DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Descripción de imágenes Mediante el Relevamiento de Marcas Naturales y Descriptores Neurológicos

Tesis de Licenciatura

Mariano Hernán Tepper

Directora de tesis: Marta Mejail
Buenos Aires, 2006

RESUMEN

La extracción de la información implícita presente en las imágenes generadas por un sistema de visión, puede ser usada tanto para reconocer, para comparar y para estimar las posiciones de los objetos presentes en las mismas como para que un robot pueda estimar su propia posición. Uno de los objetivos centrales de un sistema visual es lograr que desarrolle sus tareas en el modo más autónomo posible, necesitando poca o ninguna información a priori y que sea capaz de recolectar y organizar información relevante del ambiente. Muchos usan marcas artificiales tales como los reflectores de código de barras, patrones visuales, etc. pero éstos, obviamente, no funcionan cuando no hay marcas de este tipo. Por lo tanto, los sistemas basados en visión que utilizan marcas naturales estables en un ambiente, son altamente recomendados en un amplio rango de aplicaciones. La construcción de mapas a partir de estas marcas naturales sirve como base para realizar tareas de alto nivel, tales como navegación de un robot móvil. En este trabajo presentamos, en primer lugar, un sistema de detección de marcas naturales mediante una representación multiescala de la imagen, el espacio-escala. Desarrollamos algunas técnicas para la extracción de marcas naturales, mediante el estudio del espacio-escala y su estructura. Luego introducimos una forma de generar descriptores para las marcas, basada en ideas extraídas del sistema neurológico. Incorporamos a estos descriptores un método para producir una representación vectorial de los mismos invariante a rotación. Finalmente mostramos algunas técnicas para el almacenamiento, apareamiento (árboles k-d) y comparación robusta (Transformación de Hough, RANSAC) de imágenes a través de sus respectivas marcas y los descriptores asociados.

ABSTRACT

Implicit information in images can be extracted by a vision system and then used in object recognition problems as well as in location and mapping problems in robotics. One of the main objectives of a visual system is that it must work as autonomously as posible. It must use few information a priori, or none, and be able to collect and organize the relevant incoming visual data. Many systems use artificial marks such as code bar reflectors or visual patterns. These systems obviously fail to cope with the absence of such marks. Therefore, systems based on stable natural marks are highly recommended for a large range of applications. Building maps from these natural marks is essential in high level tasks, i.e. autonomous mobile robot navigation. In this work we introduce a natural marks based system. It works by analyzing a multi-scale representation of the images, called the scale-space. We have developed a method for natural marks extraction, following a study of the scale-space structure. This work also adresses the description of the marks by means of a neurobiological-based aproach. We developed as well a rotation invariant method to serialize descriptors. We also present techniques for descriptor storage (k-d trees) and robust matching (Hough Transform and RANSAC). Finally we show some visual examples and then provide theoretical and practical conclusions.

ÍNDICE GENERAL

1. INTRODUCCIÓN

Desde hace miles de años, el problema de la interrelación entre percepción y pensamiento ha sido un foco de interés primordial de la filosofía. Más recientemente, la psicología y la neurociencia se han dedicado al problema. Según Arnheim [1], el modelo clásico para relacionar la visión y el pensamiento, en pos de claridad y prolijidad teóricas, separa el proceso mental involucrado en dos etapas disjuntas: la percepción, entendida como recepción de información, y el tratamiento intelectual que se haga de la misma. Se sitúa a la percepción en un nivel menor de abstracción que al pensamiento; en síntesis, no existe una modalidad perceptual del mismo. El pensar consiste en operaciones intelectuales centradas en material cognoscitivo abstracto, despojado de sus características visuales y, por ende, adecuado para las operaciones intelectuales. Se entiende entonces que existe una etapa intermedia del proceso que transforma los perceptos en bruto en conceptos.

Más recientemente, surgieron corrientes tendientes a unificar estos procesos. Ya no se los entiende como la combinación sucesiva y secuencial de dos etapas. Operaciones tales como la exploración activa, la selección, la captación de lo escencial, la simplificación, la abstracción, el análisis y la síntesis, el completamiento, la corrección, la comparación, como también la combinación, la separación y la inclusión en un contexto no son operaciones inherentes a ninguna de las funciones mentales. Es decir que el conjunto de operaciones cognoscitivas no son un privilegio exclusivo de un estrato conceptual abstracto, sino que existen y se articulan en todas y cada una de las funciones del proceso mental.

Citando al mismo Arnheim:

> "Al mirar un objeto, tratamos de alcanzarlo con un dedo invisible, nos movemos a través del espacio que nos rodea, nos dirigimos a los lugares distantes donde se encuentran las cosas, las

> tocamos, las asimos, examinamos sus superficies, seguimos sus bordes, exploramos su textura."

En este contexto, es posible inteligir la percepción como una operación compleja de procesamiento de información constitutiva e intrínsecamente perceptual.

El campo visual, definido por la luz que penetra en el mismo, posee una estructura definida y objetiva. La psicología de la atención ha estudiado cómo, cuando el foco de atención del ojo difiere del centro de atención del campo visual, se produce una tensión que únicamente es eliminada mediante la fijación de la atención en este centro.

El sentido de la expresión "centro de atención del campo visual" es aquí intuitivo y el objetivo de esta tesis es, en primer lugar, intentar aportar una definición formal y correcta desde la intuición, aunque pueda no serlo desde un análisis psicológico riguroso (un objetivo no perseguido en este trabajo). Obviamente, ya que el acercamiento es desde la ciencia de la computación y, más precisamente, del procesamiento digital de imágenes, buscamos que dicha definición sea computable, reproducible, repetible y, en la medida de lo posible, computacionalmente eficiente.

Esto nos remite a la teoría del espacio-escala, elaborado por Witkin [23], continuado por Koenderink y Van Doorn [14] y completado por Lindeberg [15]. La tarea de un sistema de procesamiento visual consiste en la extracción de información significativa y significante acerca del mundo exterior desde un conjunto de valores de píxeles que son el resultado de mediciones de luminosidad de una escena física.

El problema de la visión es conceptualmente imposible de resolver si se lo plantea como puramente matemático. Es claro que, dada una imagen, existen infinitas escenas reales que podrían haberla originado. Desde este punto de vista, el problema está mal planteado en el sentido que no cumple con los tres criterios dados por Hadamard [7]:

1. existe una solución,
2. la solución es única y
3. la solución depende únicamente de los datos de entrada.

Sin embargo, los sistemas de visión biológicos presentes en la naturaleza son capaces de superar esta indeterminación con una notable heterogeneidad de acercamientos y soluciones. Podemos intentar un acercamiento heurístico al problema, aunque no por eso desprovisto de ciertos elementos formales.

Las personas son capaces de percibir extructuras en las imágenes, más allá de la percepción de tridimensionalidad y del reconocimiento de objetos conocidos previamente. Como establecimos anteriormente, imponemos un orden, una organización objetiva, aún cuando no sabemos qué estamos viendo. Aunque la escuela de psicología gestáltica estableció ciertas reglas a este respecto, no tenemos un entendimiento satisfactorio sobre cómo estos mecanismo actúan cuantitativamente.

Esto nos lleva al problema de la escala ya que los objetos en el mundo real y los detalles en una imagen sólo existen y tienen sentido en un número limitado de escalas. por ejemplo, los detalles "finos" siempre están presentes en los objetos, pero son obviados casi siempre que miramos un objeto. Podría especularse que la estructuración de la percepción en escalas brinda la posibilidad de organizar un entorno extremadamente complicado en forma jerárquica y simplificada: es posible encontrar las propiedades de las imágenes que permiten este proceso. Lo llamaremos de aquí en adelante representación multiescala y denominaremos suavizado en el espacio-escala al mecanismo de simplificar sistemáticamente la imagen reduciendo la "información de alta frecuencia".

La siguiente pregunta que surge naturalmente es ¿qué es estructura? Witkin [23] observó que el número de ceros en la segunda derivada de una señal decrecía monótonamente con la escala y lo tomó como la característica básica de la representación del espacio-escala. De hecho, esta propiedad es válida para derivadas de cualquier orden y, en consecuencia, la cantidad de extremos locales en cualquier derivada de una señal también decrece con la escala.

En otras palabras, podemos analizar los extremos locales en alguna derivada de una señal y tendremos una medida de la estructura presente en una escala determinada.

1.1. Organización

En el capítulo 2, introducimos la noción de espacio-escala con mayor formalidad, para luego, en el capítulo 3, pasar a proponer un nuevo método para encontrar puntos clave en la estructura subyacente de una imagen. Ya que estos puntos son inherentes a las propiedades que presenta una imagen dada, los llamaremos marcas naturales.

En el capítulo 4, presentamos un mecanismo que hace posible describir localmente la imagen en cada marca natural y que cuenta con algunas propiedades realmente interesantes, a saber, tolerancia frente a cambios de luminosidad e invarianza a rotaciones. Éste está basado en la estructura y comportamiento de un tipo de células del córtex primario (la zona del cerebro que más involucrada está en los procesos perceptuales) llamadas células complejas. Describiremos también el algoritmo que desarrolamos para producir una representación vectorial e invariante a rotaciones, de cada descriptor.

