Reconocimiento de patrones utilizando mapeo logarítmico-polar y método SIFT

Saul Mena Avila

Licenciatura en Ciencias de la Computación
Facultad de Matemáticas
Universidad Autónoma de Yucatán
2008

Declaración

Por este medio declaro que soy el autor de este trabajo
el cual representa mi tesis de Licenciatura.

Saul Mena Avila
Mérida, Yucatán
México
17 de julio de 2008

A Dios,

que me dio la Vida, y ha estado conmigo en mis días más oscuros.

A mi padres,

que me han dado todo cuanto he necesitado.

Agradecimientos

Al Dr. Arturo Espinosa Romero, mi asesor, por su apoyo y gran optimismo.

Este trabajo se llevó a cabo con el apoyo del Programa de Impulso y Orientación a la Investigación (PRIORI) de la Universidad Autónoma de Yucatán.

Resumen

La visión computacional es un área de investigación cuyo objetivo es dotar a las computadoras del sentido de la vista; más específicamente, se desea que las computadoras puedan "entender" el mundo real a través de las imágenes de éste de modo que puedan interactuar. Inmediatamente se desprende la idea de reconocer los elementos de una imagen dada.

El reconocimiento de patrones es un área de investigación que trata el problema de la clasificación de objetos, cualesquiera que éstos sean. En el contexto de la visión computacional, los objetos del mundo real son clasificados a partir de los datos que se pueden obtener de sus imágenes. Con esos datos se crea una representación (o patrón) que sirve para distinguirlo de otros objetos de manera confiable.

En este trabajo se describen dos métodos para representar imágenes: el mapeo logarítmico polar y el método SIFT. Ambas transformaciones proveen características de invariabilidad que pueden ser aprovechadas en el reconocimiento de patrones y con la intención de probarlo se implementaron también métodos sencillos de reconocimiento para cada uno.

El mapeo logarítmico-polar es una función matemática que al aplicarse a una imagen, genera otra llamada imagen log-polar, que posee propiedades de invariabilidad a la escala y a la rotación. Estudios en la visión de los primates sugieren que los estímulos captados por la retina son mapeados en su trayecto a la corteza cerebral mediante esta función. Esto representa una motivación para su implementación y augura buenos resultados.

El método SIFT toma una imagen como entrada y devuelve un conjunto de rasgos que la describen. Éstos reciben el nombre de rasgos SIFT y son invariables a la escala, la rotación y en menor medida al cambio de punto de vista y de iluminación.

La aplicación de reconocimiento utilizando el mapeo logarítmico-polar se construyó con base a una versión modificada del algoritmo k–medios. Los resultados obtenidos en este trabajo muestran que la aplicación del mapeo contribuyó significativamente a identificar correctamente objetos rotados, sin embargo la clasificación de objetos escalados no tuvo tan buenos resultados. No se hizo experimentación exhaustiva respecto al reconocimiento de patrones utilizando el método SIFT ya que se detectaron problemas de repetibilidad en la implementación desarrollada; sin embargo se llevó a cabo un único experimento de reconocimiento con fines ilustrativos que demostró la efectividad del método.

Índice general

Índice de figuras

Capítulo 1

Introducción

La visión computacional es un área de investigación que busca dotar a las computadoras del sentido de la vista, de modo que puedan reaccionar conforme a lo que ven. Este es un paso natural en el camino de construir una computadora inteligente dada la importancia que la vista tiene en nuestra vida diaria. Las aplicaciones derivadas de la investigación en esta área ya están presentes en la industria, sin embargo aún queda un largo camino por recorrer.

El reconocimiento de patrones es un área multidisciplinaria que trata el problema de la asignación de objetos a un conjunto de clases. Dentro de la visión computacional, el reconocimiento de patrones es primordial ya que es necesario saber cuál es el objeto que se está viendo antes de poder actuar.

En este trabajo se describen e implementan dos métodos para la representación de imágenes, el mapeo logarítmico-polar (en adelante, log-polar) y el método SIFT, que pueden repercutir positivamente en el reconocimiento de patrones dentro del contexto de la visión computacional. Ambos métodos proveen de características de invariabilidad que son deseables en el reconocimiento de patrones.

El mapeo log-polar es una función matemática notable por su trasfondo biológico ya que existen estudios que indican que el proceso de visión temprana de los primates está basado en él. Por su parte, el método SIFT fue desarrollado y patentado por David G. Lowe y es relativamente nuevo; actualmente es uno de los métodos de representación más populares y existe mucho estudio a su alrededor.

En este capítulo se presenta el trabajo de tesis, indicando la hipótesis y los objetivos de la misma. En el Capítulo 2 se da una introducción al tema de la visión computacional y el reconocimiento de patrones así como también un repaso de los temas que se tratan directa o indirectamente en el desarrollo del trabajo.

En el Capítulo 3 se presenta el mapeo log-polar cubriendo el modelo matemático, el modelo biológico y sus propiedades; también se hace un recuento de algunas de las implementaciones existentes y posteriormente se describe la implementación desarrollada. Por último se ejemplifica la aplicación del mapeo log-polar.

El Capítulo 4 introduce el método SIFT y describe el proceso para transformar una imagen en sus rasgos SIFT. Luego se describen los detalles de implementación que corresponden a la implementación desarrollada en este trabajo y finalmente se presentan los resultados de la misma.

El Capítulo 5 se divide en dos partes. La primera está dedicada al reconocimiento de patrones utilizando el mapeo log-polar y la segunda, al reconocimiento de patrones utilizando el método SIFT. En ambos casos se cubre la descripción del método de reconocimiento, las condiciones de experimentación y los resultados obtenidos.

Las conclusiones derivadas del trabajo se presentan en el Capítulo 6, así como también el trabajo futuro pertinente.

1.1. Hipótesis

El mapeo logarítmico-polar puede implementarse como parte de un método de reconocimiento para clasificar letras del alfabeto castellano utilizando el método SIFT como un punto de comparación.

1.2. Objetivos

1. Descripción e implementación del mapeo log-polar.
2. Descripción e implementación del método SIFT.
3. Aplicación y evaluación del mapeo log-polar en el reconocimiento de patrones.

4. Aplicación y evaluación del método SIFT en el reconocimiento de patrones.

Originalmente se tuvo la intención de integrar el mapeo log-polar y el método SIFT para crear un nuevo método de representación que se beneficiara de las características de ambas transformaciones. Sin embargo, esto no fue posible debido a que la implementación del método SIFT presentó un mayor reto del que se había considerado.

1.3. Trabajo previo

Durante el IV Verano de la Investigación 2005 se desarrolló una implementación del mapeo log-polar en conjunto con el Dr. Arturo Espinosa Romero. Dicha implementación se retomó para desarrollar el presente trabajo y como parte de las actividades llevadas a cabo, fue revisada, refinada e incluida en la biblioteca VisionLibs[1].

[1] Biblioteca de uso interno de la Facultad de Matemáticas.

Capítulo 2

Estado del Arte

En este capítulo se presenta una compilación de varios temas que se tratan directa o indirectamente en el desarrollo del presente trabajo con el propósito de introducir al lector en el contexto del mismo, así como para proveer de una base de conocimientos para aquellos lectores sin experiencia en el área de visión computacional o que desconocen alguno de los temas tratados.

2.1. ¿Qué es la visión computacional?

La vista es sin duda el sentido más explotado por los seres humanos y es vital en la gran mayoría de actividades humanas. Nalwa (1993) dice acertadamente "una imagen es la que vale mil palabras, y no un toque, no un sonido, no un sabor y no un olor".

En la meta para construir una verdadera computadora inteligente, dotar a las máquinas de percepción visual es a la vez un objetivo primario y un obstáculo mayor. Superar este obstáculo es el fin último de la investigación en torno a la visión computacional, presente desde 1960.

En un sentido más práctico, el propósito de la visión computacional es lograr que las computadoras puedan interpretar el mundo real con base en imágenes captadas y reaccionar apropiadamente. La extensión de su estudio abarca desde la generación de las imágenes hasta el uso "inteligente" de los datos que contienen, lo que la convierte en un área que se apoya en otras disciplinas como la geometría, óptica, fisiología, psicología, procesamiento de imágenes, inteligencia artificial, etcétera.

Entre las dificultades que presenta la visión computacional está la naturaleza ambigua del mundo real debido a la cual diferentes escenas pueden ser representadas por la misma imagen, por ejemplo una bola de billar y un balón de futbol americano visto desde arriba. Además, la representación del mundo real en una medio bidimensional, como lo es una imagen, implica la pérdida de información útil.

¿Por qué no imitar el proceso de visión humano? (Nalwa, 1993). La respuesta es que aún no se sabe lo suficiente acerca del funcionamiento interno del cerebro, del cuál solo se tienen modelos y conjeturas. Sin embargo se sabe que la visión humana aunque extraordinaria no es infalible, como lo demuestran las ilusiones ópticas; en este sentido se espera que la visión computacional supere las capacidades humanas y animales (Davis, 1997).

A pesar de que aún se está lejos de lograr la percepción visual en las computadoras, ya hay aplicaciones que están presentes en la industria. Por ejemplo, se utiliza la visión computacional en el control de calidad de partes mecánicas así como en la industria de alimentos, el reconocimiento dactilar, en criminología y en equipo militar.

2.2. Reconocimiento de patrones

El reconocimiento de patrones es un área de investigación multidisciplinaria que trata el problema de la asignación de objetos a un conjunto de clases sin prestar atención al tipo de objeto, ya sea abstracto o concreto.

El término patrón se utiliza para denominar al conjunto de observaciones o mediciones que representan a un objeto cualquiera, razón por la cual también se le conoce como representación o rasgo. La definición del patrón es una de las tareas cruciales en esta área.

El reconocimiento de patrones es una parte muy importante dentro de la visión computacional ya que es necesario identificar aquello que se ve antes de tomar una acción. Bajo este contexto, los objetos del mundo real son clasificados a partir de los rasgos obtenidos de sus imágenes.

2.3. La imagen digital

La imagen digital es el resultado del muestreo y la cuantización de la imagen óptica correspondiente. Esta última, también llamada imagen física, se forma cuando los rayos luminosos que han sido reflejados por un objeto inciden en una superficie bidimensional llamada plano de la imagen.

El muestreo consiste en tomar un conjunto finito de puntos igualmente espaciados del plano de la imagen mientras que la cuantización asigna el valor de intensidad de luz que incide sobre cada uno de ellos. Aunque existen diferentes dispositivos que pueden llevar a cabo este proceso, hoy en día el más popular es el dispositivo de carga acoplada o CCD por su nombre en inglés (*charged-coupled device*), presente en todas las cámaras digitales actuales.

Cabe mencionar que la imagen digital es una clase especial de señal que se mide en función del espacio, y no del tiempo como ocurre comúnmente: la intensidad de luz se mide en diferentes puntos del plano de la imagen.

La imagen digital es entonces el conjunto de valores muestreados y cuantizados almacenados en un arreglo bidimensional como se muestra en la Figura 2.1(a). A cada valor del arreglo se le llama *pixel*, abreviación del término en inglés *picture element* (elemento de imagen), y se le asigna una posición de acuerdo al direccionamiento matricial (el primer índice para el renglón y el segundo, para la columna). En ocasiones se hace referencia a un *punto* en una imagen digital; en este caso, los conceptos punto y pixel son equivalentes.

En general, dado el contexto, es innecesario el uso del calificativo "digital" al referirse a las imágenes; por esta razón, en adelante se omite dicho calificativo y debe darse por sobrentendido.

Cada pixel de una imagen, salvo por los que pertenecen a los bordes, tiene 8 vecinos adyacentes. El conjunto de los dos verticales y los dos horizontales recibe el nombre de *vecindad en 4*; si a este conjunto se le agregan los pixeles en las diagonales, se le conoce como *vecindad en 8*. La idea de vecindad puede extenderse a cualquier conjunto de pixeles especificando el pixel al que le pertenece la vecindad y la forma de ésta; la

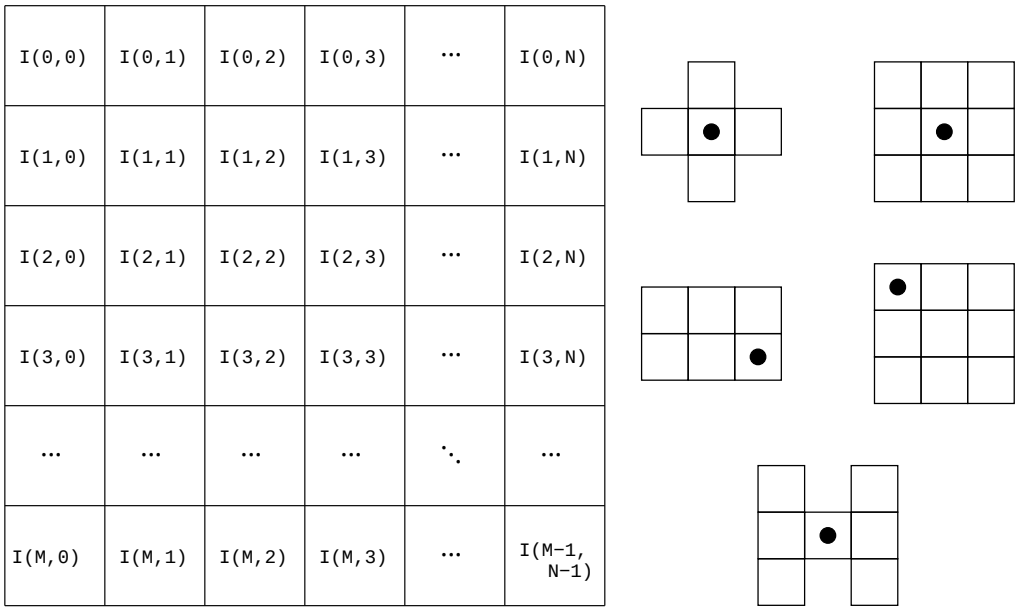


Figura 2.1: (a) Una imagen digital de tamaño $M \times N$. (Basada en una imagen de González y Woods (1996).) (b) Ejemplos de vecindades. El punto marca el pixel al que le pertenece la vecindad.

Figura 2.1(b) muestra algunos ejemplos de vecindades.

El procesamiento de imágenes provee un conjunto de herramientas para el mejorado, la restauración y la compresión de imágenes (Nalwa, 1993) así como también para la extracción de información. En su forma más simple, este procesamiento consiste en la aplicación de operaciones aritméticas a una imagen, pixel a pixel, llevadas a cabo entre una imagen y una constate o entre los pixeles correspondientes de diferentes imágenes; operaciones más complejas involucran una vecindad para cada pixel de la imagen, como es el caso de la convolución que se explica en la siguiente sección.

2.4. La convolución

La convolución es una operación que el procesamiento de imágenes hereda del procesamiento de señales para combinar dos imágenes y generar una tercera. Es la operación de filtrado de imágenes por excelencia que cumple las propiedades de conmutatividad, asosiatividad y distributividad sobre la suma.

A continuación se presenta el caso más simple en el contexto del procesamiento de imágenes con fines ilustrativos.

> Sea I una imagen de $M \times N$ y H una imagen de $(2R+1) \times (2R+1)$ con $R \in \mathbb{N}$. La convolución de H con I es definida por
>
> $$I'(m,n) = (I * H)(m,n) = \sum_{i=-R}^{R} \sum_{j=-R}^{R} H(R-i, R-j) I(m+i, n+j),$$
> $$R \leq m \leq M-R-1, R \leq n \leq N-R-1$$
>
> donde la imagen I', de $M-R \times M-R$, es la convolución de H con I y $*$ es el operador de convolución.

El proceso de convolucionar H con I consiste en reflejar H vertical y horizontalmente (reflexión por el origen), luego desplazar la matriz resultante sobre todo pixel de I y en cada paso realizar la suma de productos entre los pixeles correspondientes de H e I. El resultado de la convolución es una nueva imagen I' cuyo pixel (m,n) es una suma ponderada de los pixeles de I en una vecindad del pixel $(m+R, n+R)$.

H recibe los nombres de máscara, filtro o *kernel* y es usualmente más pequeña que I. La vecindad y la ponderación de las que se habla en el párrafo anterior son especificadas, respectivamente, por el tamaño y los valores de H; esto significa que al asignarle los valores adecuados se logra un efecto definido, como el difuminado o la detección de bordes.

La correlación es una operación similar a la convolución cuya única diferencia radica en que el *kernel* H no es reflejado por el origen, sino que es utilizado tal cual. Cuando H es simétrica respecto al origen, la convolución es equivalente a la correlación.

2.5. Teoría del espacio de escala

La escala juega un papel importante en el estudio de la Naturaleza. Existe una infinidad de objetos de estudio cada uno con diferentes características que a su vez pueden ser objetos de estudio por si mismos; sin embargo, no todas las características de un objeto de estudio se perciben en la misma escala que el objeto; Lindeberg da un claro ejemplo:

> *"Un árbol, por ejemplo, puede parecer tener una forma redonda o cilíndrica al ser visto a cierta distancia, a pesar de que está constituido por un gran número de ramas. En una inspección más cercana, las hojas se hacen distinguibles individualmente, y podemos observar que a su vez, éstas tienen textura en una escala aún más pequeña."* (Lindeberg, 1994b)

Este comportamiento en la Naturaleza se convierte en un problema en el procesamiento de señales cuando se requiere analizar una señal de la cual se desconoce la escala en que fue recolectada. Como consecuencia, es difícil elegir el tipo de operador a aplicar, así como decidir en donde aplicarlo e incluso determinar el tamaño adecuado del mismo.

