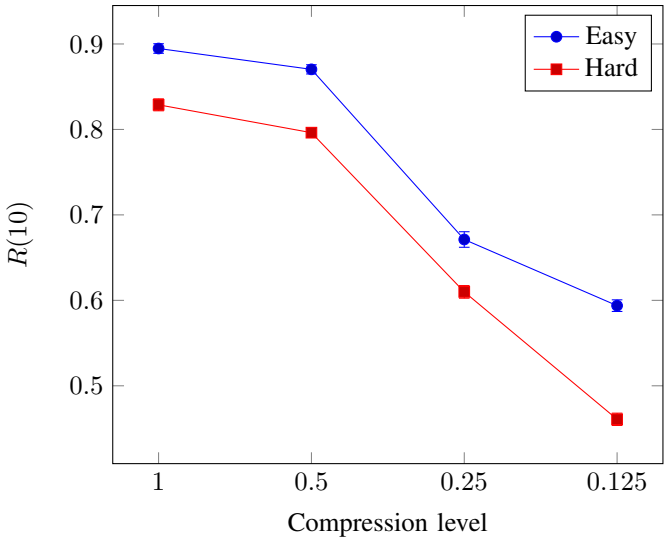


Fig. 8: R(10) of the system with the easy and the hard query set

Comparing the easy and hard query sets, we can see in Figure 7 that the performance of the system with the easy query set is around 89%, whereas the performance with the hard query set is around 83%. This shows that the system has problems with color transformations, which are not completely addressed with the features we use.

In Figure 8 we can see that we can cut the images in half their sizes without too much performance loss, but beyond that the system starts to fail, and at one octave of the size, the system has an unacceptable performance, lower than the 70%. This suggests that, in order to achieve the smallest image database possible, we could add a downsizing step by half while maintaining an acceptable performance.

VI. Future work

As we saw in Section V-G, we must improve the performance of the system with color-transformed images. For this, we could extract other texture-based features. GLCM or Gabor filters can be replaced by other more accurate features and with a fast extraction time, such as the Perceptual Hash [27] or wavelets [28].

Since we use independent groups of features in order to do the search, their computations can be parallelized, thus making the database building process faster.

If more features are used, a grid search will be not enough to find their weights. In this case, a more general, non-convex optimization scheme would be a better alternative, such as an evolutionary algorithm.

In terms of the size of the database, we could try dimensionality reduction approaches in the feature vectors in order to build the smallest possible feature database. This way we could discard the images quickly, making it a less storage-consuming system.

In terms of integration, what we mentioned in Section IV is just a draft of the real integration with the Docode

system. Aside from what we mentioned, we have to develop an interface for this system for Docode to interact with it, modify the pipeline for document analysis in order to extract and analyze the images in the documents, and, most important for the product, we have to implement a user-friendly interface to integrate the results of this system with the results of Docode itself.

Finally, the main problem for the commercial deployment of this system will be populating the database. Collections like Wikipedia's can be seen as a starting point, but our estimation is that a general web-based image collection will be required. For this, we would need to build a web crawler to populate the database. This crawler is definitely out of the scope of this paper, and would require a very different expertise to carry it out.

VII. Conclusions

The proposed CBIR system with fused features demonstrate to gather enough positive results to become a tool for detecting image plagiarism with definitions of plagiarism simulated by a modification on the database images from the Spanish-language Wikipedia.

Finally, we see that fusing features improves the system, so that we can use it as a general-purpose engine, and, with some of the future work done, it could fulfill our objective of adding the image analysis feature to our current plagiarism detection system.

Acnowledgements

This work was supported partially by Fondecyt 1160117 and the Millennium Institute on Complex Engineering Systems (ICM: P-05-004-F, CONICYT: FBO16).

References

[1] H. A. Maurer, F. Kappe, and B. Zaka, "Plagiarism-a survey." *J. UCS*, vol. 12, no. 8, pp. 1050–1084, 2006.

[2] K. M. Dames, "Understanding Plagiarism and How It Differs from Copyright Infringement." *Computers in Libraries*, vol. 27, no. 6, pp. 25–27, 2007.

[3] G. Pizarro V. and J. D. Velásquez, "Docode 5: Building a real-world plagiarism detection system," *Engineering Applications of Artificial Intelligence*, vol. 64, pp. 261–271, 2017.

[4] J. D. Velásquez, Y. Covacevich, F. Molina, E. Marrese-Taylor, C. Rodríguez, and F. Bravo-Marquez, "DOCODE 3.0 (DOcument COpy DEtector): A system for plagiarism detection by applying an information fusion process from multiple documental data sources," *Information Fusion*, vol. 27, pp. 64–75, 2016.