En el capítulo 5 mostramos un sistema eficiente para el almacenamiento y búsqueda de descriptores que nos permitea armar conjuntos de correspondencias entre los descriptores de dos fuentes (dos imágenes o una imagen y una base de datos).

Para poder comparar y medir efectivamente estos métodos es necesario proveer mecanismos para evaluarlos. Naturalmente, el reconocimiento y la estimación de la pose de una imagen en otra ofrecen una idea cabal de la correctitud y exactitud de los mismos. Para ellos utilizamos estimadores robustos, descriptos en el capítulo 6.

Finalmente, en los capítulos 7 y 8 presentamos y discutimos los resultados obtenidos, aportando conlusiones y posibles trabajos futuros.

2. LA FUNCIÓN ESPACIO-ESCALA

Cuando nos encontramos ante la tarea de extraer información de una imagen nos topamos con la noción de escala. El acercamiento elegido para procesar una imagen está fuertemente ligado a la escala de las características que se desea encontrar. Durante mucho tiempo este concepto fue tratado implícitamente. Más recientemente, la noción de escala empezó a ser utilizada en mayor medida, sobre todo en las primeras etapas del procesamiento. Llamaremos a estas etapas visión temprana.

La escala tiene una ingerencia directa y fundamental en dos cuestionamientos básicos que surgen al tratar con imágenes: ¿qué información debe ser extraída? ¿y qué operaciones deben utilizarse para hacerlo?

Es de vital importancia encontrar un acercamiento que no presuponga ningún conocimiento a priori acerca de la naturaleza del contenido de la escena representada por la imagen.

En particular, dado que no contamos con información que nos indique la escala apropiada para procesar la imagen, debemos hacerlo con todas las escalas posibles. En otras palabras, necesitamos una representación multiescala.

Es de esperarse también que la representación no se vea afectada, o lo sea lo menos posible, por los fenómenos presentes en una escena (cambios de luminosidad, movimiento de objetos rígidos, etc.). Éstas son las restricciones que tenemos al definir una representación multiescala.

El espacio-escala nos brinda una teoría formal que define claramente las operaciones a realizar para contar con una representación natural temprana de la imagen.

A modo de disgresión histórica, usualmente se acepta que Witkin [23] es el precursor de la teoría (en señales unidimensionales), aunque Iijima, ya en 1959, introdujo la noción en una serie de artículos en japonés [12]. La teoría

fue sujeto de numerosas investigaciones en Japón, durante mucho tiempo desconocidas en occidente [10, 11, 13, 22].

El espacio-escala es una representación multiescala con dos características principales:

1. tiene un parámetro de escala continuo, y

2. preserva el mismo nivel de muestreo en todas las escalas.

La idea es simple y consiste en introducir la señal original en una familia monoparamétrica de señales derivadas. [1] Éstas son construídas mediante la convolución con una familia, también monoparamétrica, de kernels gaussianos de varianza creciente.

2.1. Tratamiento en señales continuas

El espacio-escala de una función (o señal) unidimensional $f : \mathbb{R}^N \to \mathbb{R}$ es otra función $L : \mathbb{R}^N \times \mathbb{R}_+ \to \mathbb{R}$ tal que

$$L(\bullet, t) = \begin{cases} f & \text{si } t = 0 \\ g(\bullet, t) * f & \text{si } t \neq 0 \end{cases} \tag{2.1}$$

donde $t \in \mathbb{R}_+$ es el parámetro de escala y $g : \mathbb{R}^N \times \mathbb{R}_+ \setminus \{0\} \to \mathbb{R}$ es el kernel gaussiano:

$$g(x, t) = \frac{1}{(\pi t)^{N/2}} e^{\frac{-x^T x}{2t}} \qquad \text{con } x \in \mathbb{R}^N \tag{2.2}$$

También podemos ver la función L como la solución de la siguiente ecuación de difusión:

$$\partial_t L = \frac{1}{2} \nabla^T \nabla L \tag{2.3}$$

con condición inicial $L(\bullet, 0) = f$ que no es otra cosa que la ecuación física de difusión isotrópica (en un medio con conductividad homogénea) del calor.

A partir de esta representación, podemos definir las derivadas espaciales multiescala como

$$L_{x^{\vec{n}}}(\bullet, t) = \partial_{x^{\vec{n}}} L(\bullet, t) = g_{x^{\vec{n}}}(\bullet, t) * f \tag{2.4}$$

[1] Derivada no en el sentido de "diferenciación" sino que se calculan a partir de la original.

donde $\vec{n} = (n_1, n_2, \ldots, n_N)^T \in \mathbb{Z}^N$ y
$\partial_{x^{\vec{n}}} = \partial_{x_1^{n_1}} \partial_{x_2^{n_2}} \ldots \partial_{x_N^{n_N}}$ con $x = (x_1, x_2, \ldots, x_N)^T \in \mathbb{R}^N$.

En términos de integrales la ecuación anterior puede ser escrita como

$$\begin{aligned} L_{x^{\vec{n}}}(x,t) &= \int_{x' \in \mathbb{R}^N} g_{x^N}(x - x', t) f(x') dx' \\ &= \int_{x' \in \mathbb{R}^N} g_{x^N}(x', t) f(x - x') dx' \end{aligned} \tag{2.5}$$

Si está acotada por un polinomio, es decir que existen $C_1, C_2 \in \mathbb{R}_+$ tales que

$$|f(x)| \leq C_1 (1 + x^T x)^{C_2} \qquad \text{con } x \in \mathbb{R}^N \tag{2.6}$$

entonces la integral converge para todo $t > 0$. [2]

La ecuación 2.4 nos permite construir derivadas multiescalas (de cualquier orden) de la función f, más allá de que f sea o no diferenciable. El espacio-escala puede, por ende, considerarse como infinitamente diferenciable.

2.2. Propiedades del espacio-escala

Como ya dijimos anteriormente, el espacio-escala va reduciendo la información "fina" a medida que crece el parámetro de la escala. Intuitivamente, si queremos suprimir las estructuras con un tamaño característico menor a σ, podemos hacerlo mediante la convolución con un kernel gaussiano con varianza σ^2. [3] Para ver ejemplos, dirigirse al capítulo 7.

Witkin observó en sus trabajos que en la familia L no se crean nuevos extremos locales. Pese a que Witkin trabajó con funciones unidimensionales, el resultado es válido para funciones N-dimensionales. Como la diferenciación conmuta con la convolución:

$$\partial_{x^{\vec{n}}} L(\bullet, t) = \partial_{x^{\vec{n}}} (g(\bullet, t) * f) = g(\bullet, t) * \partial_{x^{\vec{n}}} f \tag{2.7}$$

la propiedad puede ser aplicada para cualquier derivada de f calculada a partir de la representación del espacio-escala.

En trabajos posteriores, Lindeberg [15] mostró que esta característica puede ser usada para detectar estructuras en una imagen.

[2] cuando f es una imagen está evidentemente acotada.

[3] Esto no se cumple exactamente, sólo a nivel intuitivo.

El suavizado gaussiano había sido usado previamente por Marr [20], considerando los ceros del Laplaciano de imágenes convolucionadas con kernels gaussianos. Sin embargo, es el espacio-escala el que aportó una forma sistemática para relacionar el concepto de escala con el de suavizado.

Koenderink [14] avanzó en el concepto del espacio-escala, aplicándolo a imágenes [4]. Toda imagen está acotada en escala por dos valores:

- *la escala interior:* que se corresponde con la escala de mayor detalle alcanzable (determinada por el tamaño del píxel en imágenes digitales o por la granularidad de la película en fotografía);
- *la escala exterior:* determinada por el tamaño de la imagen.

El parámetro de escala en el espacio-escala nos permite analizar este rango: el que va de la escala interior hasta la exterior.

Koenderink también introdujo el concepto de *causalidad*, que significa que nuevas superficies de nivel $\{(x, y, t) \in \mathbb{R}^2 \times \mathbb{R} : L(x, y, t) = L_0\}$ no deben ser creadas en el espacio-escala cuando la escala crece. Combinando la causalidad con las nociones de *isotropía* y *homogeneidad*, que básicamente expresan que todas las posiciones espaciales y todas las posiciones en la escala deben ser tratadas de forma similar, demostró que la representación del espacio-escala debe satisfacer la ecuación de difusión

$$\partial_t L = \frac{1}{2}\nabla^2 L. \tag{2.8}$$

Como el kernel gaussiano es el que resuelve la ecuación del calor en un dominio infinito, se desprende que también es el único kernel para generar el espacio-escala. Resultados similares son válidos también en una o más de dos dimensiones.

Otro concepto fundamental de la representación del espacio-escala es el de *semi-grupo*:

$$h(\bullet, t_1) * h(\bullet, t_2) = h(\bullet, t_1 + t_2) \tag{2.9}$$

donde h es una kernel "suavizador". Esta propiedad es fundamental en la construcción del espacio-escala.

[4] En realidad, extendió la noción para funciones N-dimensionales.

Finalmente, concluiremos el tratamiento de la noción de espacio-escala en señales continuas enunciando algunas propiedades interesantes del kernel gaussiano:

- es invariante a rotación, y
- es separable.