Una solución a este problema es pues analizar la señal en todas las escalas. La teoría del espacio de escala sienta las bases para construir una familia de señales representativas en diferentes escalas derivadas de la señal original. El conjunto de estas señales recibe el nombre de *espacio de escala* o también, *representación multi-escalar*.

Un requerimiento indispensable es que no se debe crear nuevos detalles en la señal como resultado de la construcción del espacio de escala; esto es, todo detalle en las señales derivadas deben ser resultado de la simplificación de los detalles en la señal original.

Existen varias formas de representar un espacio de escala de acuerdo a cómo se construye. A continuación se presentan algunas de estas representaciones en el contexto de la visión computacional. Cabe mencionar que el método SIFT utiliza la representación denominada *pirámide sobremuestreada*, sin embargo en favor de la claridad se describen previamente otras dos representaciones que muestran la evolución del espacio de escala.

2.5.1. Espacio de escala lineal

El espacio de escala lineal es un conjunto ordenado de imágenes L que se derivan de la convolución de una imagen inicial con una familia de *kernels* de difuminado con nivel de difuminación creciente. Se ha demostrado que el *kernel* Gaussiano es la única opción para la familia de *kernels* (Lindeberg, 1994a). La definición formal del espacio de escala lineal, que determina al mismo tiempo el método para construirlo, se presenta a continuación:

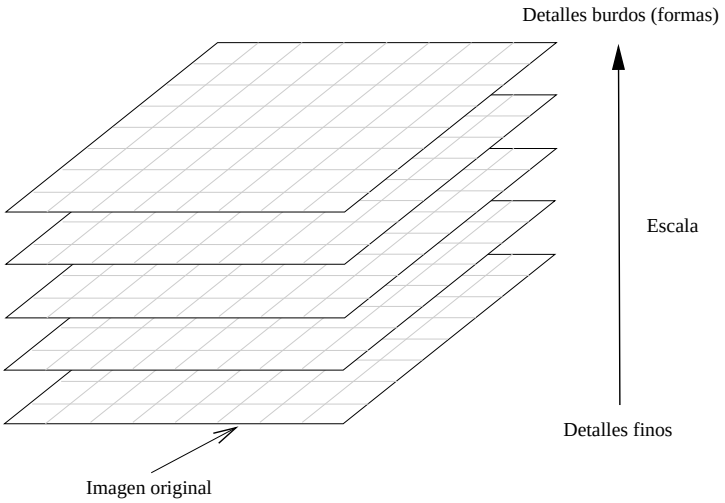


Figura 2.2: Espacio de escala lineal de una imagen. Se genera una pila de imágenes en cuya base se encuentra la imagen original y las imágenes restantes representan a la original en una escala diferente. (Basada en una imagen de Lindeberg (1994a)).

Sea $f : \mathbb{R}^2 \to \mathbb{R}$ una imagen. El espacio de escala lineal $L : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{R}$ es definido por la familia de imágenes $L_0(x, y; 0) = f$ y

$$L_i(x, y; t) = G(x, y; \sqrt{t}) * f, \quad (i \in \mathbb{N})$$

donde $*$ es el operador de convolución, $t \in \mathbb{R}$ es el parámetro de escala y $G : \mathbb{R}^2 \times \mathbb{R}^+ - \{0\} \to \mathbb{R}$ es un *kernel* Gaussiano definido por:

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \tag{2.1}$$

La desviación estándar del *kernel* Gaussiano utilizado para construir la imagen $L_i(x, y; t)$ es la *medida de escala* y es igual a la raíz cuadrada del parámetro de escala.

El espacio de escala resultante se puede ver en la Figura 2.2; la familia de imágenes se concibe como una pila en cuya base se encuentra la imagen original representando la escala más pequeña y en la que se aprecian los detalles finos; conforme se sube de nivel en la pila, la escala se incrementa y como consecuencia se pierden detalles pero se conservan las formas. Se puede pensar en ello como una serie de fotografías a un objeto, incrementando la distancia entre tomas.

2.5.2. Pirámide

En la representación piramidal, además del difuminado por convolución, se aplica una operación de submuestreo en la construcción de cada imagen de la familia. El submuestreo provoca un decremento en el tamaño de las imágenes y es a la vez su ventaja y desventaja; por un lado, se reduce el número de operaciones en el procesamiento, sin embargo la variación del tamaño dificulta relacionar los rasgos entre imágenes. Un ejemplo de este espacio de escala se muestra en la Figura 2.3.

Se propone la siguiente definición para la construcción del espacio de escala piramidal:

> Sea $f : \mathbb{R}^2 \to \mathbb{R}$ una imagen. El espacio de escala piramidal $L : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{R}$ es definido por la familia de imágenes $L_0(x, y; 0) = f$ y
>
> $$L_i(x, y; t) = G(x, y; t') * \text{SUBSAMPLE}(L_{i-1}; s), \quad (i \in \mathbb{N})$$
>
> donde $*$ es el operador de convolución, $t \in \mathbb{R}$ es el parámetro de escala, $G : \mathbb{R}^2 \times \mathbb{R}^+ - \{0\} \to \mathbb{R}$ es un *kernel* Gaussiano definido como en la Ecuación 2.1, SUBSAMPLE es el operador de submuestreo definido por
>
> $$\text{SUBSAMPLE}(f; s) = f(sx, sy), \tag{2.2}$$
>
> donde s es el período de muestreo y
>
> $$t' = \sqrt{t - t_{i-1}}, \tag{2.3}$$
>
> donde t_{i-1} es el parámetro de escala de la imagen L_{i-1}.

El período de muestreo indica que se debe tomar cada s-ésimo pixel en cada renglón y columna; usualmente es 2, de modo que el tamaño de la imagen L_i sera la mitad de la imagen L_{i-1}.

La presencia de t' se debe a que la convolución tiene un efecto acumulativo que debe tomarse en cuenta dada la naturaleza recursiva de la definición. En el caso de la convolución con Gaussianas, las desviaciones estándar se combinan siguiendo la Ecuación (Forsyth y Ponce, 2003)

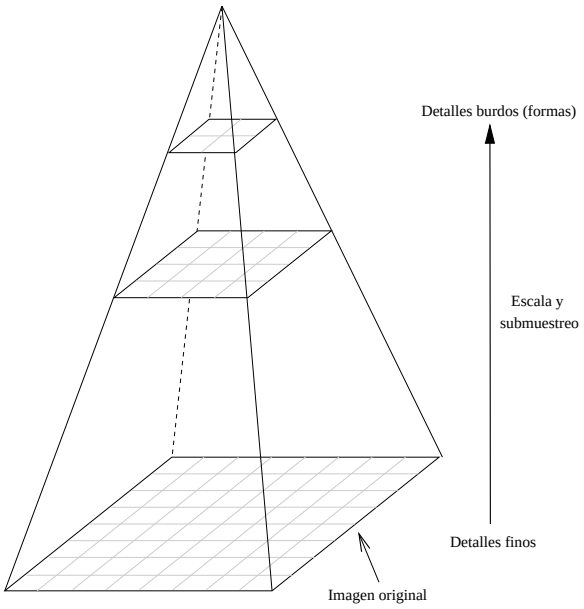


Figura 2.3: Espacio de escala piramidal o pirámide. El submuestreo reduce el tamaño de la imagen original rápidamente. (Basada en una imagen de Lindeberg (1994a)).

$$G_{\sigma_1} * G_{\sigma_2} = G_{\sqrt{\sigma_1^2+\sigma_2^2}}, \tag{2.4}$$

de la cual se deduce la definición de t'.

2.5.3. Pirámide sobremuestreada

Es una variación de la representación piramidal en la que el submuestreo no ocurre para todas las imágenes de la familia. Ello causa que la pirámide agrupe varias imágenes del mismo tamaño, tal como se muestra en la Figura 2.4.

A falta de una definición formal, no encontrada en la literatura revisada, se propone la siguiente definición derivada de aquellas para las representaciones lineal (Sección 2.5.1) y piramidal (Sección 2.5.2):

> Sea $f : \mathbb{R}^2 \to \mathbb{R}$ una imagen. La pirámide sobremuestreada $L : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{R}$ de r imágenes por bloque es definida por $L_0(x, y; 0) = f$ y
>
> $$L_i(x, y; t) = \begin{cases} i \text{ mód } r = 0, & G(x, y; t') * \text{SUBSAMPLE}(L_{i-1}; s) \\ otro\,caso, & G(x, y; t') * L_{i-1} \end{cases}$$

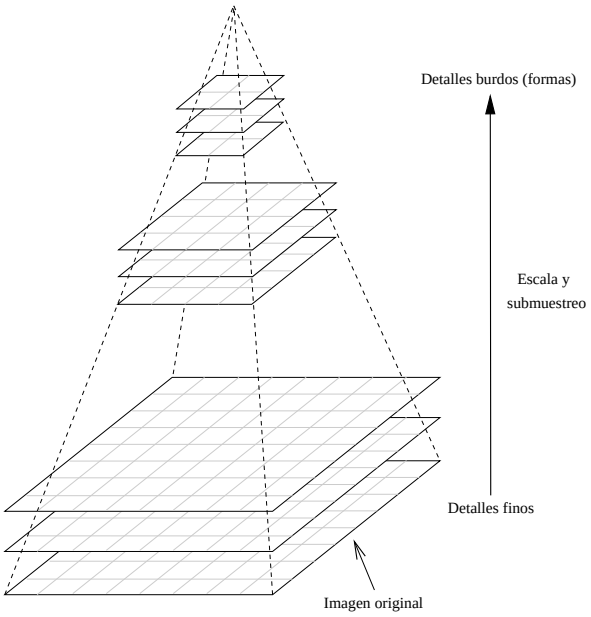


Figura 2.4: Pirámide sobremuestreada. El submuestreo se realiza cada cierto número de imágenes.

donde $i \in \mathbb{N}$, $*$ es el operador de convolución, $t \in \mathbb{R}$ es el parámetro de escala, $G : \mathbb{R}^2 \times \mathbb{R}^+ - \{0\} \to \mathbb{R}$ es un *kernel* Gaussiano definido como en la Ecuación 2.1 y SUBSAMPLE y t' se definen como en las ecuaciones 2.2 y 2.3, respectivamente.

La primera regla de la definición obliga a que el submuestreo se ejecute únicamente cuando se hayan generado r imágenes del mismo tamaño.

2.6. Algoritmo k-medios

K-medios es un algoritmo de agrupamiento que se utiliza para clasificar un conjunto de elementos en cierto número predefinido de grupos o clases. En la Figura 2.5 se muestra el diagrama de flujo del algoritmo.

Se definen k centroides, cada uno de los cuales juega el papel de elemento representativo para una clase. Después, cada elemento del conjunto es asignado a la clase del centroide más cercano de acuerdo a una métrica preestablecida; en este punto se tiene una clasificación inicial. Luego los centroides son recalculados en función de los elemen-

tos de su conjunto, de modo que reflejen al conjunto correspondiente. Estos dos últimos pasos, la clasificación y el recálculo de centroides, se repiten hasta que ningún elemento cambia de conjunto entre iteraciones o lo que es lo mismo, los centroides permanecen constantes entre iteraciones.

La métrica utilizada para determinar la cercanía entre un elemento y un centroide se puede definir en función de cualquier atributo medible. Esto provee un importante grado de flexibilidad pues permite usar k–medios para clasificar objetos diversos.

Los centroides para la iteración inicial pueden ser construidos aleatoriamente o escogidos dentro del conjunto de elementos. La elección de los centroides iniciales es importante ya que la calidad del resultado dependerá de lo representativos que sean; por ejemplo, si los centroides iniciales son muy cercanos podría resultar una distribución irregular de los elementos en las clases, clases vacías o la generación de un ciclo infinito al haber un elemento en la frontera de dos o más clases.

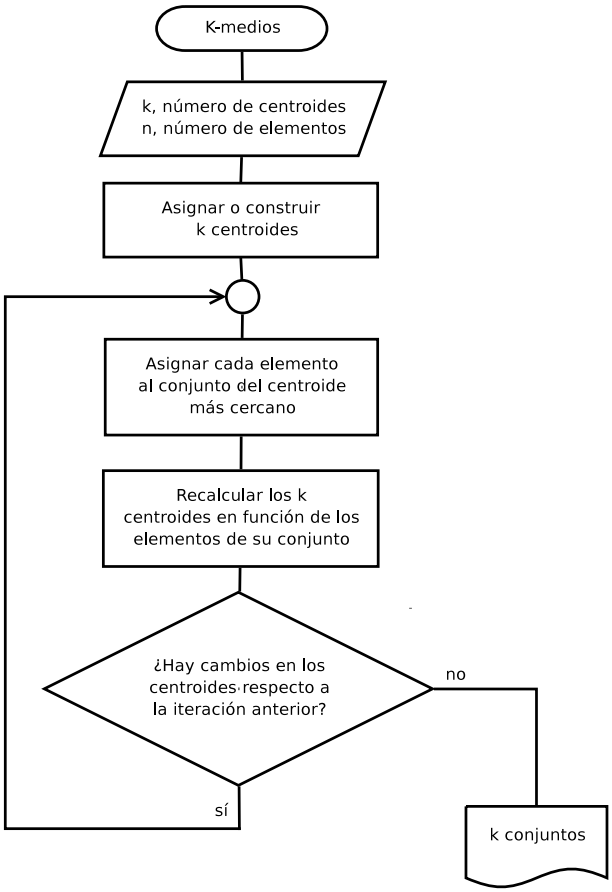


Figura 2.5: Algoritmo k-medios genérico.

Capítulo 3

Mapeo Logarítmico-Polar

El mapeo log-polar es la función inversa del mapeo exponencial, una función importante en la teoría de variable compleja. Biológicamente, es la abstracción matemática que describe el proceso de visión temprana en los primates, lo cual lo hace atractivo para su estudio y aplicación en la visión computacional.

La aplicación del mapeo log-polar a una imagen proporciona una nueva representación de ésta que posee características de invariabilidad a la escala y la rotación, además de reducir la información de la imagen original. La invariabilidad implica que los efectos del escalamiento y la rotación son neutralizados tras la aplicación del mapeo log-polar; esto quiere decir que un par de imágenes diferentes únicamente en escala u orientación tendrán esencialmente la misma representación log-polar.

Sus características de invariabilidad son deseables en el reconocimiento de patrones, sin embargo por sí solo no es una técnica de esta clase sino que únicamente provee de una representación para tal fin.

En este capítulo se presenta el modelo matemático y el modelo biológico del mapeo log-polar, así como las propiedades que lo hacen atractivo para su uso en el reconocimiento de patrones. Posteriormente se describen algunas de las implementaciones existentes, la implementación desarrollada y por último se ejemplifica la aplicación del mapeo.

3.1. Modelo matemático

Siendo el mapeo log-polar la función inversa del mapeo exponencial, resulta más sencillo explicar el primero en función del segundo. Por esta razón se presenta el mapeo exponencial y a partir de él se deduce el mapeo log-polar.

Sean Z y W los planos complejos cuyos puntos son de la forma $z = x+iy$ y $w = u+iv$, respectivamente. El mapeo exponencial es una función que toma un punto del plano Z y lo lleva al plano W mediante la regla de correspondencia

$$w = e^z, \qquad w, z \in \mathbb{C}.$$

Desarrollando

$$\begin{aligned} w &= e^z \\ &= e^{x+iy} \\ &= e^x e^{iy} \\ u + iv &= e^x \cos y + ie^x \sin y \end{aligned}$$

Para la interpretación geométrica, considérese la rejilla rectangular a la izquierda de la Figura 3.1 que representa una región del plano Z. La aplicación del mapeo exponencial deforma esta rejilla en la configuración que aparece a la derecha de la Figura 3.1, una nueva rejilla en la que los rayos y las circunferencias corresponden respectivamente a las líneas horizontales y verticales de la rejilla rectangular. Además, el área entre la primera circunferencia y el origen de la rejilla exponencial no corresponde a ningún punto de la rejilla rectangular, es decir, se trata de un área vacía.

Esta nueva rejilla ocupa por supuesto una región del plano W y como resultado del mapeo, los rayos tienen separación angular constante mientras que las circunferencias están espaciadas exponencialmente; por esta razón se le llama en adelante "rejilla exponencial".

Desde el punto de vista de las rejillas como un arreglo de celdas, los renglones de la rejilla rectangular se transforman en rayos de celdas que aumentan exponencialmente de tamaño conforme se alejan del centro en la rejilla exponencial, mientras que las columnas forman anillos con celdas de igual tamaño.