[5] H. Chu, "Research in image indexing and retrieval as reflected in the literature," *Journal of the Association for Information Science and Technology*, vol. 52, no. 12, pp. 1011–1018, 2001.

[6] N. Singhai and S. K. Shandilya, "A survey on: Content based image retrieval systems," *International Journal of Computer Applications*, vol. 4, no. 2, pp. 22–26, 2010.

[7] M. Kokare, B. Chatterji, and P. Biswas, "A survey on current content based image retrieval methods," *IETE Journal of Research*, vol. 48, no. 3-4, pp. 261–271, 2002.

[8] M. Rehman, M. Iqbal, M. Sharif, and M. Raza, "Content based image retrieval: Survey," *World Applied Sciences Journal*, vol. 19, no. 3, pp. 404–412, 2012.

[9] J. Eakins and M. Graham, "Content-based image retrieval," 1999.

[10] H. Müller, W. Müller, D. M. Squire, S. Marchand-Maillet, and T. Pun, "Performance evaluation in content-based image retrieval: Overview and proposals," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 593–601, 2001.

[11] S. Arrish, F. N. Afif, A. Maidorawa, and N. Salim, "Shape-Based Plagiarism Detection for Flowchart Figures in Texts," *arXiv preprint arXiv:1403.2871*, 2014.

[12] P. Ovhal, "Detecting plagiarism in images," in *Information Processing (ICIP), 2015 International Conference On*, 2015, pp. 85–89.

[13] M. M. Al-Dabbagh, N. Salim, A. Rehman, M. H. Alkawaz, T. Saba, M. Al-Rodhaan, and A. Al-Dhelaan, "Intelligent bar chart plagiarism detection in documents," *The Scientific World Journal*, vol. 2014, 2014.

[14] V. Bajaja, S. Keluskar, R. Jaisawala, and R. Sawant, "Plagiarism detection of images," *International Journal of Innovative and Emerging Research in Engineering*, vol. 2, no. 2, 2015.

[15] S. Srivastava, P. Mukherjee, and B. Lall, "imPlag: Detecting image plagiarism using hierarchical near duplicate retrieval," in *India Conference (INDICON), 2015 Annual IEEE*, 2015, pp. 1–6.

[16] P. Hurtik and P. Hodakova, "Ftip: A tool for an image plagiarism detection," in *Soft Computing and Pattern Recognition (SoCPaR), 2015 7th International Conference of*. IEEE, 2015, pp. 42–47.

[17] S. Kodituwakku and S. Selvarajah, "Comparison of color features for image retrieval," *Indian Journal of Computer Science and Engineering*, vol. 1, no. 3, pp. 207–211, 2004.

[18] S. Sural, G. Qian, and S. Pramanik, "Segmentation and histogram generation using the HSV color space for image retrieval," in *Image Processing. 2002. Proceedings. 2002 International Conference On*, vol. 2. IEEE, 2002, pp. II–II.

[19] N. Herodotou, K. Plataniotis, and A. Venetsanopoulos, "A color segmentation scheme for object-based video coding," in *Advances in Digital Filtering and Signal Processing, 1998 IEEE Symposium On*. IEEE, 1998, pp. 25–29.

[20] T. Randen and J. H. Husoy, "Filtering for texture classification: A comparative study," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 4, pp. 291–310, 1999.

[21] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, 1979.

[22] Y.-H. Lee, S.-B. Rhee, and B. Kim, "Content-based image retrieval using wavelet spatial-color and Gabor normalized texture in multi-resolution database," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference On*. IEEE, 2012, pp. 371–377.

[23] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[24] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "Scikit-image: Image processing in Python," *PeerJ*, vol. 2, p. e453, Jun. 2014.

[25] F. Kaliyadan and others, "Image manipulation and image plagiarism–what's fine and what's not?" *Indian Journal of Dermatology, Venereology, and Leprology*, vol. 83, no. 5, p. 519, 2017.

[26] "Editorial policies, Journal of Cell Biology," http://jcb.rupress.org/editorial-policies#data-integrity.

[27] C. Zauner, "Implementation and benchmarking of perceptual image hash functions," 2010.

[28] S. Livens, P. Scheunders, G. van de Wouwer, and D. V. Dyck, "Wavelets for texture analysis, an overview," in *1997 Sixth International Conference on Image Processing and Its Applications*, vol. 2, Jul. 1997, pp. 581–585 vol.2.