2.3. Tratamiento en señales discretas

Las señales digitales son discretas, por lo que se impone una extensión de la teoría anterior adecuada para éstas. Esta extensión debe obviamente mantener las propiedades del caso continuo.

En [15], Lindeberg prueba que, para señales discretas, la representación del espacio-escala no es otra que $L : \mathbb{Z}^N \times \mathbb{R}_+ \to \mathbb{R}$

$$\begin{aligned} L(x,t) &= T(x,t) * f(x) \\ &= \sum_{n=-\infty}^{\infty} T(n,t) \cdot f(x-n) \end{aligned} \tag{2.10}$$

donde $T : \mathbb{Z}^N \times \mathbb{R}_+ \to \mathbb{R}$ es un kernel llamado el análogo discreto del kernel gaussiano o simplemente, el *kernel gaussiano discreto*. En [16] Lindeberg lo define en términos de las funciones modificadas de Bessel I_n:

$$T(n,t) = e^{-\alpha t} I_n(\alpha t) \tag{2.11}$$

Este kernel satisface muchas propiedades, en el dominio discreto, similares a las del kernel gaussiano en el domino continuo que lo hacen utilizable para generar el espacio-escala discreto.

Para finalizar el capítulo, incluimos un resumen del comportamiento del espacio-escala $L = \mathcal{G}f$ frente a transformaciones, siendo $\mathcal{G}$ un operador, en este caso la convolución:

Transformación	Definición	Invarianza
traslación ($\mathcal{T}$)	$(\mathcal{T}L)(x,t)=L(x+\Delta x,t)$	$\mathcal{G}\mathcal{T}f=\mathcal{T}\mathcal{G}f$
rotación ($\mathcal{R}$)	$(\mathcal{R}L)(x,t)=L(Rx,t)$	$\mathcal{G}\mathcal{R}f=\mathcal{R}\mathcal{G}f$
cambio de escala ($\mathcal{U}$)	$(\mathcal{U}L)(x,t)=L(sx,t)$	$\mathcal{G}\mathcal{U}f=\mathcal{U}\mathcal{G}f$
cambio afín de intensidad ($\mathcal{A}$)	$(\mathcal{A}L)(x,t)=aL(x,t)+b$	$\mathcal{G}\mathcal{A}f=\mathcal{A}\mathcal{G}f$

3. DETECCIÓN DE MARCAS NATURALES

En la primera parte de este capítulo, analizaremos las pirámides como representación multiescala. Muchos sistemas comerciales se basan en representaciones de pirámide para la extracción de marcas naturales. De hecho, hay implementaciones en hardware de las mismas. Hay un amplio trabajo de Lowe [17, 18, 19], al que nos referimos más adelante, que también las utiliza. Asimismo, explicaremos cómo son estas representaciones multiescala y por qué no las utilizamos en este trabajo. En la segunda parte, detallaremos los métodos utilizados en este trabajo para la detección de puntos clave.

3.1. Pirámides multiescala

Las representaciones piramidales combinan una operación de submuestro con un paso de "suavizado" (ver fig. 3.1). Como el tamaño de la imagen decrece exponencialmente con la escala, la principal ventaja de las pirámides es que el costo computacional para procesar cada nivel también decrece exponencialmente. Como es de esperar, los requerimientos de memoria también se ven reducidos en esta representación.

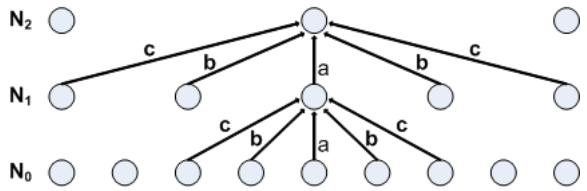

Fig. 3.1: Ejemplo de creación de puntos en los diferentes niveles de la pirámide. Los coeficientes a, b y c son elegidos para producir un efecto de suavizado. Es importante observar que en cada nivel los puntos están más espaciados por el efecto del submuestreo.

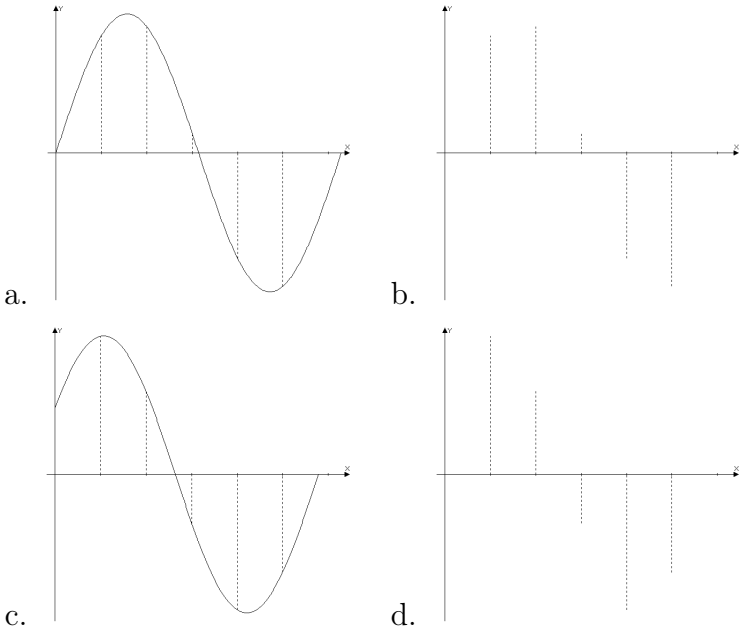

Fig. 3.2: Ejemplo de los problemas que puede ocasionar el muestreo de una función continua (a, b) y el muestreo de la misma función trasladada (c, d). El remuestro del caso discreto arroja consecuencias similares.

Las pirámides presentan algunas desventajas mayores, a saber:

- el remuestreo impone una discretización grosera en la escala. En otras palabras, no es posible la discretización del parámetro de la escala a voluntad, sino que está determinada por la naturaleza del método.
- las pirámides no son invariantes a traslaciones. Es fácil imaginarse cómo el muestreo de una señal y el muestreo de la misma señal trasladada pueden arrojar resultados altamente disímiles (ver fig. 3.2). El remuestreo realizado durante la creación de la pirámide acrecienta estos problemas de nivel en nivel.
- la construcción de esta estructura es obligadamente secuencial, como podemos ver en la fig. 3.1. No es posible a partir de la imagen original generar un nivel de escala sin generar previamente los anteriores.

Estos inconvenientes en la representación hacen que muchas personas

consideren las pirámides como predecesoras (conceptualmente, no históricamente) del espacio-escala. Alternativamente podemos verlas como una aproximación numérica del mismo. Pero las desventajas que conllevan se ven directamente reflejadas en problemas numéricos como, por ejemplo, relacionar estructuras a través de los niveles de escala.

En este trabajo hemos optado por utilizar la representación del espacio-escala, ya descrita en el capítulo 2. Para compensar el costo computacional suplementario de no utilizar submuestreo, hemos diseñado el procesamiento posterior buscando optimizar el costo total del método.

3.2. Relevamiento de marcas naturales

Mikolajczyk *et al.* [21] y Baumberg [3] utilizan el detector de esquinas de Harris [8], en diferentes niveles del espacio-escala discretizado, para detectar puntos clave.

Lowe [17, 18, 19] busca máximos y mínimos locales en la función Diferencia de Gaussianas (DG), que constituye una aproximación del Laplaciano de la Gaussiana (LG) sobre una representación multiescala de pirámide.

Ya a principios de los 80, Marr [20] utilizó la función LG y, en particular, la función DG que juegan un rol central en el sistema visual de los mamíferos. El operador Marr-Hildreth identifica los ceros de estas funciones como estructuralmente importantes.

Dada nuestra convicción de utilizar acercamientos basados en la percepción natural, también utilizaremos la función DG como la base para la detección de estructura en el espacio-escala.

Formalmente, la función DG de una imagen I es

$$\begin{aligned} DG(x,t) &= (G(x,t+\Delta t) - G(x,t-\Delta t)) * I(x) \\ &= L(x,t+\Delta t) - L(x,t-\Delta t) \end{aligned} \tag{3.1}$$

Podemos computarla con gran eficiencia a partir de la representación del espacio-escala en $O(kMN\log_2(MN))$ para una imagen de $M \times N$ y k niveles de escala.

Como dijimos anteriormente, Lowe propone buscar extremos locales en la DG. Consideramos que estos puntos son estructuralmente secundarios y

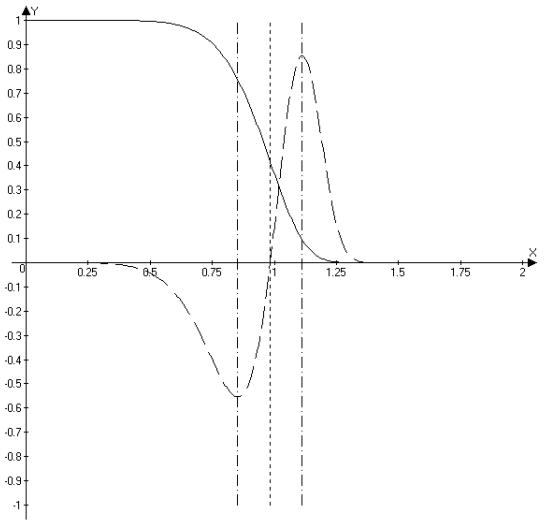

Fig. 3.3: La función $f(x) = e^{-\frac{x}{8}}$ (en línea continua) y su segunda derivada (en línea de rayas largas). La derivada está multiplicada por una constante para mejorar la visibilidad del ejemplo, pero esto no afecta la ubicación de sus ceros y de sus extremos. En líneas de rayas cortas, vemos el cero de la segunda derivada; en líneas alternadas de puntos y rayas, vemos sus extremos. Si consideramos la función f como un borde en una imagen, vemos que el punto realmente característico del borde es el que se corresponde con el cero de la f''.

están subordinados a los ceros de esta función (ver fig. 3.3). El operador Marr-Hildreth se basa en la misma premisa, que cuenta con un vasto apoyo en la literatura.