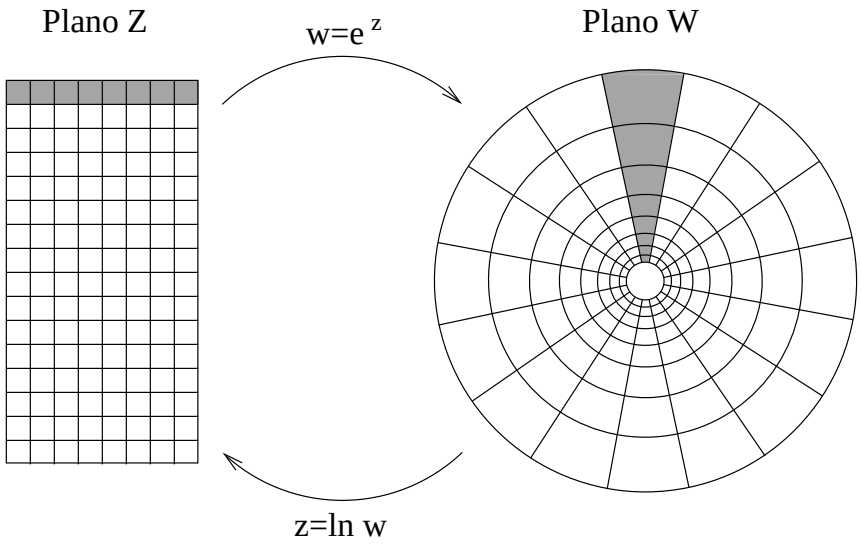


Figura 3.1: Interpretación geométrica del mapeo exponencial. A la izquierda se muestra la rejilla rectangular del plano Z y a la derecha, la rejilla exponencial que le corresponde en el plano W, resultado del mapeo exponencial. Las regiones sombreadas indican correspondencia.

El mapeo log-polar es la función inversa del mapeo exponencial y como tal, lleva un punto del plano W al plano Z mediante la regla de correspondencia

$$z = \ln w, \qquad w, z \in \mathbb{C}.$$

Desarrollando

$$\begin{aligned} z &= \ln w \\ &= \ln(u + iv) \\ &= \ln re^{i\theta}, \qquad r = \sqrt{u^2 + v^2},\ \theta = \arctan v/u \\ x + iy &= \ln r + i\theta \end{aligned}$$

La interpretación geométrica del mapeo log-polar consiste en devolver a la rejilla exponencial su forma rectangular. Esto es, dada la rejilla a la derecha de la Figura 3.1, se obtiene la rejilla mostrada a la izquierda de la Figura 3.1.

De acuerdo con estas definiciones, la rejilla rectangular corresponde al plano log-polar mientras que la rejilla exponencial corresponde al que se puede llamar plano "polar-

exponencial" —este último término no aparece en la literatura revisada sin embargo parece una nomenclatura adecuada—.

Un detalle interesante es que el mapeo exponencial toma un número complejo en forma rectangular, $x + iy$, y devuelve otro en forma polar, $re^{i\theta}$, y lo contrario ocurre en el mapeo log-polar.

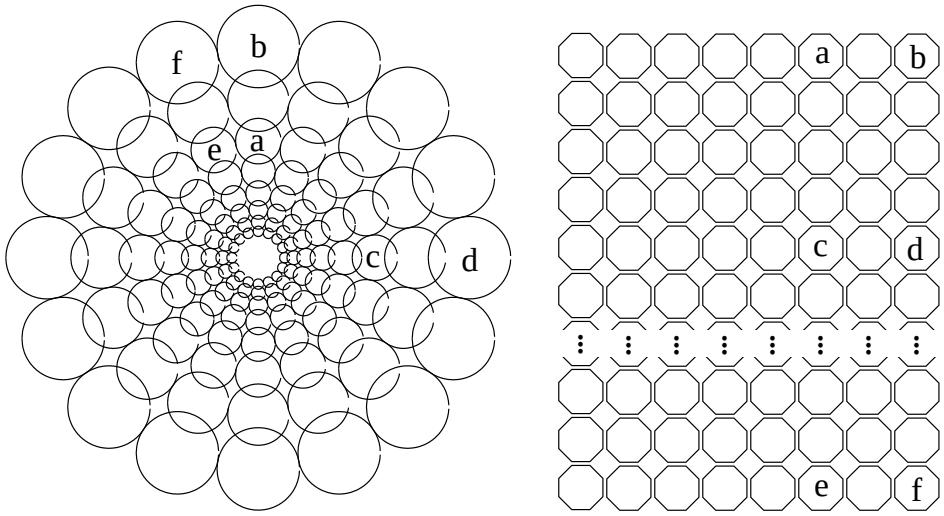
3.2. Modelo biológico

A partir del estudio del proceso de visión de los primates se dedujo un modelo que describe la transformación de los estímulos luminosos captados por la retina al ser enviados a la corteza cerebral (Wilson, 1983). Recibe el nombre de mapeo retino-cortical y la abstracción matemática que lo describe resulta ser el mapeo log-polar.

De acuerdo al modelo, la retina está formada por un conjunto de unidades receptoras (células) organizadas alrededor de un área llamada fóvea como se muestra en la Figura 3.2(a). La retina está organizada al mismo tiempo en rayos y anillos; un rayo es conformado por las unidades receptoras que se encuentran en la misma dirección angular mientras que un anillo lo es por las unidades receptoras que se encuentran a la misma distancia del centro de la retina.

Las unidades receptoras que pertenecen al mismo anillo son del mismo tamaño, no así las que pertenecen al mismo rayo. El tamaño de las unidades receptoras aumenta exponencialmente conforme se alejan del centro de la retina y de acuerdo con Wilson (1983) se traslapan en más o menos el 50 % de su área. Dada esta organización, si se sustituyen los rayos por segmentos de recta y los anillos por circunferencias concéntricas, en ambos casos atravezando los centros de las unidades receptoras, resultaría una rejilla exponencial (véase Figura 3.1).

En la corteza cerebral, un arreglo de unidades procesadoras uniformes (Figura 3.2(b)) recibe los estímulos captados por la retina de modo que existe una relación uno a uno entre éstas y las unidades receptoras. Se trata de un arreglo bidimensional en el que cada renglón está conectado a un rayo de la retina y cada columna, a un anillo; las letras en las Figuras 3.2(a) y 3.2(b) muestran esta idea. El propósito de las unidades



(a) (b)

Figura 3.2: Modelo biológico del mapeo log-polar. (a) La retina, formada por células receptoras. (b) Arreglo en la corteza cerebral formado por células procesadoras. (Ambas basadas en imágenes de Wilson (1983)).

procesadoras consiste en resumir los estímulos recibidos en un único mensaje descriptivo para luego enviarlo a otra parte del cerebro y continuar el procesamiento.

El modelo refleja una implementación biológica del mapeo log-polar en la que el arreglo de unidades procesadoras juega el papel de la rejilla rectangular y la retina, el de la rejilla exponencial mientras que la correspondencia entre dominio y contradominio se da mediante conexiones nerviosas.

3.3. Propiedades del mapeo log-polar

El mapeo log-polar se utiliza comúnmente para obtener la representación log-polar de una imagen. Esta representación, llamada imagen log-polar, es invariable a la escala y a la rotación de la imagen original. Esto significa que el mapeo log-polar permite extraer información constante acerca de aquello en la rejilla exponencial independientemente de su tamaño o su orientación. Además, la información extraída es un resumen de la información en la imagen original.

Como consecuencia, un conjunto de imágenes que difieren únicamente en escala u orientación, tendrán esencialmente la misma imagen log-polar. Son estas propiedades de

invariabilidad las que hacen deseable el uso de imágenes log-polares en el reconocimiento de patrones.

3.3.1. Reducción de información

Esta propiedad es resultado de la aplicación discreta del mapeo log-polar. Es particularmente deseable por su implicación: menos información, menos procesamiento, menos tiempo.

Para entender en que consiste la reducción de información, considérese el plano W como una señal continua bidimensional y el plano Z como el conjunto de puntos de intersección de la rejilla rectangular (véase Figura 3.1). Si a cada punto del plano Z se le asigna el valor de su imagen[1] en el plano W, se obtendrá una muestra de este último; el plano Z será entonces la representación discreta del plano W o dicho de otro modo, un "resumen" de la señal bidimensional.

Siguiendo la misma idea, si cada punto en Z es asignado con el promedio del conjunto de valores alrededor de su imagen en W, la muestra será aún más representativa. Este esquema es utilizado por Kurita *et al.* (1998) y Weiman y Chaikin (1979), así como también en el modelo biológico del mapeo log-polar.

Ahora bien, el muestreo del plano W no es uniforme; como resultado de la distribución de los puntos en la rejilla exponencial, se tiene una mayor resolución (mayor número de puntos muestreados) en el área alrededor del origen y disminuye gradualmente hacia la periferia. Esto implica una reducción de la información pero dando mayor resolución al área alrededor del origen; por esta razón el origen de la rejilla exponencial se coloca sobre el área de interés.

3.3.2. Invariabilidad a la escala

El escalamiento es una operación que aumenta o disminuye el tamaño de una imagen y da la sensación de profundidad. Sea (x, y) un punto del plano W y $(\ln r, \theta)$ su representación log-polar, donde $r = \sqrt{x^2 + y^2}$ y $\theta = \arctan y/x$. Aplicando un escalamiento de k unidades, el punto es ahora (kx, ky) y su respectiva representación log-polar es

[1] En sentido matemático.

$(\ln kr, \theta)$.

En la representación rectangular ambas componentes del punto fueron alteradas por el escalamiento mientras que en la representación log-polar, únicamente se modificó la componente del radio vector. Esto es, el mapeo log-polar "reduce" el escalamiento a un desplazamiento en la dirección del eje que representa al radio vector en la rejilla rectangular (usualmente el eje X) .

3.3.3. Invariabilidad a la rotación

La rotación es una operación que mueve una imagen alrededor de un punto, llamado centro de rotación, el cual puede estar afuera o adentro de la imagen, en cuyo caso la imagen rota alrededor de sí misma. Considérese ahora una rotación de θ_1 radianes alrededor del origen, el punto (x, y) se transforma en $(xcos\theta_1 - ysin\theta_1, xsin\theta_1 + ycos\theta_1)$ y su representación log-polar es $(\log r, \theta + \theta_1)$.

De modo similar al caso anterior, ambas componentes en la representación rectangular fueron alteradas mientras que únicamente lo hace el ángulo polar en la representación log-polar. Por tanto, el mapeo log-polar "reduce" la rotación a un desplazamiento en la dirección del eje que representa al ángulo polar en la rejilla rectangular (usualmente el eje Y).

3.3.4. ¿Qué pasa con la traslación?

Para obtener una representación log-polar, el origen de la rejilla exponencial se coloca en algún pixel de la imagen de interés (en forma implícita o explícita de acuerdo al método utilizado); llámese a este pixel, punto de aplicación. La traslación de la imagen provoca que el punto de aplicación se mueva.

Considérese una imagen I y los pixeles p y q de la misma. Si la misma rejilla exponencial se utiliza para obtener dos representaciones log-polares de I, utilizando p y q como puntos de aplicación, se obtendrán imágenes log-polares diferentes; a mayor separación entre puntos de aplicación, mayor será la diferencia entre las representaciones log-polares. Esto significa que la representación log-polar depende del punto de aplicación de la rejilla exponencial.

Para evitar este problema el origen de la rejilla exponencial debe moverse junto con el punto de aplicación o viceversa; en otras palabras, se debe nulificar el efecto de la traslación de modo que siempre se obtenga la misma representación log-polar.

3.4. Implementaciones diversas del mapeo log-polar

La aplicación del mapeo log-polar para obtener una imagen log-polar puede hacerse de varias formas. En cualquier caso, se considera que la imagen inicial ocupa una región del plano W mientras que la imagen log-polar, en el plano Z (véase Sección 3.1).

El modelo matemático ofrece dos alternativas de implementación. La primera consiste en aplicar el modelo matemático en forma directa (Wolberg y Zokai, 2000; Escobar y Ruiz-del Solar, 2002; Schindler, 2004). Los índices de pixel de la imagen original sirven como dominio para el mapeo log-polar y los índices de la imagen log-polar, como el contradominio; el valor de los pixeles de la imagen original se copia a los de la imagen log-polar utilizando el mapeo log-polar como regla de correspondencia.

La aplicación del modelo matemático en sentido inverso (Kurita *et al.*, 1998) es la otra cara de la moneda. Se utilizan los índices de pixel de la imagen log-polar como el dominio para el mapeo exponencial y los de la imagen original, como contradominio; la imagen log-polar se genera entonces aplicando el mapeo exponencial para determinar qué pixel del contradominio (la imagen original) le corresponde a cada pixel del dominio (la imagen log-polar).

Tanto en la aplicación del modelo matemático en forma directa como en la inversa, se necesita un paso intermedio de normalización debido a que el uso de las funciones exponencial y logarítmica causan que los puntos correspondientes estén fuera del rango de los índices de imagen, en el primer caso, o que los puntos estén contenidos en un rango muy pequeño, en el caso del segundo.

Una tercera forma de implementación consiste en utilizar la rejilla exponencial como una plantilla que se sobrepone a la imagen original, de modo que cada celda de la plantilla tome el valor promedio de los pixeles que ocupa en la imagen (Weiman y Chaikin, 1979). La imagen log-polar se genera copiando los valores obtenidos a una

rejilla rectangular respetando la relación renglones–rayos y columnas–anillos, propia de los mapeos exponencial y log-polar. Este enfoque es el equivalente a la aplicación del modelo biológico en el que la plantilla que se sobrepone a la imagen original es la retina descrita y el arreglo de unidades procesadoras corresponde a la imagen log-polar.

Cobos y Monasterio-Huelin (1999) implementan esta última idea pero por hardware; un sensor especial captura imágenes del mundo real directamente en forma log-polar. Esto ahorra tiempo de procesamiento al evitar la aplicación del mapeo por software.

Un problema importante es determinar el tamaño de la rejilla exponencial, a saber, el número de rayos y anillos necesarios para cubrir un área determinada; hay que recordar que el número de rayos y anillos se refleja en el número de renglones y columnas que la imagen log-polar tendrá. Además, en la práctica es útil cambiar la base del mapeo exponencial para controlar el crecimiento de la rejilla.

Cabe mencionar que el mapeo log-polar no solo puede tratarse como un operador. Belongi *et al.* (2002) desarrollaron un descriptor llamado *shape context* para especificar la forma de un objeto. Este descriptor es la rejilla rectangular que resulta de la aplicación del mapeo log-polar; ellos utilizan la rejilla exponencial correspondiente como un histograma en el que cada clase corresponde a una celda de la rejilla.

3.5. Implementación desarrollada

La implementación del mapeo log-polar consistió en reproducir el modelo biológico (véase Sección 3.2). Para ello se construye una retina que se utiliza como plantilla para muestrear la imagen original.

La retina se abstrae como un arreglo de estructuras de datos que representan a las unidades receptoras; cada una almacena una máscara de filtrado circular y su posición en la retina, la cual se calcula mediante el modelo matemático (véase Sección 3.1). Durante la aplicación del mapeo, cada unidad receptora "coloca" su máscara sobre la imagen original y calcula el valor promedio de los pixeles que la máscara cubre. Por último, estos valores son copiados a la imagen log-polar (hasta este momento vacía) respetando las relaciones rayo–renglón y anillo–columna del mapeo. Este esquema se

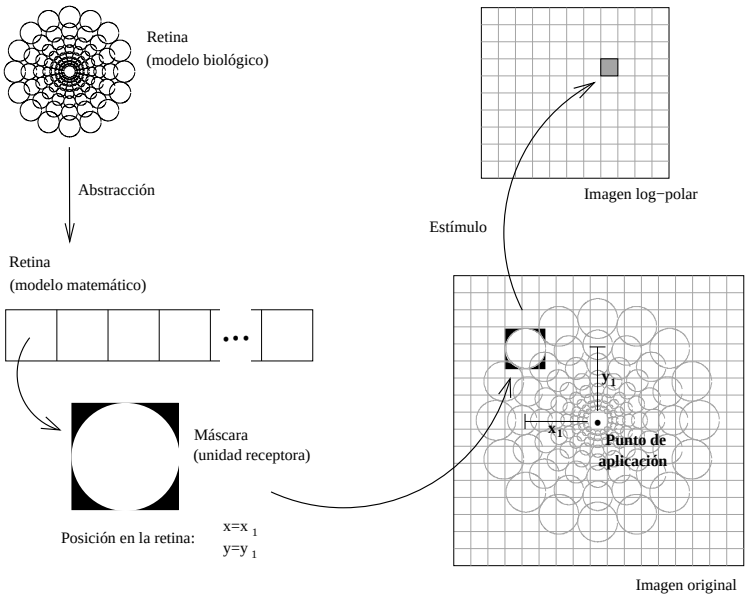


Figura 3.3: Implementación del modelo biológico del mapeo log-polar.

muestra en la Figura 3.3.

El tamaño de la retina es determinado por el área que se requiere cubrir. En función de dicha área y el número de rayos y anillos se calcula la base para utilizar en el mapeo exponencial que genere una retina del tamaño adecuado. De este modo se tiene suficiente flexibilidad para variar un parámetro sin afectar los otros; por ejemplo, utilizar el mismo número de rayos y anillos para diferentes áreas o aumentar el número de anillos para aumentar la cantidad de información.