Recordemos que las propiedades de no creación de nueva estructura a medida que se avanza en el espacio-escala es válida para derivadas de cualquier orden.

Para realizar esta tarea, decidimos remitirnos a la búsqueda de extremos locales en una aproximación de la derivada de DG, que llamaremos dDG (ver fig. 3.4). Esta técnica presenta ciertas ventajas que detallamos a continuación

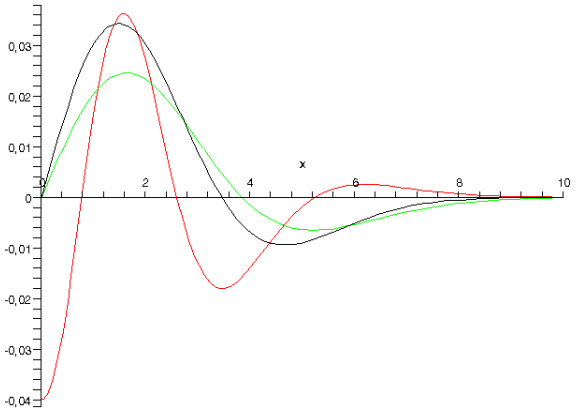

Fig. 3.4: Ejemplo de la segunda derivada de la función gaussiana $f(x, t = 2) = \frac{1}{(2\sqrt{(t\pi)}}e^{-(x^2)/(2t^2))}$ (en negro), la derivada de la función DG de f (en verde) y la función dDG (en rojo).

- la dDG es fácilmente computable:

$$\begin{aligned} dDG(x,t) &= DG(x,t+\Delta t) - 2DG(x,t) + DG(x,t-\Delta t) \\ dDG(x,t) &= L(x,t+2\Delta t) - 3L(x,t+\Delta t) \\ &\quad +3L(x,t-\Delta t) + L(x,t-2\Delta t) \end{aligned} \tag{3.2}$$

  Como podemos observar dDG tiene el mismo orden $O(cMN\log_2(MN))$ que DG para una imagen de $M \times N$ y c niveles de escala.

- las búsqueda de extremos locales es computacionalmente más fácil que la de ceros y no sufre de alteraciones numéricas.

- los extremos de la función dDG se corresponden con cambios de signo de la pendiente de la función DG. Cuando tomamos una función gaussiana y observamos su segunda y su tercera derivadas, es posible observar cómo los ceros de la segunda derivada son cercanos (pensando en términos discretos) a los extremos de la tercera derivada (consideramos sólo el de valor más alto en el intervalo) (ver fig. 3.5). En el capítulo anterior, mostramos que la diferenciación conmuta con la convolución, por lo que esta observación puede ser utilizada en el análisis del espacio-escala de una imagen.

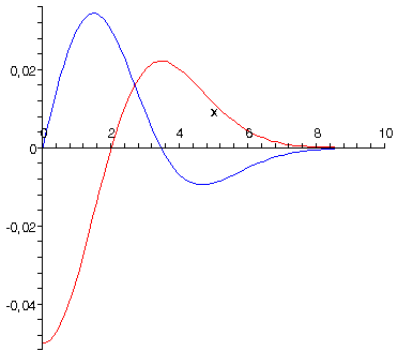

Fig. 3.5: Segunda derivada de la función gaussiana $f(x, t = 2) = \frac{1}{(2\sqrt{(t\pi)}}e^{-(x^2)/(2t^2)}$ (en rojo) y su tercera derivada (en azul).

- en [16], Lindeberg detalla el problema de la suposición de iso-intensidad del espacio-escala. El suavizado gaussiano va decrementando la intensidad de la función a medida que se avanza en la escala. Es por eso que la comparación entre escalas es difícil ya que es necesario encontrar un método para modelar la reducción de la intensidad. Pero utilizando *dDG* incorporamos información adicional de las escalas adyacentes sin presuponer la iso-intesidad de las mismas.

¿Por qué no trabajamos directamente con la primera derivada de la señal? Simplemente porque el cálculo de la segunda y la tercera derivadas es extremadamente sencillo y rápido, gracias a las aproximaciones *DG* y *dDG* respectivamente.

El proceso para obtener la función *dDG* consiste entonces en la construcción del espacio-escala con una discretización del parámetro de la escala y aplicar sobre éste, la función 3.2. El algoritmo resultante es el siguiente:

3.3. Paralelismo

Como dijimos anteriormente la construcción de la representación multi-escala de pirámide es necesariamente secuencial. Por el contrario, si nos basamos en la propiedad de *semi-grupo* del espacio-escala (detallada en §2.2) vemos que en el algoritmo 3.1 cada $L(\bullet, t)$, y cada $dDG(\bullet, t)$ puede computarse independientemente de los demás.

Algoritmo 3.1 Construcción de $dDG_k(x,t)$ discretizada con $0 \leq t \leq h$ donde k es la diferencia de escala entre niveles consecutivos y h es la varianza del kernel del último nivel utilizado.

$L(x,0) \leftarrow I(x)$
$i \leftarrow k$
while $i \leq h$ **do**
  $L(x,i) \leftarrow I(x) * G(x,i)$
  $i \leftarrow i + k$
end while
$i \leftarrow 2k$
while $i \leq h - k$ **do**
  $dDG(x,i) \leftarrow L(x,i+k) - 3L(x,i) + 3L(x,i-k) + L(x,i-2k)$
  $i \leftarrow i + k$
end while

Luego, queremos encontrar extremos locales en cada $dDG(\bullet, t_i)$. Para esto tampoco es necesario contar con información de otro nivel $dDG(\bullet, t_j)$ con $j \neq i$. Esta operación también puede realizarse independientemente para cada nivel.

En resumen, la independencia entre niveles de las operaciones a realizarse conduce inmediatamente a la posibilidad de paralelizar el algoritmo. Esto representa una reducción sustancial de la complejidad del algoritmo de $O(cMN \log_2(MN))$ en la construcción y $O(cMN)$ en la detección de extremos a $O(MN \log_2(MN))$ y $O(MN)$ respectivamente si contamos con c procesadores paralelos, cada uno correspondiente a una de las funciones del espacio-escala utilizadas.

3.4. Descripción del filtrado

Para encontrar marcas naturales en una imagen, buscamos entonces extremos locales de la función dDG. El número de estos extremos locales es extremadamente elevado. Si optásemos por utilizarlos todos para las etapas posteriores de descripción y apareamiento, éstas se volverían extremadamente costosas. Es suficiente con encontrar un subconjunto que sea representativo

de la función dDG en cada nivel. Necesitamos entonces una forma de elegir este subconjunto.

Definimos como la potencia de un punto en dDG a la función $P : \mathbb{Z}^2 \to \mathbb{R}$ tal que

$$P(x,t) = \left| dDG(x,t) - \frac{1}{(\#E) - 1} \sum_{\substack{y \in E \\ y \neq x}} dDG(y,t) \right| \tag{3.3}$$

donde E es un entorno de x.

Utilizamos esta función porque es más robusta frente a cambios de luminosidad:

- *globales:* al utilizar una diferencia de niveles de gris, es insensible a sumar una constante al nivel de gris de cada pixel.
- *locales:* el valor que toma es local, lo que reduce su sensibilidad a variaciones de luz no-lineales (por ejemplo, aplicar un foco en una zona de la imagen).

El algoritmo que desarrollamos para el filtrado es el siguiente. Sea $\mathcal{C}_i$ el conjunto de extremos locales del i-ésimo nivel del espacio-escala. Definimos la función $P^* : \{\mathbb{Z}^2\} \to \mathbb{R}$ como

$$P^*(\mathcal{C}_i) = \frac{1}{\#E} \sum_{x \in \mathcal{C}_i} P(x,i) \tag{3.4}$$

P^* es la potencia media de los extremos del conjunto $\mathcal{C}_i$.

Seleccionamos finalmente el conjunto de $\mathcal{C}_i^F$ de extremos:

$$\mathcal{C}_i^F = \{x | x \in \mathcal{C}_i \wedge P(x,i) \geq kP^*(\mathcal{C}_i)\} \tag{3.5}$$

Es decir que seleccionamos los extremos de un nivel cuya potencia es k veces más grande que la potencia media de lo extremos del mismo nivel.

La elección del valor de k es particularmente difícil. Adicionalmente, hasta aquí ignoramos el tamaño del entorno E para la búsqueda de los extremos locales. En el capítulo 7 presentamos mayores detalles sobre la elección empírica de estos dos valores.

4. DESCRIPTORES NEUROLÓGICOS

En las secciones anteriores, encontramos los puntos claves en la imagen, asignándoles una ubicación y una escala. Estos parámetros imponen un sistema de coordenadas bidimensional, determinístico y local en el cual es posible describir la región de la imagen, obteniendo invarianza a traslaciones y cambio de escala. Debemos ahora encontrar un descriptor, para la región circundante al punto clave, que sea lo suficientemente distintivo como para caracterizarla, tratando de maximizar la invarianza a los restantes grados de libertad (rotación, iluminación, punto de vista 3D, etc.)

Un acercamiento obvio sería computar las intensidades locales alrededor del punto clave en la escala apropiada, y aparearlas utilizando correlación normalizada. Sin embargo, la simple correlación es altamente sensible a cambios tales como transformaciones afines, cambio del punto de vista 3D o deformaciones no rígidas.

Edelman, Intrator y Poggio [4] encuentran una mejor representación, basada en un modelo de visión biológica para neuronas complejas en el córtex visual primario, que es la que utilizamos en este trabajo.

4.1. Fundamentos neurobiológicos

Gran parte del córtex en los primates está dedicado al procesamiento visual. Casi toda la información visual llega al córtex a través de la mayor y más importante área visual cortical (V1). Las células en V1 tiene campos receptivos (CR) alargados y, en consecuencia, responden mejor a estímulos también alargados (barras, bordes...). Hubel y Wiesel [9] clasificaron estas células siguiendo la complejidad de su respuesta, dividiéndolas en dos grupos: simples y complejas (ver fig. 4.1 a & b). Los campos receptores de las células simples contienen sub-regiones que ejercen una influencia excitatoria en la

respuesta de la célula (en gris claro) y otras sub-regiones, que ejercen una inhibitoria (en gris oscuro). La primera línea debajo de las células indica el comienzo y el fin de la estimulación. Las líneas verticales debajo de ésta, muestran los impulsos nerviosos individuales. La estimulación más efectiva para un CR es la que pone grandes cantidades de luz en la zona excitatoria, y poca en la inhibitoria. Debe tener la orientación, la posición y el tamaño adecuados. Los estímulos que no encajan dentro de estos parámetros, son menos efectivos.

Las células complejas son las más numerosas en V1 (hasta un 75 % de la cantidad total). Como las células simples, responden sólo a estímulos orientados apropiadamente pero, a diferencia de las anteriores, no son exigentes sobre la posición del estímulo, mientras caiga dentro del CR. Muchas células complejas también son selectivas a la dirección, en el sentido que responden sólo a un estímulo que se mueve en una dirección y no cuando se mueve en la dirección opuesta.

Hubel y Wiesel fueron los primeros en descubrir que las células en V1 están dispuestas en una forma precisa y ordenada. El ajuste de orientación es una característica prominente en sus respuestas individuales. A medida que se avanza adentrándose en el córtex, a través de sucesivas capas de células y perpendicularmente a la superficie, todas las células en una determinada capa tienen el mismo ajuste de orientación. No sólo eso, el ajuste va cambiando gradualmente de capa a capa, para cubrir todo el rango angular. Se denomina "columnas de orientación" a este arreglo (ver fig. 4.1 c).

Otro gran determinante de las respuestas de las células es el ojo de origen. La gran mayoría de las células pueden ser excitadas por estímulos provenientes de cualquier ojo pero, en general, "prefieren" uno en particular por sobre el otro (propiedad llamada dominancia ocular). La dominancia ocular se presenta alternativamente en el sentido tangencial a la superficie del córtex (ver fig. 4.1 c).

Si vemos una columna de orientación desde arriba, con las células que la componen superpuestas, el esquema sería el de la figura 4.2

Edelman *et al.* hipotetizan que la función de estas neuronas (en especial la de las complejas) es permitir apareamiento y reconocimiento de objetos 3D desde diferentes puntos de vista. El modelo para la construcción de los des-

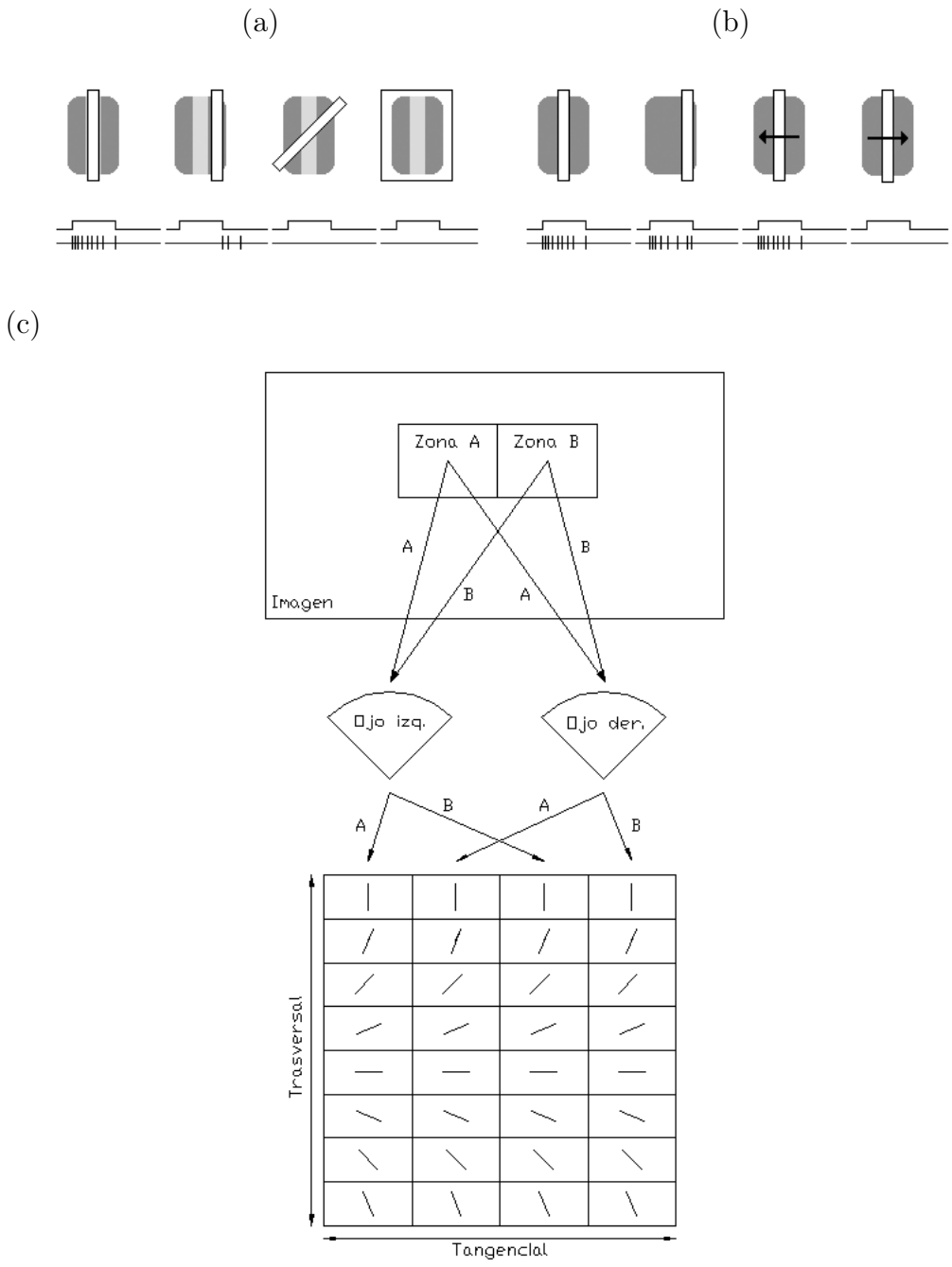

Fig. 4.1: Respuesta a diferentes estímulos de las células simples (a) y de las complejas (b) en el córtex primario. (c) Organización de las células en el córtex y representación de la dominancia ocular. Horizontalmente, la dirección es tangencial a la superficie del córtex y, verticalmente, es trasversal.

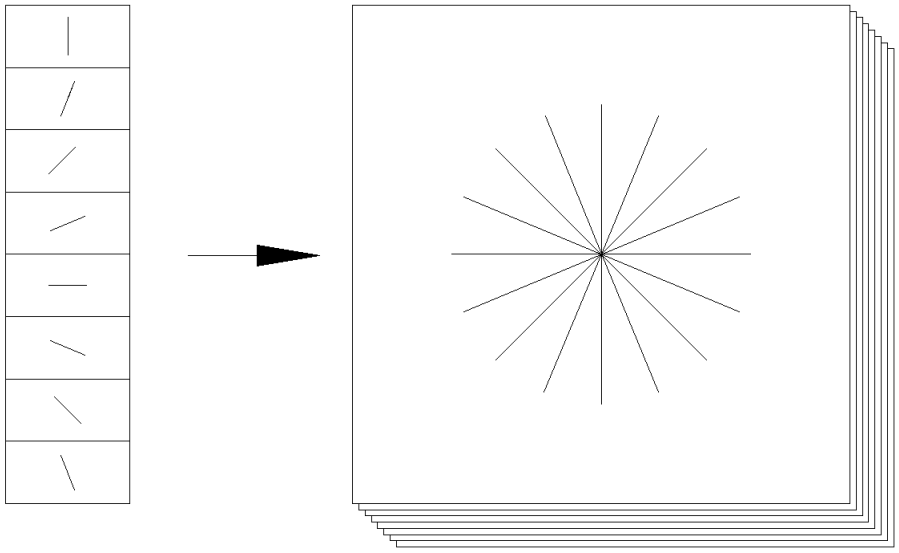

Fig. 4.2: Columna de orientación como superposición de las células del córtex

criptores presentado a continuación fue propuesto por Lowe [17] y se inspira en estas ideas.