En la siguiente sección se muestran los resultados de esta implementación ejemplificando la aplicación del mapeo log-polar.

3.6. Aplicación del mapeo log-polar

La mejor forma de apreciar las propiedades de invariabilidad del mapeo log-polar es mediante la visualización de las mismas en las imágenes log-polares. A continuación se presentan ejemplos de la aplicación del mapeo log-polar y al mismo tiempo se muestran los diferentes casos de invariabilidad, así como también el efecto de la traslación en la representación log-polar; en todos los casos se utilizó una retina de 256×256, radio de fóvea de 10 pixeles y el mismo punto de aplicación, salvo que se diga otra cosa.

(a) (b)

Figura 3.4: (a) Imagen de Lena. (b) Imagen log-polar correspondiente.

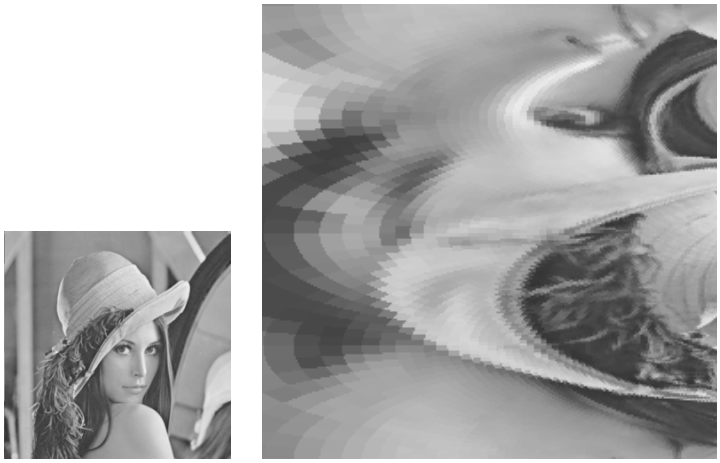
La Figura 3.4(a) muestra la imagen original utilizada para estas pruebas; ésta es usada tradicionalmente en el área de visión computacional como imagen de prueba y se le conoce como "la imagen de Lena". La Figura 3.4(b) muestra la imagen log-polar correspondiente tras la aplicación del mapeo log-polar con el punto de aplicación ubicado en el ojo derecho de Lena.

La Figura 3.5(a) muestra la imagen de Lena escalada por un factor de 0.5 y la Figura 3.5(b), su correspondiente imagen log-polar. Por efecto de la invariabilidad a la escala, la imagen log-polar sufrió un desplazamiento horizontal hacia la izquierda; en el caso cuando el factor de escalamiento es mayor que 1, el desplazamiento ocurre hacia la derecha.

La invariabilidad a la escala no solo desplaza la imagen sino que además introduce nueva información para reemplazar a la que se pierde. Obsérvese que en la imagen de la Figura 3.4(b) no aparece la boca de Lena, pero después del escalamiento, ya está presente en la imagen de la Figura 3.5(b).

La Figura 3.6(a) muestra la imagen de Lena rotada 90° y la Figura 3.6(b) su correspondiente imagen log-polar. Por efecto de la invariabilidad a la rotación, la imagen log-polar sufrió un desplazamiento vertical hacia abajo; cuando la rotación es en sentido horario, el desplazamiento ocurre hacia arriba.

Notar que a diferencia del caso anterior el desplazamiento es circular, es decir, los



(a) (b)

Figura 3.5: (a) Imagen de Lena escalada por un factor de 0.5. (b) Imagen log-polar correspondiente.

renglones que "salen" de los límites de la matriz a causa del desplazamiento son insertados en el extremo opuesto; esto implica que sin importar la rotación que la imagen sufra, la información de la imagen log-polar será siempre la misma.

Por último, en la Figura 3.7 se muestra la imagen log-polar que corresponde a la imagen de Lena tras aplicarle el mapeo con el punto de aplicación en la nariz de la modelo, lo cual equivale a una traslación. Claramente ya no coincide con las imágenes log-polares anteriores incluso si se aplicasen desplazamientos.

(a) (b)

Figura 3.6: (a) Imagen de Lena rotada 90°. (b) Imagen log-polar correspondiente.

Figura 3.7: Imagen log-polar correspondiente a la imagen de Lena con el punto de aplicación en la nariz de la modelo.

Capítulo 4

Método SIFT

Desarrollada y patentada por David G. Lowe, la transformación de rasgos invariantes a la escala o SIFT por su nombre en inglés (*scale invariant features transform*) es un método para extraer los rasgos distintivos de una imagen, llamados rasgos SIFT. Es una técnica relativamente nueva que por su rapidez y efectividad es una de las mejores hoy en día y se ha convertido en punto de referencia para evaluar nuevos métodos.

Los rasgos SIFT son vectores de datos cada uno de los cuales caracteriza una región de la imagen y por el modo en como son construidos, son invariantes a la escala, la rotación y en menor medida al cambio de punto de vista y al cambio de iluminación. En este sentido el conjunto de rasgos SIFT de una imagen conforman una firma distintiva que es deseable en el reconocimiento de patrones.

El método SIFT es también invariante a la traslación, sin embargo esta invarianza no se construye como parte del proceso sino que es consecuencia de la naturaleza de los rasgos; éstos caracterizan regiones de la imagen tomando en cuenta los valores de pixel independientemente de su posición en la imagen.

El proceso en general consta de 4 etapas:

1. Detección de extremos en espacio de escala. El espacio de escala de diferencia de Gaussianas se recorre para detectar aquellos puntos bien definidos y formar un conjunto de *keypoints* potenciales.

2. Ajuste de coordenadas. Las coordenadas discretas de cada *keypoint* potencial son interpoladas para obtener una mejor aproximación y son sometidos a validaciones

de estabilidad.

3. Asignación de orientación. Una o más orientaciones son asignadas a un *keypoint* de acuerdo a las direcciones dominantes en su vecindad, duplicando el *keypoint* por cada orientación extra.

4. Construcción de descriptores. Por cada *keypoint* se construye un vector con las medidas de gradiente en una vecindad, de tal forma que se caracteriza la región alrededor del *keypoint*.

En la primera parte de este capítulo se describe el método SIFT etapa por etapa y luego se comentan los detalles de implementación pertinentes de cada una respecto a la implementación desarrollada. Por último se muestran los resultados de su aplicación.

4.1. Descripción del método SIFT

El proceso para transformar una imagen en sus rasgos SIFT consta de 4 etapas las cuáles se describen a continuación y al mismo tiempo se sientan las bases del algoritmo general para su implementación. Cabe mencionar que la construcción del espacio de escala necesario para el proceso no es trivial por lo que se explica previamente en una sección propia.

4.1.1. Espacio de escala

La implementación más reciente del método SIFT (Lowe, 2004) utiliza una pirámide sobremuestreada como representación del espacio de escala y agrega un conjunto de restricciones a su construcción. Además, se calcula un espacio de escala de diferencia de Gaussianas en función del primero exclusivamente para la búsqueda de extremos locales y el ajuste de coordenadas.

De acuerdo a lo expuesto en la Sección 2.5.3, la familia de imágenes de una pirámide sobremuestreada se organiza en grupos de imágenes de igual tamaño que difieren en el parámetro de escala; siguiendo la nomenclatura de Lowe, se llamará *octava* a cada uno de estos grupos y en adelante se hará referencia a la pirámide sobremuestreada simplemente como “espacio de escala”.

Además, Lowe utiliza una notación ligeramente diferente a la de la teoría del espacio de escala para hacer referencia a las imágenes del espacio de escala. Formalmente, el tercer parámetro en la notación $L(x, y; t)$ indica el parámetro de escala, el cual es igual a la varianza utilizada para el *kernel* de convolución, mientras que Lowe lo utiliza para indicar la medida de escala, es decir, la desviación estándar del *kernel* de convolución. Se consideró que la notación de Lowe da mayor claridad por lo que es la que se utiliza en adelante.

Suponiendo n octavas, cada una con s imágenes, una desviación estándar inicial σ_0 y un período de muestreo de 2, las restricciones que el espacio de escala debe cumplir para servir a los propósitos del método SIFT son las siguientes:

1. La primera imagen del espacio de escala es $L(x, y; \sigma_0)$, lo cual indica que la imagen original no forma parte del espacio de escala.

2. Se generan 3 imágenes adicionales por octava por lo que en la práctica se tendrán $s + 3$ imágenes por octava.

3. La medida de escala (la desviación estándar del *kernel* de convolución) se duplica cada s imágenes. Esto es, se debe cumplir que

$$\sigma_{i,j} = 2\sigma_{i-1,j} \qquad 1 \leq i \leq n, 1 \leq j \leq s,$$

   donde i indica el índice de octava y j, el índice de imagen en la octava.

4. La desviación estándar para todo *kernel* Gaussiano es de la forma

$$k^m \sigma_0, \qquad m \in \mathbb{N} \cup \{0\}.$$

   Y de la restricción 3 se deduce que

$$k = 2^{1/s}.$$

5. La operación de submuestreo para comenzar la i-ésima octava se lleva a cabo sobre la imagen de la octava anterior que ha duplicado su medida de escala respecto a la primera imagen de dicha octava; esta imagen es siempre la primera imagen extra generada para la octava $i - 1$.

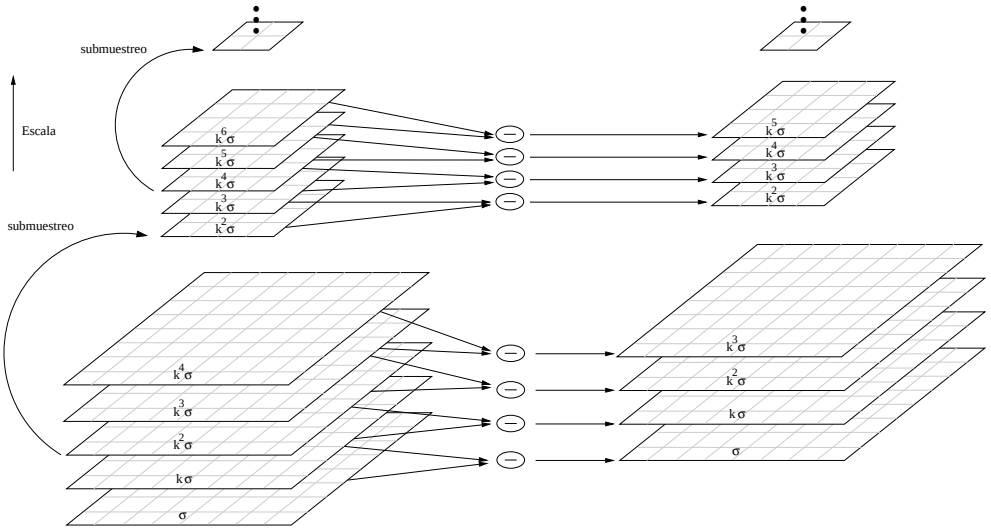


Figura 4.1: Pirámide sobremuestreada acondicionada para el método SIFT (izquierda) con $s = 2$ y el espacio de escala de diferencia de Gaussianas (derecha) que le corresponde. (Basada en una imagen de Lowe (2004)).

Con base en el cumplimiento de estas restricciones el espacio de escala resultante se muestra en el lado izquierdo de la Figura 4.1.

Como consecuencia de la restricción 2, las 3 últimas imágenes de una octava y las 3 primeras de la siguiente tendrán la misma medida de escala una a una pero diferente tamaño. El propósito de esta redundancia es cubrir una octava completa durante la búsqueda de extremos locales (Lowe, 2004).

Una vez construido el espacio de escala, se procede a restar cada par de imágenes contiguas en cada octava. El resultado será una nuevo espacio de escala conformado por diferencias de Gaussianas, con n octavas de $s + 2$ imágenes cada una; éste se muestra en el lado derecho de la Figura 4.1.

Para determinar la medida de escala que corresponde a cada imagen del espacio de escala de diferencia de Gaussianas (es decir, la desviación estándar del *kernel* de convolución) se sigue la regla dada por Lowe (2004)

$$D(x, y; \sigma) = L_1(x, y; k\sigma) - L_2(x, y; \sigma),$$

la cual indica que la diferencia de imágenes convolucionadas con Gaussianas cuyas desviaciones estándar difieren en un factor k, genera una imagen aproximada a la con-

volución de la imagen original con un *kernel* Laplaciano de Gaussiana cuya desviación estándar es igual a la menor de las originales.

4.1.2. Búsqueda de extremos locales

Inicialmente se localizan aquellos pixeles que son distintivos en la imagen, los cuales se espera que estén presentes bajo diferentes vistas del objeto. En este caso se utilizan los pixeles cuyos valores son máximos o mínimos en su vecindad, es decir, los extremos locales. Del cálculo elemental, localizar los máximos y mínimos de una función son el primer recurso para caracterizarla y bosquejar su gráfica; la misma idea aplica en este caso, donde una imagen es una función bidimensional.

La búsqueda de extremos locales se realiza sobre el espacio de escala de diferencia de Gaussianas. La presencia del espacio de escala despliega un abanico de diferentes escalas sobre las cuales buscar los extremos locales, proporcionando la invariabilidad a la escala del método SIFT.

El criterio para determinar qué puntos son extremos locales es muy directo. Un pixel es un extremo local, si y solo si, es mayor o menor que todos sus vecinos. La vecindad de comparación es tridimensional debido al espacio de escala: 8 vecinos en la imagen a la que pertenece el pixel, 9 en la imagen superior y 9 en la inferior; 26 vecinos en total.

Cada extremo local es considerado un *keypoint* en potencia y se guarda su índice de octava y la medida de escala de la imagen en que fue encontrado, así como también su posición mediante las coordenadas renglón, columna e índice de imagen en la octava.

4.1.3. Ajuste de coordenadas

La teoría define un espacio de escala con carácter continuo, lo cual es expresado por el dominio del parámetro de escala (ver Sección 2.5.3), es decir, se contemplan todas las escalas posibles. Sin embargo, en la práctica es necesario discretizar el parámetro de escala y construir un espacio de escala discreto como se ha hecho.

Hasta este punto, los extremos locales (*keypoints* potenciales) están localizados en alguna de las imágenes del espacio de escala de diferencia de Gaussianas, en algún

renglón y columna. La idea en esta etapa consiste en calcular la posición de cada extremo en el espacio de escala continuo que le corresponde, lo que significa mejorar la aproximación que representan su coordenadas discretas; las coordenadas en espacio continuo son utilizadas no solo para localizar con mayor precisión la posición del extremo local, en ocasiones denominada precisión de subpixel, sino que en función de ellas se aplican criterios de estabilidad para aceptar o rechazar un *keypoint* potencial.

Para el ajuste de coordenadas se aplica un método de interpolación desarrollado por Brown (Brown y Lowe, 2002) que intuitivamente consiste en utilizar el criterio de la primera derivada para calcular las coordenadas en espacio continuo del extremo local.

Se desarrolla la serie de Taylor hasta el término cuadrático para la función de diferencia de Gaussianas centrada en las coordenadas del extremo local $\mathbf{x} = (x, y, \sigma)$, siendo las componentes, renglón, columna e índice de la imagen en la octava; esto es

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}. \tag{4.1}$$

Derivando e igualando a cero, el extremo de $D(\mathbf{x})$ se encuentra en

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}. \tag{4.2}$$

Se utilizan diferencias de pixeles centradas en el extremo local para aproximar las derivadas y resolver la Ecuación 4.2. El punto $\hat{\mathbf{x}}$ resultante es un desplazamiento; si alguna de sus componentes es mayor que 0.5 en valor absoluto, significa que el extremo está más cerca del punto $\mathbf{x} + \hat{\mathbf{x}}$, y se repite la interpolación sobre este último. En el caso contrario, si toda componente de $\hat{\mathbf{x}}$ es menor o igual que 0.5 en valor absoluto, se considera que el extremo local en espacio continuo se ha encontrado en el punto $\mathbf{x} + \hat{\mathbf{x}}$.

Una vez obtenida la posición continua de un extremo local, se valida su contraste y su razón de curvas principales; si supera ambas pruebas, el extremo local es promovido a *keypoint*.

Validación de contraste. El valor de la función $D(\mathbf{x})$ evaluada en el extremo encontrado es útil para filtrar los extremos locales que tienen bajo contraste, el cual es un signo de inestabilidad. La condición de aceptación es:

$$D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}} \geq \text{Límite inferior de contraste}$$

donde el miembro izquierdo resulta de evaluar la Ecuación 4.1 en 4.2 y el miembro derecho es el mínimo valor de contraste aceptable.

Razón de curvas principales. También es necesario eliminar los extremos locales que la función de diferencia de Gaussianas detectó debido a la presencia de bordes mal definidos; como resultado, el extremo local tendrá una curva principal muy grande y la otra muy pequeña.