4.2. Orientación canónica del descriptor

La asignación de una orientación basada en las propiedades locales de la imagen alrededor del punto clave es esencial para obtener invarianza frente a rotaciones. La forma más sencilla de realizar esta operación es utilizando el gradiente en el punto. Para aproximarlo, utilizamos diferencias finitas sobre la imagen de la pirámide de gaussianas a la que más cerca esté el punto en términos de escala. Llamando a esta imagen I, tenemos que la orientación θ y la magnitud m del gradiente son

$$\begin{aligned} \theta &= \tan^{-1}\left(\frac{I_{x,y+1} - I_{x,y-1}}{I_{x+1,y} - I_{x-1,y}}\right) \\ m &= \sqrt{(I_{x+1,y} - I_{x-1,y})^2 + (I_{x,y+1} - I_{x,y-1})^2} \end{aligned}$$

Esta orientación canónica basada en el gradiente del punto clave, va a ser modificada luego ya que es necesario robustecer su cálculo.

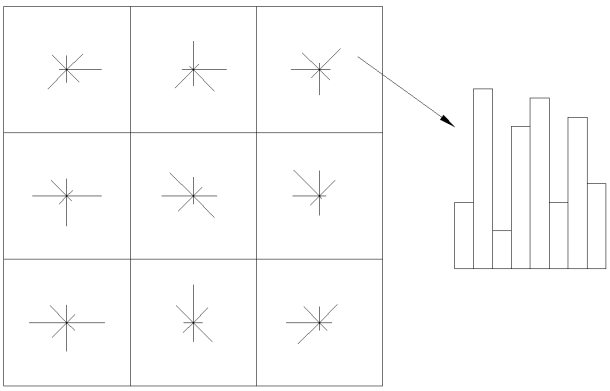

Fig. 4.3: Descriptor centrado en el punto clave. Los histogramas están representados en el círculo trigonométrico y mostramos la representación tradicional de uno de ellos para entender su conformación.

4.3. Construcción del descriptor

El proceso de construcción del descriptor (ver fig. 4.3) comienza tomando una ventana W_h, de $h \times h$ alrededor del punto clave de muestras de las magnitudes y orientaciones, utilizando la escala del punto para seleccionar el nivel de la imagen apropiado.

Una función gaussiana con $\sigma = h/2$ es usada para ponderar las magnitudes de cada punto. El propósito de esta ponderación es dar mayor estabilidad al descriptor, ya que los márgenes del mismo están más sujetos a distorsiones.

La ventana W_h se divide en r^2 regiones, como una matriz de $r \times r$ sub-ventanas; se la nota W_h^r. Estas sub-ventanas, de tamaño $\frac{h}{r} \times \frac{h}{r}$ modelan una columna de orientación (ver fig. 4.2), mediante la asociación con un histograma. Cada casillero de este histograma representa una orientación (una célula compleja).

Supongamos que sacamos dos fotos del mismo objeto, ubicando la cámara casi en la misma posición, variando la posición infinitesimalmente. Al ojo, las dos imágenes serán idénticas, pero es harto probable que no lo sean efectivamente. Es decir algunos pixels pueden cambiar ligeramente su tonalidad, o quedar ubicados ligeramente decalados. Los puntos que se "muevan" dentro de una sub-ventana, seguirán contribuyendo a ésta. Esto permite que

el descriptor sea tolerante a pequeñas distorsiones en la imagen local.

Se pondera la contribución de un punto en una sub-ventana de acuerdo a la distancia de éste al centro de la misma, para evitar problemas de vecindad entre sub-ventanas. Es un proceso similar al utilizado anteriormente sobre la totalidad del descriptor con la ponderación gaussiana.

Cada punto, luego de ponderar la magnitud de su gradiente en las formas ya detalladas, contribuye al histograma de la sub-ventana a la que pertenece, proyectando su gradiente en cada orientación y acumulando el valor en el casillero correspondiente. El algoritmo 4.1 ofrece una visión esquemática del proceso.

Algoritmo 4.1 Contrucción del descriptor local alrededor del punto $p = (x_p, y_p, \sigma_p)^T$ usando una ventana de W_h de $h \times h$

$\theta_p \leftarrow orientacion(gradiente(x_p, y_p, \sigma_p))$ // fija la orientación canónica
for all $(i, j)^T \in W_h$ **do**
  $\theta \leftarrow orientacion(gradiente(i, j, \sigma_p))$
  $m \leftarrow magnitud(gradiente(x, y, \sigma)) \cdot G(i, j, n/2)$
  $histo \leftarrow elegirHistograma(i, j)$ // determina a qué sub-ventana de $r \times r$ pertenece $(i, j)^T$
  $m \leftarrow m/distancia(histo, (i, j)^T)$
  $agregar(histo; m; \theta_p - \theta)$ // el ángulo está normalizado a $[0; 2\pi]$
end for

4.4. Serialización del descriptor

El descriptor es, en realidad, una representación vectorial de los valores de los histogramas. Esta representación tiene ciertas ventajas a la hora de comparar descriptores o calcular las distancias entre los mismos. En esta sección presentaremos un algoritmo para realizar esta conversión en la representación del descriptor.

Para lograr que el descriptor sea realmente invariante a rotaciones es escencial la correcta serialización del descriptor. Llamamos serialización al proceso de pasar de una representación conceptual (ver fig. 4.3) a una repsentación vectorial.

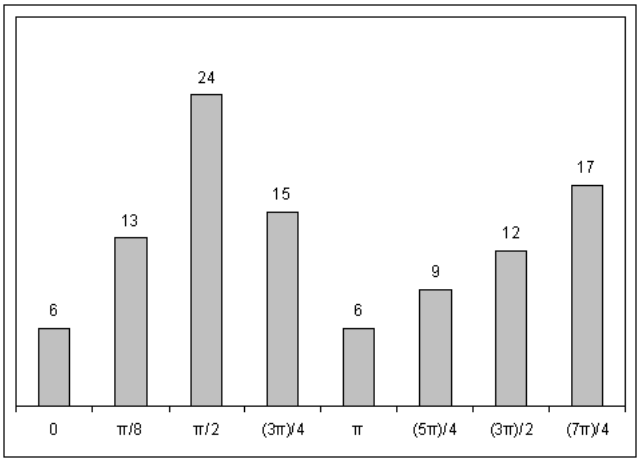

Fig. 4.4: La serialización del histograma de orientación de la figura es $\vec{V} = (6, 13, 24, 15, 6, 9, 12, 17)^T$.

El algoritmo trivial para esta tarea no es efectivo. Veamos la causa de esta aseveración con un ejemplo. Imaginemos que tenemos un punto clave en una imagen y utilizamos el algoritmo naif para serializar el descriptor en el punto P (que es una matriz de $M \times M$ histogramas):

$V \leftarrow <>$
for all $i = 1$ hasta m **do**
  for all $j = 1$ hasta m **do**
    $V \leftarrow V + Serial(H_{ij})$
  end for
end for

donde $+$ es la concatenación de vectores y $Serial(H)$ es la representación vectorial del histograma H (ver fig. 4.4).

Supongamos que el descriptor en el punto P es, esquemáticamente, $\vec{V} = (A, B, C, D, E, F, G, H, I)^T$ (ver fig. 4.5). Si rotamos la imagen con una matriz R_π, es de esperar que el descriptor en el punto $P' = RP$ sea el mismo que el de P pero rotado también π (ver fig. 4.6). La serialización del descriptor de P' arrojaría el vector $\vec{V'} = (I, H, G, F, E, D, C, B, A)^T$. Es evidente que comparar directamente $\vec{V}$ y $\vec{V'}$ no es posible. El objetivo de realizar la serialización es, desde un primer momento, facilitar la comparación entre descriptores; es

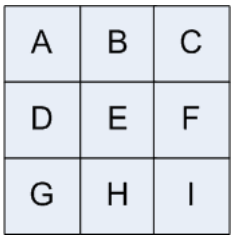

Fig. 4.5: La serialización del descriptor en el punto P de la figura utilizando el algoritmo trivial es $\vec{V} = (A, B, C, D, E, F, G, H, I)^T$.

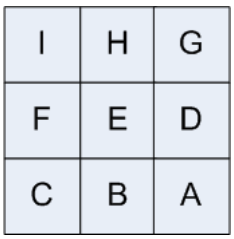

Fig. 4.6: La serialización del descriptor en el punto $P' = R_\pi P$ de la figura utilizando el algoritmo trivial es $\vec{V'} = (I, H, G, F, E, D, C, B, A)^T$.

claro por qué esta técnica no es útil.