Previo a la validación se calcula la matriz Hessiana utilizando diferencias de pixeles en las coordenadas del extremo local:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}.$$

Sea r el máximo valor aceptable de la razón de curvas principales (la mayor entre la menor). Entonces, para verificar que la razón de curvas principales del extremo local es menor que r, se debe cumplir que

$$\frac{\mathsf{Tr}(\mathbf{H})^2}{\mathsf{Det}(\mathbf{H})} \leq \frac{(r+1)^2}{r}$$

donde

$$\begin{aligned} \mathsf{Tr}(\mathbf{H}) &= D_{xx} + D_{yy} \\ \mathsf{Det}(\mathbf{H}) &= D_{xx}D_{yy} - (D_{xy})^2. \end{aligned}$$

Llegado este punto, el espacio de escala de diferencia de Gaussianas ya ha cumplido su propósito y no vuelve a ser utilizado.

4.1.4. Asignación de orientación

La asignación de orientación a los *keypoints* es el primer paso para lograr la invarianza a la rotación. Se calcula tomando en cuenta las orientaciones de los pixeles en la vecindad y duplicando el *keypoint* por cada valor de orientación significativa, de modo que existirán *keypoints* con las mismas coordenadas pero con diferente orientación.

Dado un *keypoint*, se selecciona la imagen $L_{i,j}$ del espacio de escala cuya medida de escala es la más cercana a la suya. Luego, para una vecindad N_1 centrada en el *keypoint*,

se calcula la magnitud $m(x, y)$ y la orientación $\theta(x, y)$ del gradiente. Estos valores se obtienen mediante diferencias de pixeles utilizando las fórmulas:

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

$$\theta(x,y) = \arctan \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}.$$

Se construye entonces un histograma de orientación cuyas clases conforman las orientaciones posibles, las muestras son las orientaciones de los pixeles en la vecindad N_1 y las frecuencias indican tendencias de orientación para el *keypoint*. Las clases se generan dividiendo los 360° en 36 intervalos regulares, siendo las marcas de clase los puntos medios de cada intervalo. Fuera del uso convencional del histograma, el incremento de frecuencia no es unitario; éste se calcula como el producto de la magnitud del pixel muestra por la suma ponderada que resulta de aplicar una Gaussiana centrada en el pixel muestra con una desviación estándar que es 1.5 veces la medida de escala del *keypoint*.

Las clases de mayor frecuencia representan direcciones dominantes en la vecindad del *keypoint*. La elección obvia para la orientación es la clase modal; además, también son consideradas significativas las clases cuya frecuencia es mayor o igual al 80 % de la frecuencia modal por lo que se crean copias del *keypoint* para cada una de ellas.

Por último, hay que determinar cuál valor dentro de la clase asignar como orientación; para ello se ajusta una parábola con los puntos determinados por la clase y sus vecinas adyacentes, utilizando la marcas de clase como abscisas y sus respectivas frecuencias como ordenadas (véase Figura 4.2). La abscisa del máximo o mínimo de la parábola es asignada como orientación del *keypoint*.

4.1.5. Construcción de descriptores

Esta es la última parte del método SIFT. Un descriptor, también llamado rasgo SIFT, es propiamente un conjunto de histogramas de orientación que resumen la información de gradientes alrededor de un *keypoint*.

Una vez más, se selecciona la imagen $L_{i,j}$ del espacio de escala cuya medida de escala es la más cercana a la del *keypoint*. Considérese una vecindad cuadrada N_2 del

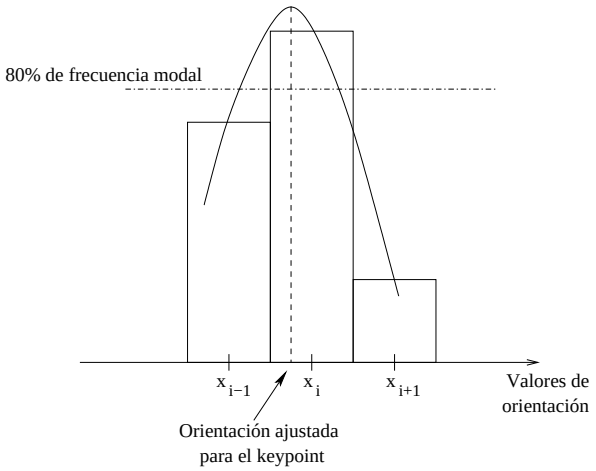


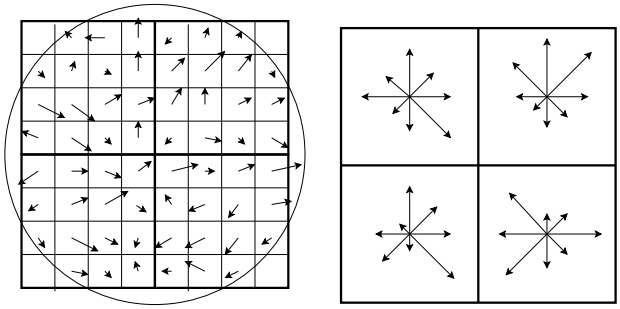
Figura 4.2: Ajuste de orientación. El valor de orientación que se asigna a un *keypoint* es la abscisa del punto máximo de la parábola ajustada.

keypoint dividida en cuadrantes de igual número de pixeles cada uno como se muestra en la Figura 4.3(a) para el caso de 4 cuadrantes de 16 pixeles cada uno. Se calcula el gradiente para cada pixel de la vecindad y se le aplican las siguientes operaciones:

1. La magnitud de cada pixel es ponderada por el valor correspondiente de una Gaussiana centrada en el *keypoint* cuya desviación estándar debe ser la mitad del ancho de la vecindad N_2 (representado por la circunferencia en la Figura 4.3(a)). Al hacer esto se asigna un peso a cada pixel en función de su cercanía al *keypoint* lo cual evita que el descriptor sufra cambios bruscos con pequeños cambios en la posición de la vecindad.

2. La orientación de cada pixel es modificada para que su valor sea relativo a la orientación del *keypoint*. Esto es particularmente importante ya que proporciona al método la invariabilidad a la rotación.

Cada cuadrante representa un conjunto de muestras con el que se construye un histograma de orientación semejante al utilizado durante la asignación de orientación; la Figura 4.3(b) muestra este resultado. A diferencia del histograma utilizado en la asignación de orientación, el incremento de frecuencia de un pixel muestra se distribuye entre la clase que le corresponde y las clases vecinas. Se pueden presentar tres casos según la relación entre la orientación del pixel muestra y la clase que le corresponde:



(a) (b)

Figura 4.3: (a) Vecindad de un *keypoint* sobre la que se construye el descriptor; las flechas representan el gradiente de los pixeles y la circunferencia la ponderación por una Gaussiana. (b) Los gradientes de cada cuadrante son resumidos en un histograma de orientación; cada flecha representa una clase del histograma y su longitud, el valor de frecuencia. (Ambas basadas en imágenes de Lowe (2004)).

- La orientación del pixel muestra es igual a la marca de clase: las clases vecinas no son afectadas.
- La orientación es menor que la marca de clase: la clase anterior es afectada.
- La orientación es mayor que la marca de clase: la clase siguiente es afectada.

El incremento de frecuencia es el producto de la magnitud del pixel muestra (ponderado por la Gaussiana) por un factor que varía en función de la clase que se va a afectar. Este último es igual a $(1-d)^3$, donde d es la distancia entre el valor de orientación del pixel muestra y la marca de clase, dividido por el tamaño de las clases.

Se genera entonces un vector cuyos elementos son los valores de frecuencia de las clases de todos los histogramas; este vector es el descriptor que corresponde al *keypoint*. Su tamaño está en función del número de histogramas (cuadrantes) y el número de clases en que se dividen éstos (el mismo para todos). Por ejemplo, con 4 histogramas de 8 clases cada uno, se tendrían $4 \times 8 = 32$ elementos para el descriptor.

Por último, para disminuir los efectos debidos a los cambios de iluminación, se normaliza el vector y se verifica que todo elemento sea menor o igual a 0.2; los elementos mayores son truncados y se normaliza una vez más.

4.2. Detalles de Implementación

En esta sección se describen los detalles que son específicos de la implementación desarrollada con el fin hacerla repetible.

4.2.1. Construcción del espacio de escala

Aunque no es parte del algoritmo formal, Lowe (2004) comenta que se asume que la imagen original tiene un difuminado como si hubiera sido convolucionada con una Gaussiana con una desviación estándar de 0.5. Esto significa que la imagen original no se convoluciona con σ_0, sino con una desviación estándar ligeramente más pequeña que debe cumplir la regla dada por la Ecuación 2.4, de tal modo que la base de la pirámide tenga una medida de escala igual a σ_0.

También menciona que con el propósito de aprovechar toda la información de la imagen original, ésta es duplicada en tamaño y es a partir de esta última que se construye el espacio de escala. Para esta implementación se omitió este paso por simplicidad ya que tampoco forma parte del algoritmo formal descrito por Lowe.

4.2.2. Búsqueda de extremos locales

Con el fin de incluir en la búsqueda a los pixeles de los bordes de la imagen, se agrega un borde falso a todas las imágenes en ambos espacios de escala. Los pixeles del borde falso son asignados con el valor promedio de los pixeles de la imagen original. Sin embargo, la primera y última imagen de cada octava no son tomadas en cuenta en la búsqueda ya que ninguno de sus pixeles tiene vecindad tridimensional.

4.2.3. Ajuste de coordenadas

Al llevar a cabo las primeras pruebas de la implementación se observó que en varios casos los valores interpolados de $\hat{\mathbf{x}}$ de la Ecuación 4.2 fluctúan en el rango de los números reales. Para evitar los índices fuera de rango de la imagen o de la octava, se agregó validación para estas situaciones en cuyo caso, el extremo en turno es eliminado.

Además, con el fin de evitar posibles ciclos infinitos causados por la reinterpolación

continua del mismo extremo, se agregó un número máximo de reinterpolaciones; pasado este número, se fuerza a tomar el siguiente extremo local y el extremo en turno es eliminado.

4.2.4. Asignación de orientación

La magnitud y la orientación del gradiente son calculados para todos los pixeles de todas las imágenes del espacio de escala; esto se hace previamente, de modo que estén disponibles en el instante en que se necesitan, tanto para la asignación de orientación como para la construcción de descriptores.

Respecto al ajuste de la parábola para determinar la mejor aproximación del valor de orientación, se dan dos casos especiales que se deben prever. El primero se da cuando los puntos a interpolar son colineales en cuyo caso la marca de clase es tomada como el valor de orientación. El segundo ocurre cuando la abscisa del extremo (sea máximo o mínimo) se encuentra fuera del intervalo válido de orientación; en este caso se utiliza el carácter cíclico de la medida de ángulos para determinar el ángulo equivalente dentro del intervalo válido de orientación.

4.2.5. Construcción de descriptores

Considérese nuevamente la Figura 4.3(a); en ella el *keypoint* es representado como un *punto* en un plano continuo, sin embargo debido al aspecto discreto de las imágenes, en la práctica un *keypoint* está necesariamente ligado a un pixel.

La solución a este problema fue solapar los cuadrantes que comparten un renglón o columna con el *keypoint*; en la Figura 4.4 se muestran dos casos de la aplicación de este esquema. Una alternativa es ignorar el renglón y la columna redundantes como parte de la vecindad, sin embargo ello implica la pérdida de información.

Se encontró que para garantizar que la vecindad N_2 esté centrada en el *keypoint* bajo este esquema de solapamiento se deben cumplir las siguientes condiciones:

1. El número de cuadrantes en la vecindad, equivalente al número de histogramas por descriptor, debe ser par y tener raíz cuadrada exacta.

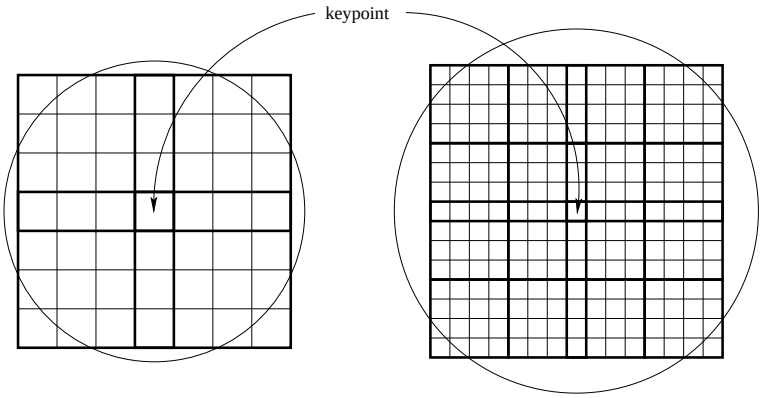


Figura 4.4: Vecindad de un *keypoint* sobre la cual se construye un descriptor tomando en cuenta el aspecto discreto de la imagen digital. A la izquierda, un descriptor de 4 cuadrantes con 16 pixeles cada uno; a la derecha, un descriptor de 16 cuadrantes con 16 pixeles cada uno.

2. E número de pixeles por cuadrante, equivalente al número de pixeles muestra por histograma, debe tener raíz cuadrada exacta.

4.3. Resultados de la implementación

Los rasgos SIFT son vectores de valores que no tienen sentido por sí solos. Por otro lado, los *keypoints* muestran los pixeles en la imagen susceptibles a ser detectados en un sistema de reconocimiento por lo que tienen ubicación, magnitud y orientación.

Con el propósito de mostrar el funcionamiento de la implementación, se aplicó el método SIFT a la imagen de la Figura 4.5(a) y se graficaron los *keypoints* encontrados en la Figura 4.5(b). Ahora compárese con la Figura 4.5(d) la cual muestra el resultado de la aplicación de una implementación desarrollada por Rob Hess[1]; como se puede observar, la implementación de Hess encontró muchos más *keypoints* que la implementación desarrollada, sin embargo existen coincidencias entre ambos resultados.

Cabe mencionar que ambas implementaciones fueron ejecutadas con los mismos parámetros y éstos permanecen para todos los ejemplos y experimentos, salvo se diga otra cosa; los valores de parámetros se listan en el Cuadro 4.1. En particular, la implementación de Hess no requiere del número de octavas y no permite modificar

[1] El código es proporcionado en un CD anexado a este documento y también puede descargarse desde `web.engr.oregonstate.edu/~hess`.

Parámetro	**Valor**
Número de octavas (n)	5
Niveles por octava (s)	3
Desviación estándar inicial (σ_0)	1.6
Máximo num. de interpolaciones por *keypoint*	5
Límite mínimo de contraste	3 % del máx. valor en rango dinámico
Razón de curvas principales máximo	10
Radio de la vecindad N_1	5 pixeles
Clases en el histograma de orientación	36
Histogramas por descriptor	16
pixeles muestra por histograma de descriptor	16
Clases por histograma de descriptor	8

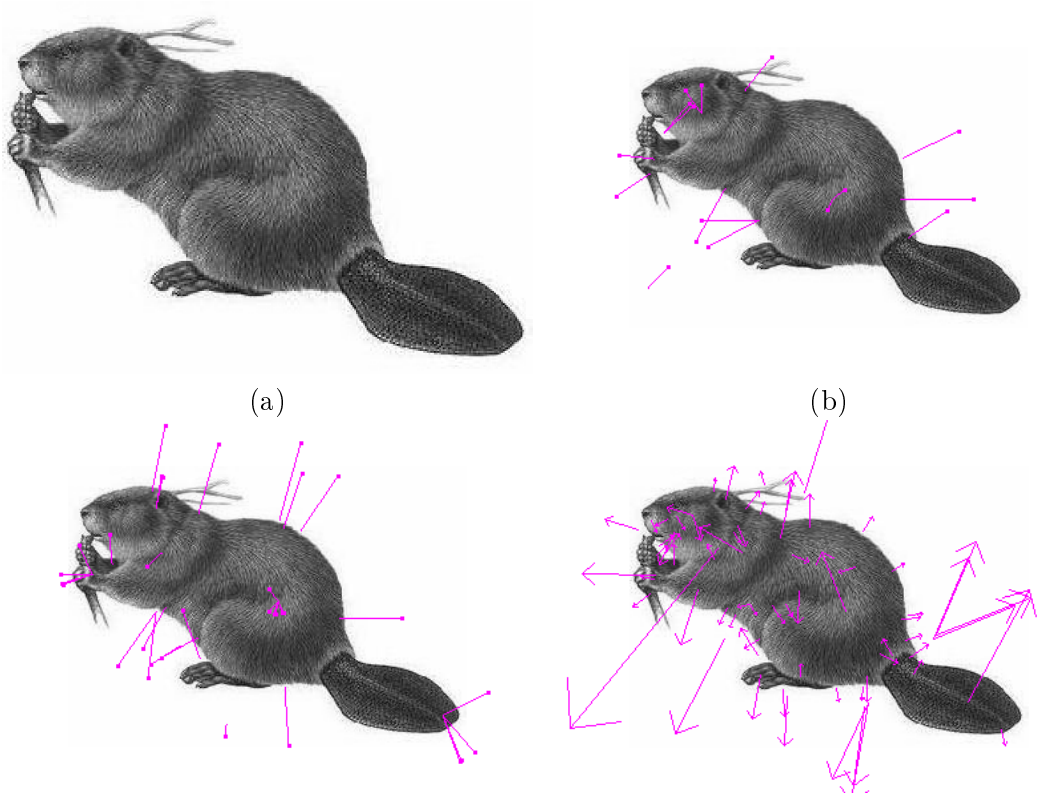
Cuadro 4.1: Valores de parámetros para la aplicación del método SIFT.

directamente el número de pixeles muestra por histograma de descriptor.

Ahora bien, la Figura 4.5(c) es resultado de la implementación desarrollada aplicando un pequeño cambio al parámetro σ_0. Claramente se obtuvieron resultados diferentes a los de la Figura 4.5(a). Sin embargo, ambas imágenes tienen coincidencias entre sí y con la Figura 4.5(d) e incluso se encontraron *keypoints* en una que no se encontraron en la otra y sin embargo, sí se encuentran en los resultados de la implementación de Hess. Por otro lado, también se encontraron *keypoints* que la implementación de Hess no detectó; esto lleva a pensar que la implementación desarrollada detecta *keypoints* falsos.

Se realizaron pruebas variando los parámetros n, s y σ y en general el comportamiento descrito prevaleció. La implementación de Hess demostró ser mucho más constante aunque sí tuvo pequeñas variaciones.

A pesar de ello, la implementación desarrollada permite demostrar las propiedades de invariabilidad del método SIFT, aunque fue necesario encontrar un conjunto de parámetros adecuado. En la parte superior de la Figura 4.6 se muestra este resultado; se puede observar que existen coincidencias de *keypoints* en todas las instancias de la letra, sin embargo es clara la superioridad de los resultados obtenidos con la implementación de Hess mostrados en la parte inferior de la misma figura.



(a) (b) (c) (d)

Figura 4.5: (a) Imagen original. (b) Resultado de la implementación desarrollada con $n = 5$, $s = 3$ y $\sigma_0 = 1.6$. (c) Resultado de la implementación desarrollada con $n = 5$, $s = 3$ y $\sigma_0 = 1.5$. (d) Resultado de la implementación de Hess con $s = 3$ y $\sigma_0 = 1.6$

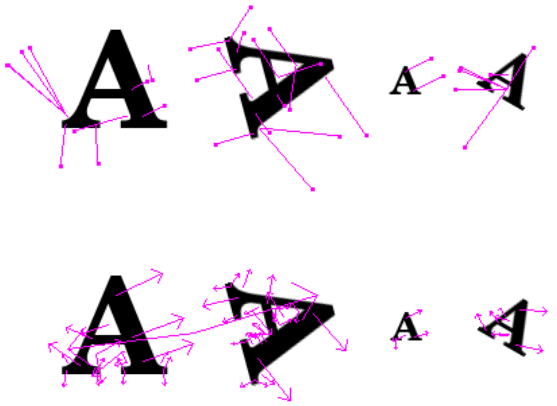


Figura 4.6: Demostración de las propiedades de invariabilidad del método SIFT. La línea superior muestra los resultados de la implementación desarrollada y la inferior, los de la implementación de Hess. En ambos casos se utilizaron los parámetros $n = 4$ (Hess no requiere este parámetro), $s = 3$ y $\sigma_0 = 1.2$.

Capítulo 5

Experimentos y Resultados

Con el fin de demostrar el desempeño tanto del mapeo log-polar como del método SIFT en el reconocimiento de patrones se diseñaron e implementaron métodos sencillos de reconocimiento que hicieran uso de ellas.

Los patrones para el reconocimiento en este caso es el conjunto de letras del alfabeto castellano, tanto minúsculas como mayúsculas con la excepción de la letra "ñ", sujetas a escalas y rotaciones aleatorias.

El método de reconocimiento que se implementó para el mapeo log-polar se basa en una adaptación del algoritmo k–medios para aprovechar las características de invariabilidad de las imágenes log-polares.

Para el reconocimiento utilizando el método SIFT se implementó un sistema de base de datos de rasgos muy sencillo cuyas búsquedas de coincidencia son exhaustivas.

El análisis de los resultados se hizo de forma cualitativa mediante el empleo de gráficas en el caso del mapeo log-polar. El reconocimiento mediante el método SIFT consistió en una demostración de la identificación de *keypoints* en forma gráfica; el resultado de la aplicación ilustra de forma simple la invariabilidad del método a la escala y a la rotación a pesar de los problemas que la implementación del método SIFT presentó.

En este capítulo se describen los métodos de reconocimiento implementados para mostrar el desempeño del mapeo log-polar y el método SIFT; para cada uno se presenta la descripción de los experimentos realizados, sus condiciones de ejecución y los resultados obtenidos.

5.1. Reconocimiento de patrones utilizando mapeo log-polar

Considérese un conjunto de letras diferentes en una imagen con varias instancias de la misma letra. El objetivo es dividir el conjunto inicial en subconjuntos o clases de tal modo que cada clase esté formada por las instancias de la misma letra.

5.1.1. Descripción del método

La imagen de prueba es segmentada para identificar cada letra como el conjunto de pixeles que la conforman. La segmentación es una operación común en el procesamiento de imágenes cuyo objetivo es separar la imagen en regiones en función del valor de los pixeles; para más detalle se recomienda consultar cualquier libro sobre procesamiento de imágenes o visión computacional.

Se aplica el mapeo log-polar a cada letra utilizando su centro de masa como punto de aplicación; dado que el centro de masa es invariable, se garantiza que el punto de aplicación será siempre el mismo respecto a la letra sin importar su tamaño, orientación o posición en la imagen. El número de rayos y anillos y el tamaño de la fóvea son pasados a la implementación como parámetros; falta determinar el tamaño de la retina, sin embargo por el momento se pasa por alto y se retoma en la siguiente sección.

Inicialmente, existe una sola clase formada por todas las letras. El proceso de división ocurre en forma recursiva, siempre por el mismo número de subclases, hasta que las clases quedan vacías, tengan solo un elemento o ya no puedan ser subdivididas. Por ejemplo, con división en 3 subclases, la clase inicial es dividida en 3 clases, luego cada una de estas es dividida en 3 clases (9 clases en total), éstas a su vez son divididas en 3 (27 clases en total) y así sucesivamente. Conceptualmente se genera un árbol balanceado en el que cada nodo representa una clase, los descendientes representan el refinamiento del nodo padre y las hojas son las clases finales.

La división de clases está a cargo del algoritmo k–medios (véase Sección 2.6) utilizando las imágenes log-polares como vectores multidimensionales. Esto es, las imágenes log-polares representan vectores cuya dimensión es igual al producto del número de rayos por el número de anillos y cada pixel es un componente del vector que representan. El

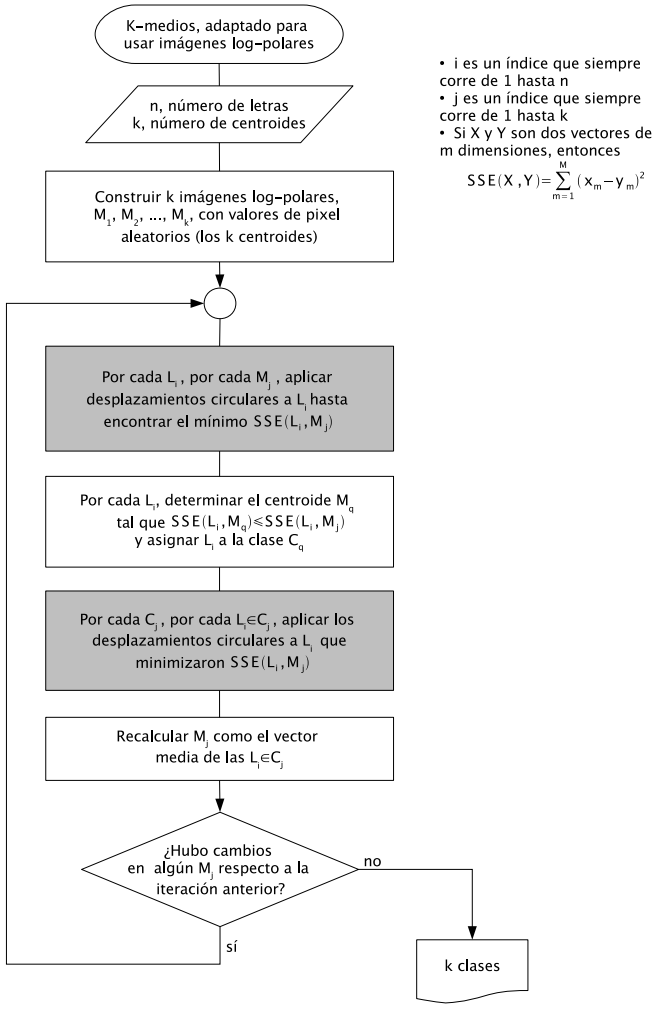


Figura 5.1: Algoritmo k-medios adaptado para aprovechar las características de invariabilidad del mapeo log-polar.

número de subclases en que se divide una clase corresponde al número de centroides para k–medios.

En la Figura 5.1 se muestra el diagrama de flujo del algoritmo k–medios adaptado para usar imágenes log-polares, lo que significa aprovechar sus características de invariabilidad para la clasificación. La métrica utilizada es la suma de diferencias al cuadrado de los pixeles o SSE por sus siglas en inglés (*sum of squared errors*). Los bloques sombreados indican las operaciones agregadas con el propósito de aprovechar las características de invariabilidad de las imágenes log-polares.

En el primer bloque sombreado de la Figura 5.1, de acuerdo al flujo del diagrama,

se busca la distancia entre la imagen log-polar de la letra en turno y cada uno de los centroides. Para ello se aplica cada posible desplazamiento vertical y horizontal a la imagen log-polar en turno y se calcula el SSE entre la imagen desplazada y un centroide; el mínimo de estos valores corresponderá a la distancia entre la letra y el centroide. La operación se repite para cada centroide y la menor de las distancias encontradas indicará la clase a la cual pertenece la letra. Ésto se repite a su vez para cada letra con el fin de clasificarlas.

Recuérdese que el desplazamiento en sentido horizontal de una imagen log-polar es el efecto de la invariabilidad a la escala del mapeo log-polar, así como el desplazamiento vertical lo es de la invariabilidad a la rotación. Entonces, el propósito del procedimiento descrito en el párrafo anterior consiste en comparar todas las posibles escalas y orientaciones de la letra con un centroide hasta encontrar el par que maximice su semejanza; cabe aclarar que la frase "todas las posibles escalas y orientaciones" hace referencia a todas aquellas que la representación log-polar admita.

Los desplazamientos aplicados son circulares unitarios; por ejemplo, si el desplazamiento ocurre hacia la derecha, las columnas se mueven un lugar a la derecha y la columna de más a la derecha se coloca en el extremo izquierdo; la misma idea se aplica cuando el movimiento es hacia la izquierda. Si el desplazamiento ocurre hacia arriba, los renglones se mueven un lugar hacia arriba y el renglón de más arriba se coloca en el extremo inferior y análogamente para el desplazamiento hacia abajo.

El segundo bloque sombreado de la Figura 5.1 es una forma de normalización. Las imágenes log-polares son desplazadas para maximizar su semejanza al centroide de la clase a la que pertenecen utilizando los desplazamientos que minimizaron el SSE (ya encontrados en un paso anterior). Luego se recalculan los centroides en función de las imágenes log-polares desplazadas; con ello se pretende encontrar el centroide más representativo de la clase para la siguiente iteración.

Como productos de la ejecución del algoritmo se generan dos archivos de datos; uno de ellos lista el conjunto de clases, cada una con las letras pertenecientes a ella y el otro es un archivo con la misma información pero dispuesta para ser graficada con el

Figura 5.2: Fragmento de la imagen de prueba con letras rotadas y escaladas para el reconocimiento utilizando mapeo log-polar.

programa Ggobi[1].

5.1.2. Condiciones de experimentación

Dado el conjunto de letras del alfabeto castellano, con la única omisión de la letra "ñ", y considerando las minúsculas y mayúsculas, se utilizaron 52 letras diferentes para los experimentos realizados.

La letra "ñ" fue eliminada del conjunto para reducir la complejidad; debido a que la virgulilla no está conectada a la letra, el proceso de segmentación separa los grafos eliminando su sentido como conjunto. Por la misma razón, se eliminaron los puntos de las letras "i" y "j"; cabe notar que eliminar la virgulilla de la "ñ" la hace indistinguible de la "n", lo que no es el caso con las anteriores.

Las imágenes de prueba fueron generadas utilizando la fuente *Century Schoolbook L Bold*; se generaron 4 imágenes de prueba con características particulares, cada una con 260 letras (5 instancias por letra):

1. Imagen de control. Instancias idénticas de cada letra.
2. Con letras rotadas. A cada instancia se le aplicó una rotación aleatoria en el rango $[0, 2\pi)$.
3. Con letras escaladas. A cada instancia se le aplicó un factor de escala aleatorio en el rango $[0.33, 1.99]$.

[1] Software libre para visualización de datos multidimensionales, descargable desde `www.ggobi.org`.

4. Con letras rotadas y escaladas (Figura 5.2). A cada instancia se le aplicó una rotación aleatoria en el rango $[0, 2\pi)$ y también un factor de escala aleatorio en el rango $[0.33, 1.99]$.

Un experimento consistió en la ejecución de la aplicación de reconocimiento y se llevaron a cabo 4 series de experimentos, una por cada imagen de prueba. Cada serie consistió a su vez en la ejecución de la aplicación cambiando el parámetro k del algoritmo k–medios, es decir, el número de subclases para el proceso de división. Los valores de prueba para k fueron 2, 4, 6, 8, 10, 13, 26 y 52 y se utilizó una retina de 30×30 con un radio de fóvea de 3 pixeles. Solo falta determinar el área que la retina debe ocupar, es decir, su tamaño; para observar el efecto del tamaño de la retina en el reconocimiento se probaron dos estrategias:

- Tamaño fijo, de modo que ocupe el área más pequeña capaz de contener la letra más grande encontrada durante la segmentación.
- Tamaño variable, de modo que la retina ocupe el área más pequeña que contenga a la letra en turno.

Cada estrategia implica la ejecución de las 4 series de experimentos como se describió y se espera que la segunda estrategia mejore los resultados del reconocimiento respecto a la primera.

5.1.3. Resultados del reconocimiento de patrones utilizando mapeo log-polar

Para graficar, se asignó un número identificador a cada letra (el mismo para todas las instancias), así como a cada clase; el eje horizontal corresponde a las letras y el vertical, a las clases, de modo que un punto del plano representa a una letra y la clase a la que pertenece. También se asignó un color a las letras que pertenecen a la misma clase, aunque existen repeticiones debido a que solo habían 12 colores disponibles. En el caso de los experimentos con las imágenes de control, se indicó en las gráficas qué letra representa cada punto; esto fue un caso especial debido a que la distribución de los puntos permite apreciar las letras, lo que no ocurre en el resto de las gráficas.

Estrictamente, un error de clasificación ocurre cuando existen varios puntos sobre la misma línea horizontal o sobre la misma línea vertical. En el primer caso se trata de diferentes letras que fueron clasificadas juntas, mientras que en el segundo, instancias de la misma letra están en diferentes clases. El reconocimiento ideal de las letras debe generar 52 puntos en la gráfica alineados sobre la recta identidad, lo cual indica que cada letra tiene su propia clase y que todas las instancias de la misma pertenecen a ella.

Para cada imagen de prueba se presenta el mejor caso; cabe mencionar que el peor caso para todos los experimentos fue siempre cuando $k = 2$, resultando en clases grandes de más 10 letras diferentes. El estatus de calidad de las gráficas se basó en sus características ya que no se definieron medidas de desempeño; en otras palabras, es un análisis cualitativo sujeto a subjetividad.

Resultados utilizando una retina con tamaño fijo

Los resultados de la serie de experimentos con la imagen de control son desde luego los mejores. Dado que todas las instancias en la imagen de prueba son constantes en tamaño y orientación, k–medios logra clasificar con gran efectividad. A pesar de esto, errores ocurrieron pero se pueden atribuir a la discretización de las imágenes.

En la Figura 5.3 se muestra el mejor caso obtenido con la imagen de control. Se puede observar una recta casi ideal con 42 clases. Aunque existen clases con letras diferentes, algunas de ellas son coherentes, por ejemplo "*bh*", "*CcUe*", "*RB*" y en particular, "*pd*", la cual es un claro indicio de invariabilidad a la rotación. El error más severo ocurrió con la clase "*liy*" donde sobra la letra "*y*".

Los resultados de los experimentos con la imagen de letras rotadas fueron los siguientes en eficacia; véase la Figura 5.4. Se puede ver aún una línea con constancia y muy pocos puntos se salieron de la forma principal de la gráfica. Esto indica que hubo un buen reconocimiento de las letras a pesar de las rotaciones.