Es entonces de vital importancia encontrar un algoritmo que subsane esta falencia.

Por un momento, pensemos que estamos trabajando con señales continuas. Sea θ la orientación del gradiente de un punto Q. Frente a una rotación $R(\alpha)$ de la imagen, la orientación del gradiente en el punto $Q' = R(\alpha)Q$ debería ser $\theta' = \theta + \alpha$. Podemos aprovechar esta propiedad para encontrar un método de serialización.

En las figuras 4.7 vemos los descriptores de los puntos Q y Q' junto a los respectivos vectores gradientes $\vec{g}$ y $\vec{g'}$.

Como la orientación del gradiente en el punto clave es invariante a rotaciones, podemos valernos entonces de la misma para serializar el descriptor.

Tenemos el descriptor W_h^r de tamaño tamaño h y formado por $r \times r$ histogramas. Definimos la función anillo A, que devuelve un conjunto de

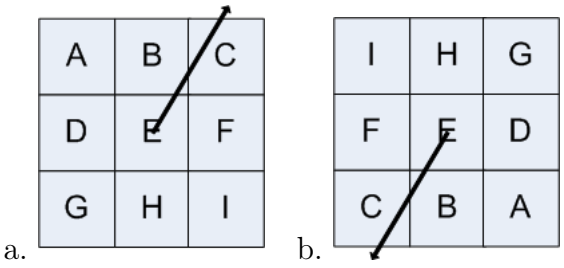

Fig. 4.7: El descriptor en los puntos Q (fig. a) y Q' (fig. b) junto a sus respectivos gradientes (representados por las flechas negras).

histogramas, como:

$$A(n) = \{h | h = W_h^r(i,j) \wedge (i = n \vee j = n)\} \tag{4.1}$$

donde $W_h^r(i,j)$ es el histograma en la posición (i,j) del descriptor.

El nuevo método de serialización consiste en procesar el descriptor por anillos. Buscamos el histograma por el que pasa el gradiente en el anillo exterior del descriptor y lo agregamos al vector. Luego agregamos todos los demás histogramas de ese anillo. Luego Pasamos al anillo interior siguiente y realizamos el mismo proceso. Por último ingresamos el histograma central, correspondiente al anillo $A(0)$. El algoritmo 4.2 es el resultante de esta serialización invariante a rotación. La fig. 4.8 muestra un ejemplo del mismo.

Algoritmo 4.2 Serialización del descriptor de $M \times M$ histogramas en el punto P

$i \leftarrow \lfloor \frac{M}{2} \rfloor$
while $i \geq 0$ **do**
  $H \leftarrow$ intersección del gradiente con $A(i)$.
  $V \leftarrow V+$ los histogramas de $A(i)$ desde H en sentido horario.
  $i \leftarrow i - 1$
end while

Un problema que surge al querer extrapolar esta idea a imágenes discretas es que la orientación del gradiente no es tan invariante a rotación como desearíamos. Esto se debe al muestreo y se hace más evidente cuando las

Fig. 4.8: La serialización a partir del gradiente aumenta la invarianza frente a rotaciones. La serialización, representada por la numeración de los histogramas, toma una forma de espiral. También se muestran con tramas diferentes cada anillo del descriptor.

rotaciones no son de 180º, por ejemplo. Necesitamos encontrar un método para robustecer la orientación canónica del descriptor.

Para lograr una mayor invarianza formaremos otro histograma, similar al utilizado para describir una subventana del descriptor. Se forma entonces un histograma de orientaciones a partir de las de los gradientes en todos los puntos dentro de una ventana alrededor del punto clave. Cada muestra agregada al histograma es pesada por la magnitud de su gradiente. El histrograma tendrá 36 casilleros, cubriendo los 360º. Los picos en el histograma corresponden a las direcciones dominantes de los gradientes locales. El máximo global M_G es detectado. También se crean nuevos puntos claves con los picos del histograma que superen el 80 % del valor M_G. Finalmente se usa una parábola para interpolar los valores de los casilleros alrededor de los picos, para refinar la precisión. Esta parábola resultará estar invertida y en su máximo se encontrará la orientación exacta de los picos (ver fig. 4.9). Si x_i es la orientación en la que se encuentra el pico, x_{i-1} la anterior y x_{i+1} la posterior, y h_i, h_{i-1} y h_{i+1} sus respectivos valores en el histograma, tenemos que:

$$f_i(x) = a_i x^2 + b_i x + c_i$$

donde a_i, b_i y c_i son las soluciones del sistema:

$$\begin{aligned} a_i x_{i-1}^2 + b_i x_{i-1} + c_i &= h_{i-1} \\ a_i x_i^2 + b_i x_i + c_i &= h_i \\ a_i x_{i+1}^2 + b_i x_{i+1} + c_i &= h_{i+1} \end{aligned}$$

Luego el máximo de f_i está en x_m y se cumple que,

$$\begin{aligned} f_i'(x) &= 2a_i x + b_i \\ f_i'(x_m) &= 2a_i x_m + b_i = 0 \\ x_m &= \frac{-b_i}{2a_i} \end{aligned}$$

De un descriptor, obtendremos varios descriptores, cada uno de ellos correspondiente a un máximo del histograma de orientaciones canónicas. La razón de esta replicación es importante: pequeños cambios en la orientación pueden hacer que el algoritmo de serialización tome como punto de partida un histograma u otro; teniendo más orientaciones, es menos probable que este fenómeno suceda en todas ellas.

4.5. Normalización

Finalmente, el vector $\vec{V}$ es normalizado para reducir los posibles efectos de cambios de iluminación. $\vec{V}$ es normalizado a longitud unitaria. Un cambio de contraste en el que cada píxel es multiplicado por un escalar c, será cancelado:

$$\frac{cv_i}{\sum_{i=0}^{o(r^2)-1} cv_i} = \frac{cv_i}{c\sum_{i=0}^{o(r^2)-1} v_i} = \frac{v_i}{\sum_{i=0}^{o(r^2)-1} v_i}$$

donde o es la cantidad de orientaciones de cada histograma.

Un cambio de brillo, en el que a cada píxel se le adiciona una constante, no afecta al descriptor, ya que fue obtenido a partir de diferencias de píxels.

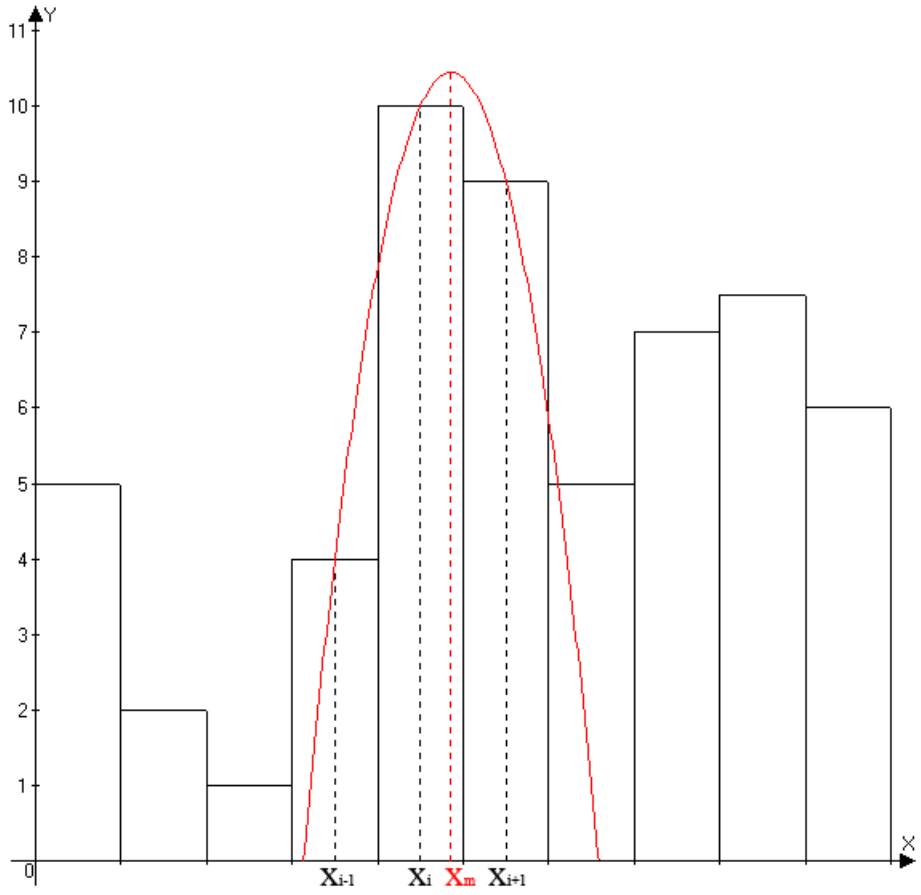

Fig. 4.9: Interpolación cuadrática del entorno de los picos del histograma. X_i es el máximo discreto y X_m es el máximo interpolado.