Nótese que existen puntos alineados verticalmente tan pegados que se solapan. Estas clases se forman porque en cada iteración de k–medios, las instancias de una misma letra son colocadas en la misma subclase hasta alcanzar su estado óptimo, cuando son los

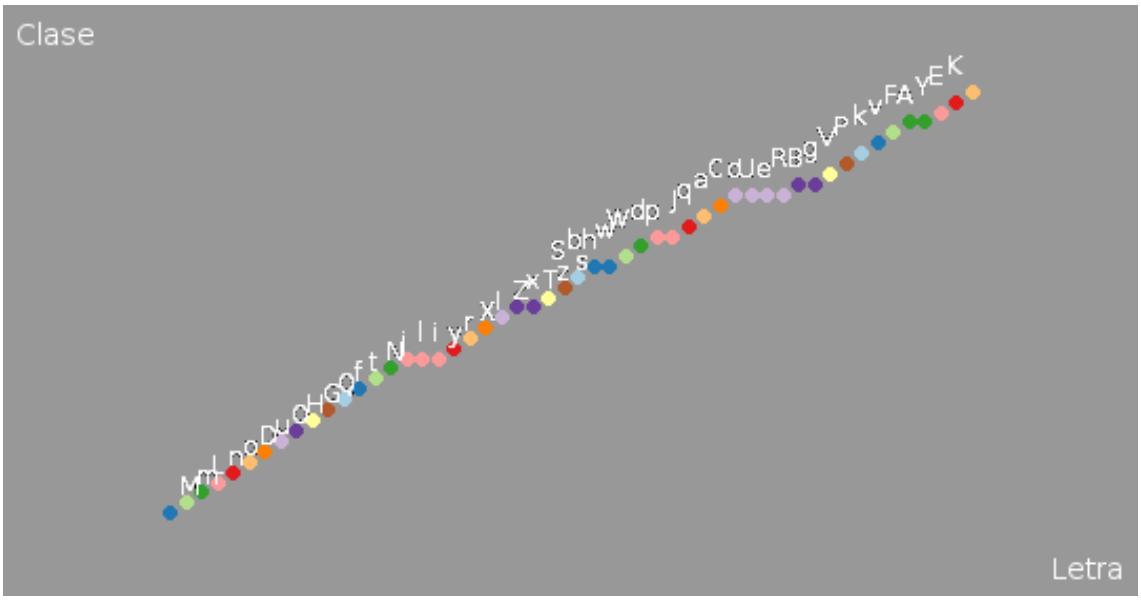


Figura 5.3: Mejor caso ($k = 10$) de los resultados obtenidos con la imagen de control y una retina de tamaño fijo.

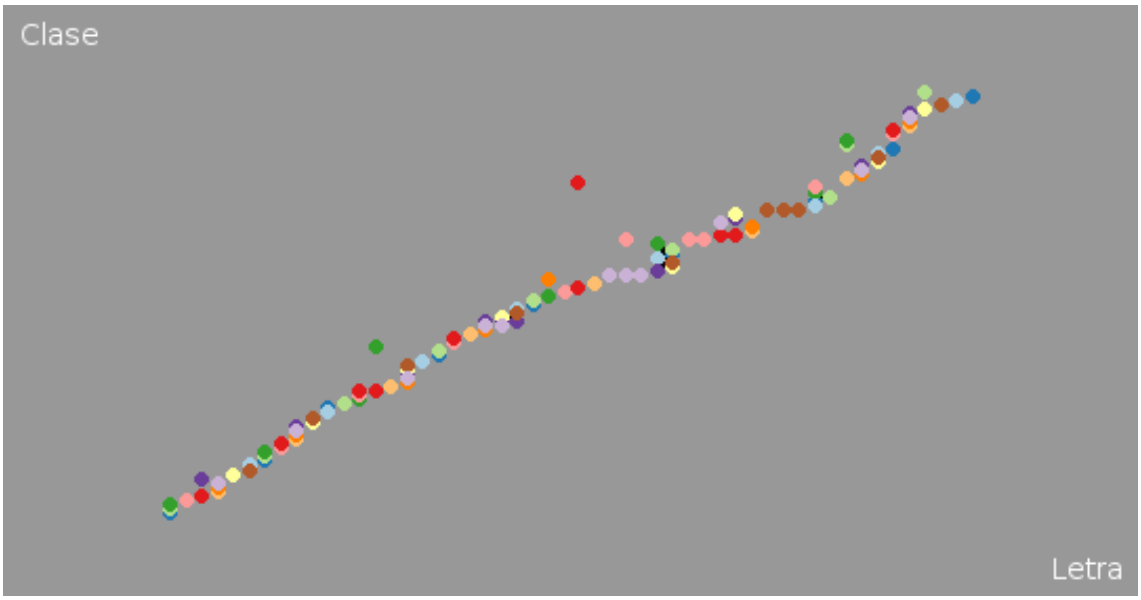


Figura 5.4: Mejor caso ($k = 26$) de los resultados obtenidos con la imagen de letras rotadas y una retina de tamaño fijo.

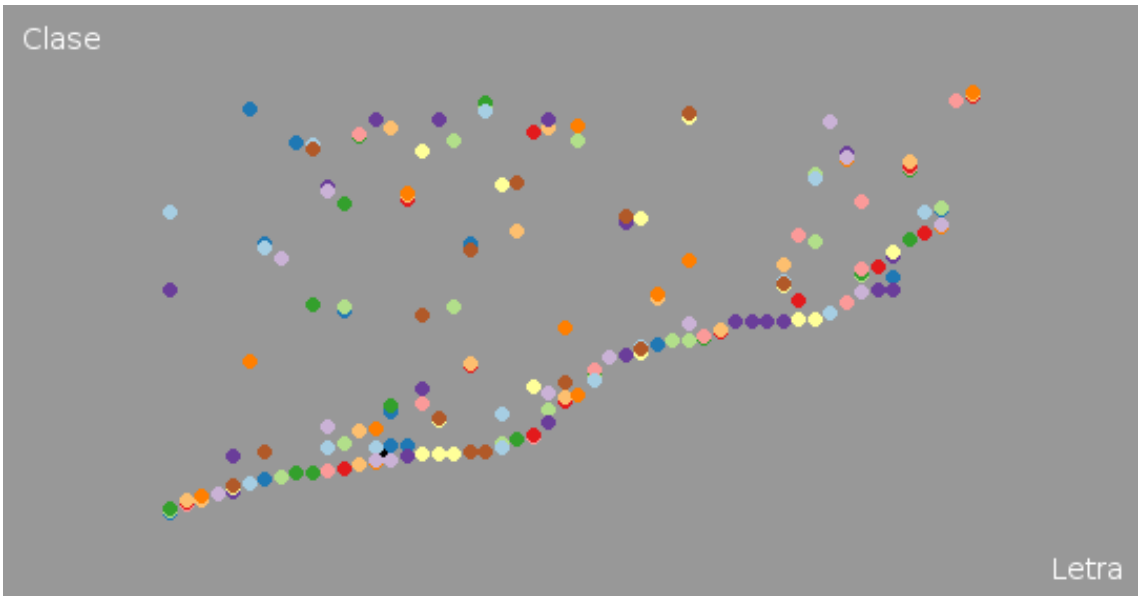


Figura 5.5: Mejor caso ($k = 52$) de los resultados obtenidos con la imagen de letras escaladas y retina de tamaño fijo.

únicos elementos de la clase. Sin embargo, el algoritmo no "sabe" que ya ha optimizado la clase y continúa la subdivisión, forzando a la separación de las instancias. Esto solo es posible si las representaciones log-polares son significativamente similares, lo cual confirma la invariabilidad a la rotación del mapeo log-polar.

El mejor caso (Figura 5.5) para los experimentos con la imagen de letras escaladas ocurrió cuando $k = 52$ generando 151 clases, algunas de las cuales fueron clases óptimas o casi óptimas en algún momento como lo indican los puntos solapados alineados verticalmente; además, se puede observar que no hay puntos alineados del mismo color entre los que están fuera de la forma principal. Esto podría llevar a pensar que hubo un buen reconocimiento, pero éste no solo consiste en separar las letras que son diferentes sino que además, se trata de juntar todas las instancias de la misma letra en la misma clase; luego, el reconocimiento fue muy pobre en este experimento.

Los puntos que están alineados verticalmente sin solaparse representan instancias de una letra que fueron separadas por k–medios en forma prematura; es decir, las instancias se separaron durante las primeras iteraciones de k–medios eliminando la posibilidad de pertenecer a la misma clase. Esto solo ocurre si las representaciones log-polares de las instancias de la misma letra son significativamente diferentes; y entonces, dado que

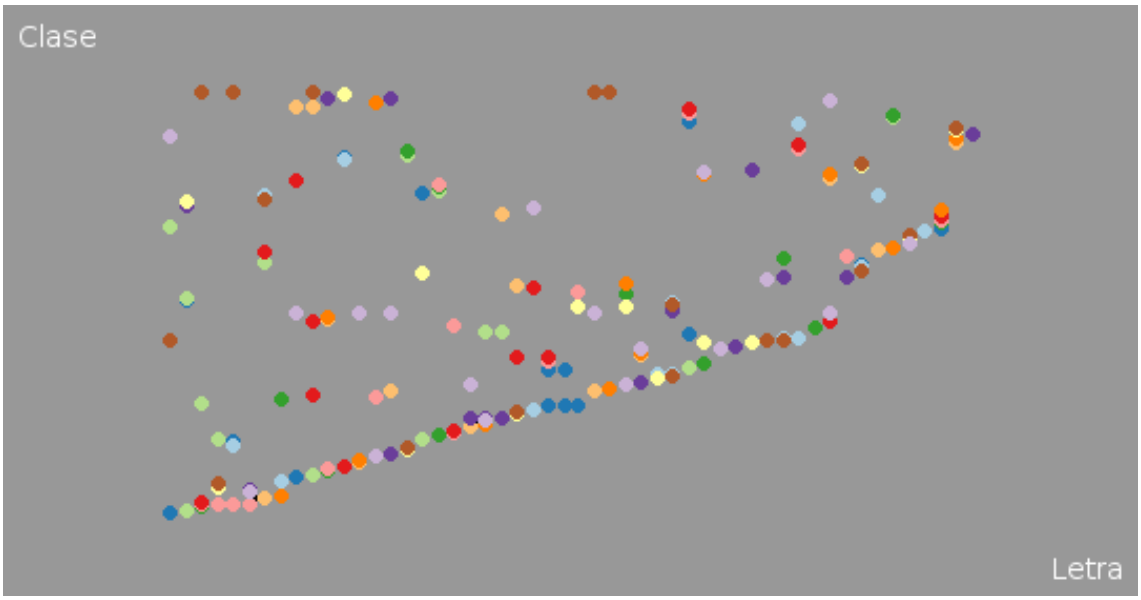


Figura 5.6: Mejor caso ($k = 52$) de los resultados obtenidos con la imagen de letras rotadas y escaladas y retina de tamaño fijo.

la escala es el único componente extra, se deduce que la escala está perjudicando el reconocimiento a pesar de que matemáticamente se sabe que el mapeo log-polar es invariable a la escala.

El problema está en el uso de la misma retina para todas las letras; debido al efecto del escalamiento, si las letras son muy pequeñas en comparación con la retina, gran parte de la información de la letra caerá sobre la fóvea, donde el mapeo log-polar no tiene efecto alguno y como consecuencia, la imagen log-polar no será representativa de la letra. Entonces, al hacer la comparación entre imágenes log-polares de dos instancias de la misma letra, una de las cuáles es pequeña y la otra es de tamaño "adecuado", la diferencia en información será significativa por lo que seguramente serán separadas.

El mejor caso al utilizar la imagen de letras rotadas y escaladas se muestra en la Figura 5.6. Es muy similar al resultado obtenido en el caso anterior, lo que implica que el perjuicio provocado por la escala tiene más efecto que el beneficio proporcionado por la invariabilidad a la rotación.

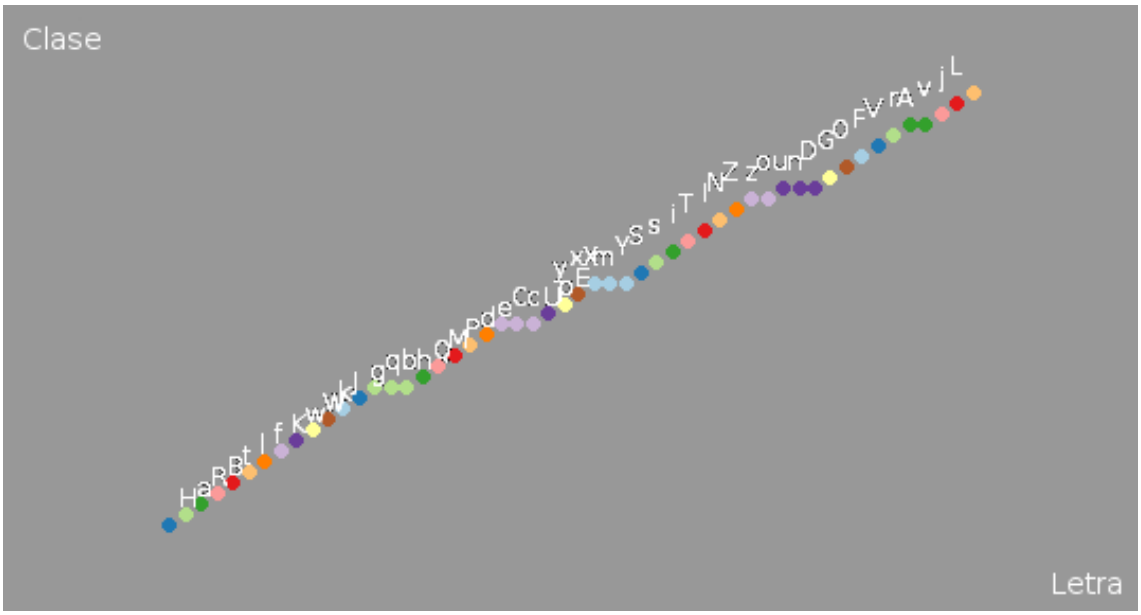


Figura 5.7: Mejor caso ($k = 10$) de los resultados obtenidos con la imagen de control y una retina de tamaño variable.

Resultados utilizando una retina con tamaño variable

Al utilizar una retina de tamaño variable, ésta es ajustada para adaptarse al tamaño de la imagen antes de aplicar el mapeo log-polar. Intuitivamente, esto debe mejorar el reconocimiento ya que el cambio de tamaño mantiene la proporción entre la retina y la imagen, de modo que la información de la imagen se distribuya en toda la retina sin importar su tamaño; a su vez, se espera que esto maximice la similitud de las imágenes log-polares entre diferentes tamaños de imágenes.

Una desventaja o punto negativo de esta idea es que utilizar una retina de tamaño variable implica construirla tantas veces como imágenes de interés, a diferencia de usar una retina de tamaño fijo, que solo requiere ser construida una vez y puede ser reutilizada.

El resultado del reconocimiento con la imagen de control utilizando una retina variable (Figura 5.7) es muy similar al experimento análogo utilizando una retina de tamaño fijo. También se generaron 42 clases aunque en este caso hubo 2 clases extrañas, "xXm" y "rA".

El experimento utilizando la imagen de letras rotadas (Figura 5.8) muestra tam-

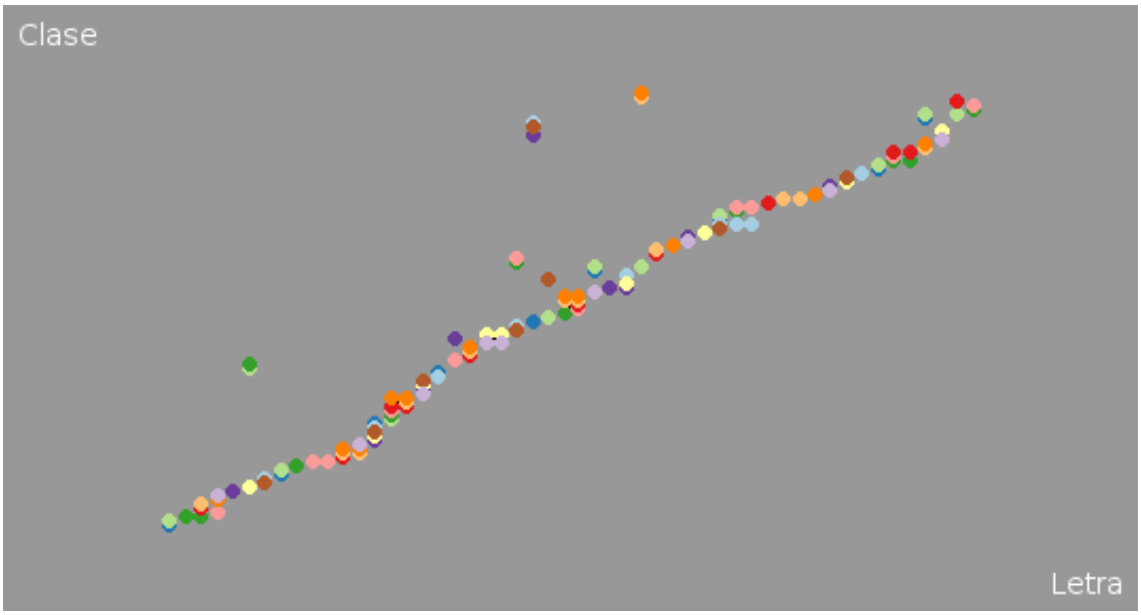


Figura 5.8: Mejor caso ($k = 8$) de los resultados obtenidos con la imagen de letras rotadas y una retina de tamaño variable.

bién un muy buen nivel de reconocimiento, similar al análogo utilizando una retina de tamaño fijo.

La Figura 5.9 muestra el resultado del experimento utilizando la imagen de letras escaladas. Se puede observar que se obtuvo un mejor reconocimiento en comparación con el experimento análogo utilizando una retina de tamaño fijo; sin embargo, aún hay varios puntos fuera de la forma principal y se presentaron también puntos alineados horizontalmente con el mismo color.

Por último está la imagen con letras rotadas y escaladas; la Figura 5.10 muestra el resultado. Una vez más se ve mejoría en el reconocimiento, sin embargo aún quedan varios puntos afuera de la forma principal, tal como en el caso anterior.

5.2. Reconocimiento de patrones utilizando el método SIFT

Para poner a prueba el método SIFT se implementó un esquema de reconocimiento gráfico basado en una idea descrita por Lowe (2004). Ésta consiste en generar una base de datos con los rasgos SIFT del conjunto de objetos que se quiere reconocer; cuando se encuentra una coincidencia entre los rasgos de la base de datos y los de una imagen

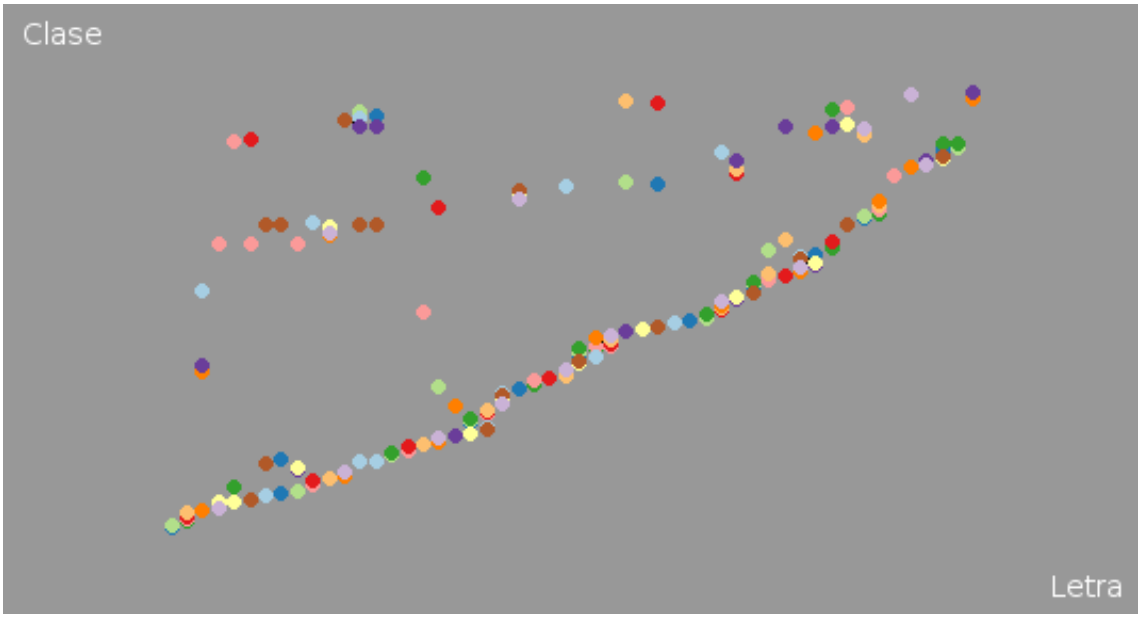


Figura 5.9: Mejor caso ($k = 26$) de los resultados obtenidos con la imagen de letras escaladas y una retina de tamaño variable.

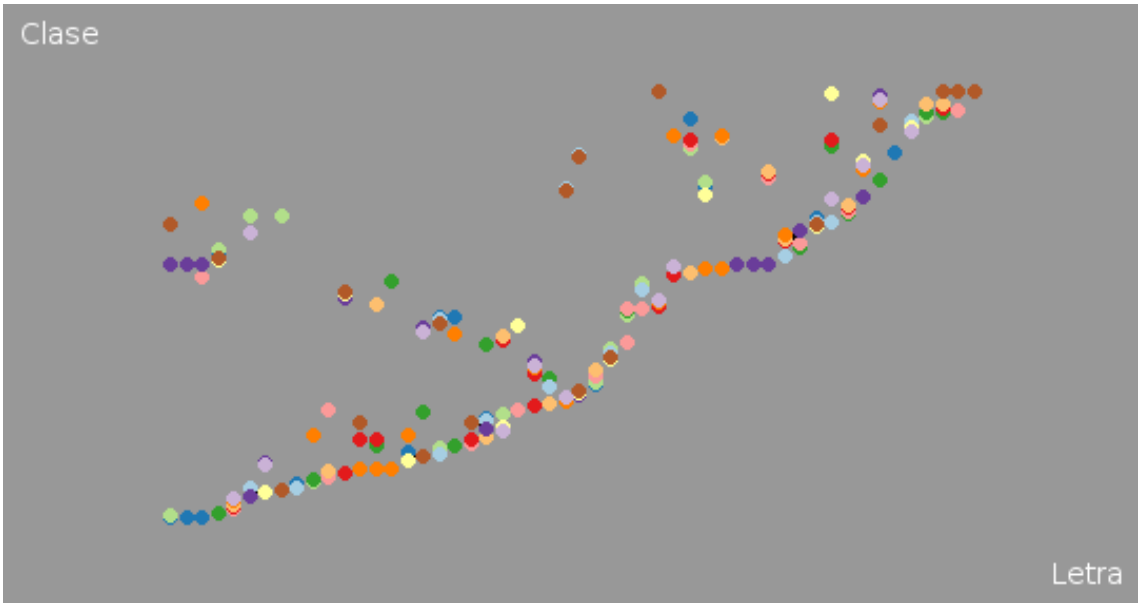


Figura 5.10: Mejor caso ($k = 13$) de los resultados obtenidos con la imagen de letras rotadas y escaladas y una retina variable.

de prueba, se dibujan los *keypoints* correspondientes en sus respectivas imágenes.

5.2.1. Descripción del método

Inicialmente se genera la base de datos con los rasgos SIFT de las imágenes de objetos de interés, llamadas imágenes de entrenamiento. Cuando una imagen de prueba es alimentada a la implementación para buscar correspondencias, lo primero es obtener sus rasgos SIFT. Luego se toma el primer rasgo de la imagen de prueba y se compara contra los rasgos de todos los objetos en la base de datos hasta encontrar el rasgo más cercano; el criterio de cercanía es la distancia euclidiana entre los rasgos que se comparan.

El siguiente paso consiste en buscar el segundo rasgo más cercano únicamente sobre los rasgos del objeto donde se encontró el más cercano; una vez encontrado se calcula la razón de distancias entre el rasgo más cercano y el segundo más cercano. Si el valor de la razón es menor que un umbral, se confirma que el rasgo de la imagen de prueba y el rasgo más cercano son una coincidencia, de lo contrario, se concluye que no existe coincidencia y se repite el proceso para el siguiente rasgo de la imagen de prueba.

Por último, se dibujan los *keypoints* correspondientes a los rasgos que hayan sido coincidencia en la imagen de prueba y en la de entrenamiento. El proceso se repite hasta que todos los rasgos SIFT de la imagen de prueba han sido comparados con la base de datos.

5.2.2. Condiciones de experimentación

Debido a los resultados que se obtuvieron de la implementación del método SIFT, la experimentación exhaustiva está fuera de lugar, sin embargo se mostrarán los resultados de la aplicación de reconocimiento para ilustrar el método. El objetivo en todo caso es identificar correctamente los rasgos de un par de letras de entre un conjunto con señuelos.

Se preparó una única imagen de prueba, mostrada en la Figura 5.11, con un conjunto de letras de entre las cuales se intentará reconocer la “A” y la “u”; estas letras son representadas tal cual en sus imágenes de entrenamiento, sin efectos de rotación o

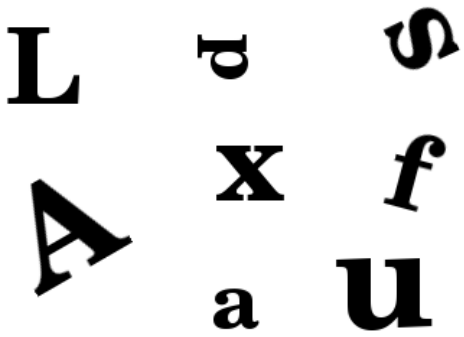


Figura 5.11: Imagen de prueba para la implementación del método SIFT.

escala. Por otro lado, la "A" fue rotada en la imagen de prueba mientras que la "u" fue escalada.

Los parámetros utilizados para aplicar el método SIFT fueron $n = 3$, $s = 3$ y $\sigma_0 = 1.6$; éstos, así como también las imágenes de entrenamiento y la de prueba, fueron escogidos deliberadamente con el propósito de obtener un resultado ilustrativo.

5.2.3. Resultados del reconocimiento de patrones utilizando el método SIFT

La aplicación ideal del reconocimiento de patrones mediante el método SIFT implica encontrar una correspondencia entre todos los rasgos de la imagen de prueba y la imagen de entrenamiento que le corresponde, sin importar los cambios de rotación, escala, cambio del punto de vista o iluminación, que cualquiera de las dos pudiera tener.

En la Figura 5.12 se muestra la imagen de prueba, en la parte superior, y las imágenes de entrenamiento, en la inferior, después de ejecutar la aplicación de reconocimiento. Las imágenes de entrenamiento tienen asignado un color exclusivo para sus *keypoints* y cada *keypoint* dibujado en la imagen de prueba corresponde a la imagen de entrenamiento con el mismo color de *keypoints*.

A pesar de las diferencias en orientación y tamaño, se encontraron rasgos en común en las letras buscadas; en cada caso se encontraron 2 coincidencias correctas. Por otro

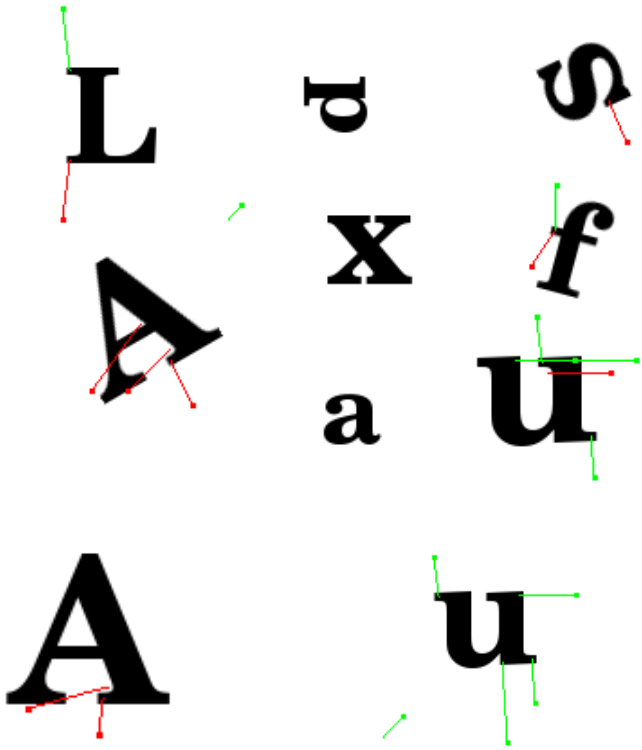


Figura 5.12: Resultado del reconocimiento utilizando el método SIFT. Cada flecha representa una coincidencia entre un *keypoint* de la imagen de prueba y otro de una de las imágenes de entrenamiento; el color rojo indica que el *keypoint* pertenece a la imagen de entrenamiento "A", mientras que el verde, a la imagen de entrenamiento "u".

lado, hubo muchas coincidencias falsas las cuáles deberían haber sido atrapadas por el algoritmo de reconocimiento; esto puede atribuirse a que los rasgos generados por la implementación no son lo suficientemente distintivos.

Capítulo 6

Conclusiones

Los resultados demostraron que el mapeo log-polar es en efecto útil para el reconocimiento de patrones. En particular, la invariabilidad a la rotación permitió que los efectos de la rotación fueran transparentes para el reconocimiento, obteniendo muy buenos resultados. Por otro lado, a pesar de la invariabilidad a la escala, el reconocimiento se vio perjudicado significativamente por las diferencias en tamaño de las imágenes.

El uso de una retina de tamaño variable favoreció el aprovechamiento de la invariabilidad a la escala por lo que se obtuvieron mejores resultados que al usar la retina de tamaño fijo. Cabe mencionar que la mejora observada no pareció significativa, sin embargo convendría hacer un análisis más profundo y determinar cuantitativamente la mejora.

Se obtuvieron resultados positivos de la implementación del método SIFT, sin embargo se detectaron problemas de sensibilidad a parámetros y de confiabilidad. La implementación detecta *keypoints* correctamente, sin embargo este conjunto cambia significativamente al variar los parámetros del método o las características de la imagen; se sospecha que estos problemas pueden deberse a la calibración de parámetros. En todo caso, la implementación desarrollada representa una primera aproximación al método SIFT que requiere de más trabajo y refinamiento.

A pesar de estos resultados se llevó a cabo un único experimento de reconocimiento con el fin de ilustrar la efectividad del método SIFT en este contexto. El experimento demostró que los *keypoints* de un objeto son encontrados en diferentes vistas del mismo, a pesar incluso de la presencia de señuelos.

6.1. Trabajo Futuro

Mapeo log-polar

1. La implementación actual del mapeo log-polar requiere que cada imagen log-polar construya su propia retina; esto resulta ineficiente si se desea generar un conjunto de imágenes log-polares con retinas iguales. Se propone rediseñar la implementación para que la construcción de la retina sea independiente de la generación de la imagen log-polar, de tal modo que varias imágenes log-polares puedan hacer uso de la misma retina.

2. Estudio del efecto del área de aplicación en el mapeo log-polar en relación con el número de rayos y anillos de la retina, de modo que se pueda determinar el número de rayos y anillos más adecuado para cierta área.

3. Realizar experimentos con la aplicación de reconocimiento ejecutando varias veces la misma corrida con el fin de estabilizar los resultados que arroja k–medios.

4. Definir medidas para los resultados del reconocimiento.

Método SIFT

1. Continuar el desarrollo del método SIFT hasta alcanzar alto grado de repetibilidad y baja sensibilidad al cambio de parámetros.

2. Rediseño de la implementación. Su estado actual hará difícil su mantenimiento futuro.

3. Agregar la implementación del método SIFT a la biblioteca VisionLibs.

4. Implementar un mejor algoritmo de búsqueda para la base de datos de la aplicación de reconocimiento.

5. Combinar el método SIFT y el mapeo log-polar para crear un nuevo método de representación que aproveche las características de ambas transformaciones.

Bibliografía

Belongi, S. I. M., Malik, J. I. M., y Puzicha, J. (2002). Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–521.

Brown, M. y Lowe, D. G. (2002). Invariant Features From Interest Point Groups. *British Machine Conference*, páginas 656–665. Cardiff, Wales.

Cobos, P. y Monasterio-Huelin, F. (1999). FPGA Implementation of a Log-Polar Algorithm for Real Time Applications. En *DCIS 1999*, páginas 63–68.

Davis, E. R. (1997). *Machine Vision.* Academic Press. ISBN 0-12-206092-X.

Escobar, M. J. y Ruiz-del Solar, J. (2002). Biologically-based Face Recognition Using Gabor Filter and Log-Polar Images. En *IJCNN*, páginas 1143–1247.

Forsyth, D. A. y Ponce, J. (2003). *Computer Vision: A Modern Approach.* Prentice Hall. ISBN 013-085198-1.

González, R. C. y Woods, R. E. (1996). *Tratamiento Digital de Imágenes.* Addison Wesley Iberoamericana. ISBN 0-201-62576-8.

Kurita, T., Hotta, K., y Mishima, T. (1998). Scale and Rotation Invariant Recognition Method Using Higher-Order Local Autocorrelation Features of Log-Polar Image. En *ACCV*, volumen 2, páginas 89–96.

Lindeberg, T. (1994a). Scale-Space Theory: A Basic Tool for Analysing Structures at Different Scales. *Journal of Applied Statistics*, 21(2):224–270. (Supplement on Advances in Applied Statistics: Statistics and Images: 2).

Lindeberg, T. (1994b). *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers. ISBN 0-7923-9418-6 (In book's description).

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International journal of computer vision*, 60(2):91–110.

Nalwa, V. S. (1993). *A Guided Tour of Computer Vision.* Addison-Wesley. ISBN 1-201-54853-4.

Schindler, K. (2004). Construction and Detection of Straight Lines, Distances, and Circles in Log-polar Images. En *5th Workshop on OMNIVIS 2004.* In conjunction with ECCV 2004.

Weiman, C. F. R. y Chaikin, G. (1979). Logarithmic Spiral Grids for Image Processing and Display. *Computer Graphics and Image Processing*, (11):197–226.

Wilson, S. W. (1983). On the Retio-Cortical Mapping. *Man-Machine Studies*, (18):361–389.

Wolberg, G. y Zokai, S. (2000). Robust Image Registration Using Log-Polar Transform. En *ICIP 2000*, volumen I, páginas 493–496.