5. ALMACENAMIENTO Y RECUPERACIÓN DE DESCRIPTORES

El indexado basado en formas utiliza vectores de características de una imagen para acceder a una estructura índice, recuperando rápidamente las posibles correspondencias con los modelos de los objetos almancenados en el mismo. Este procedimiento es mucho más eficiente temporalmente que la alternativa de la búsqueda exhaustiva, pero tiene como contrapartida el armado del índice y la memoria extra para almacenarlo.

Esta visión se corresponde con la búsqueda de vecinos más cercanos (VC) a un punto dado. La mayoría de los acercamientos utilizan tablas de hash para esta tarea aunque es ampliamente conocido que las estructuras de árbol hacen el trabajo más eficientemente.

Esto puede explicarse por el hecho de que estas técnicas se aplican en espacios de baja dimensionalidad y, en éstos las tablas de hash funcionan adecuadamente. Cuando el número de objetos es pequeño, alcanza con utilizar vectores de descripción de baja dimensionalidad para diferenciarlos, a medida que el número de objetos crece, es necesario contar un mayor grado de discriminación entre ellos. Esto implica un aumento en la dimensión de los vectores. En estos espacios de gran dimensionalidad, las técnicas de hashing se vuelven inutilizables. En resumen, y desafortunadamente, la complejidad temporal de la búsqueda de vecinos más cercanos, depende exponencialmente de la dimension del espacio.

Una estructura de datos que fue utilizada extensivamente son los árboles k-d [6] donde los efectos de la dimensionalidad del espacio no son tan severos. La falta de eficiencia en las tablas de hash es debida principalmente al hecho de que los casilleros son fijos, mientras que los de los árboles son adaptativos a la densidad local de puntos almacenados. Por lo tanto, en algunos casos (zonas de alta densidad de puntos), un acercamiento basado en hashing debe-

rá realizar una larga búsqueda lineal a través de todos los puntos contenidos en el casillero en el que el punto de entrada cae. No es el caso de los árboles, en dónde la búsqueda no se realiza linealmente en ningún nivel.

5.1. Árboles k-d

En esta sección, presentaremos la estructura de datos conocida como *árbol k-d*, para luego, en la sección siguiente, detallar algunas mejoras a los algoritmos involucrados en la búsqueda.

Estructura

Contamos con un conjunto $\mathcal{C}$ de N puntos en $\mathbb{R}^k$. Buscamos organizar estos puntos de manera que puedan ser encontrados y accedidos rápidamente. En particular, dado un punto p que puede pertenecer o no a $\mathcal{C}$, queremos obtener el punto $p' \in \mathcal{C}$, que cumpla con la siguiente restricción:

$$p' = \arg\min_{p_i \in \mathcal{C}} d(p, p_i).$$

con d alguna distancia apropiada. En otras palabras, buscamos el punto en $\mathcal{C}$, más cercano a p.

Para realizar una búsqueda que cumpla con la restricción y sea eficiente, se utiliza una técnica de *divide & conquer*. Éstas se basan en dividir el espacio de búsqueda en subespacios, resolver el problema en éstos y combinar apropiadamente los resultados parciales. Esa idea puede realizarse recursivamente hasta que, en cada subespacio, el problema es trivial.

Definimos el conjunto $\mathcal{C}_j$ como:

$$\mathcal{C}_j = \{w \mid p \in \mathcal{C} \wedge p_j = w\} \tag{5.1}$$

donde p_j es el valor del vector p en la dimensión j

La estrategia para armar un árbol k-d es la siguiente:

- dividir el conjunto $\mathcal{C}$ en dos partes, $\mathcal{C}^1$ y $\mathcal{C}^2$, partiéndolo en la mediana m de la dimensión i con mayor varianza:

$$\begin{aligned} i &= \arg \max_{1 \leq j \leq k} Var(\mathcal{C}_j) \\ m_i &= mediana(\mathcal{C}_i) \\ \mathcal{C}^1 &= \{p \mid p \in \mathcal{C} \wedge p_i \leq m_i\} \\ \mathcal{C}^2 &= \{p \mid p \in \mathcal{C} \wedge p_i > m_i\} \end{aligned} \tag{5.2}$$

  Notar que $\#\mathcal{C}^1 = \#\mathcal{C}^2$.

- Se crea un árbol binario A en cuya raíz se almacenan m e i. El subárbol izquierdo de A es el resultado de aplicar recursivamente este procedimiento sobre $\mathcal{C}^1$ y el derecho, sobre $\mathcal{C}^2$.

El algoritmo 5.1 ilustra el proceso anteriormente descripto.

Algoritmo 5.1 KD($\mathcal{C}$): contrucción del árbol k-d del conjunto de puntos k-dimensionales $\mathcal{C}$

```
if #C = 0 then
  return nodoVacio
end if
if #C = 1 then
  return hoja(p)
end if
i ← arg máx_{1≤ j≤k} Var(C_j)
m_i ← mediana(C_i)
C^1 ← {p | p ∈ C ∧ p_i ≤ m_i}
C^2 ← {p | p ∈ C ∧ p_i > m_i}
n ← nodo(i, m_i) // construye el nodo almacenando i y m_i
return bin(KD(C^1), n, KD(C^2)) // bin construye el árbol KD
```

Esto termina creando un árbol binario balanceado, ya que en cada nodo, ambos subárboles tienen la misma cantidad de elementos. La profundidad del árbol es entonces

$$d = \lceil \log_2 N \rceil. \tag{5.3}$$

Las hojas de un árbol k-d forman una partición completa de $\mathcal{C}$, con la interesante propiedad de que cada región (denotada por un nodo) es más pequeña si tiene más densidad de puntos y más grande si tiene menor densidad.

Ejemplo

Para ilustrar el proceso de creación de un árbol k-d introduciremos un ejemplo, que iremos analizando en detalle. Supongamos que tenemos un conjunto inicial $\mathcal{C}$ tal que sus puntos están en $\mathbb{R}^2$ (ver tab. 5.1 y fig. 5.1) y queremos organizarlo para realizar búsquedas VC.

$\mathcal{C}_X$	16	21	40	41	51	53	63	70	84	97
$\mathcal{C}_Y$	26	21	58	23	86	96	83	86	56	35

Tab. 5.1: Conjunto $\mathcal{C}$ de puntos a organizar en un árbol k-d.

Primero debemos determinar los límites de la región en la que están contenidos los puntos de $\mathcal{C}$. Obviamente, el mínimo y el máximo en cada dimensión son las cotas de la misma (ver fig. 5.2):

$$\begin{aligned} \text{mín}\,(\mathcal{C}_X) &= 16 \\ \text{máx}\,(\mathcal{C}_X) &= 97 \\ \text{mín}\,(\mathcal{C}_Y) &= 21 \\ \text{máx}\,(\mathcal{C}_Y) &= 96. \end{aligned}$$

Ahora procedemos a partir los datos. Calculamos las varianzas en cada dimensión,y partimos el conjunto $\mathcal{C}$ en la mediana de la dimensión con mayor varianza.

$$\begin{aligned} \text{Var}(\mathcal{C}_X) &= 665,82 \\ \text{Var}(\mathcal{C}_Y) &= 864,22 \end{aligned}$$

como $\text{Var}(\mathcal{C}_X) < \text{Var}(\mathcal{C}_Y)$ entonces

$$m_Y = 56$$

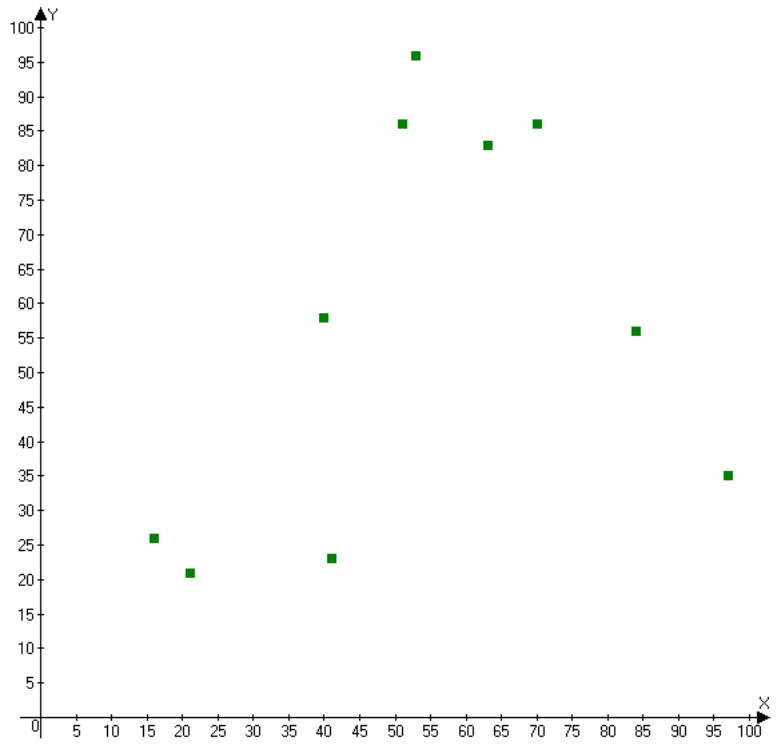

Fig. 5.1: Distribución de los puntos de la tabla 5.1 en el espacio.

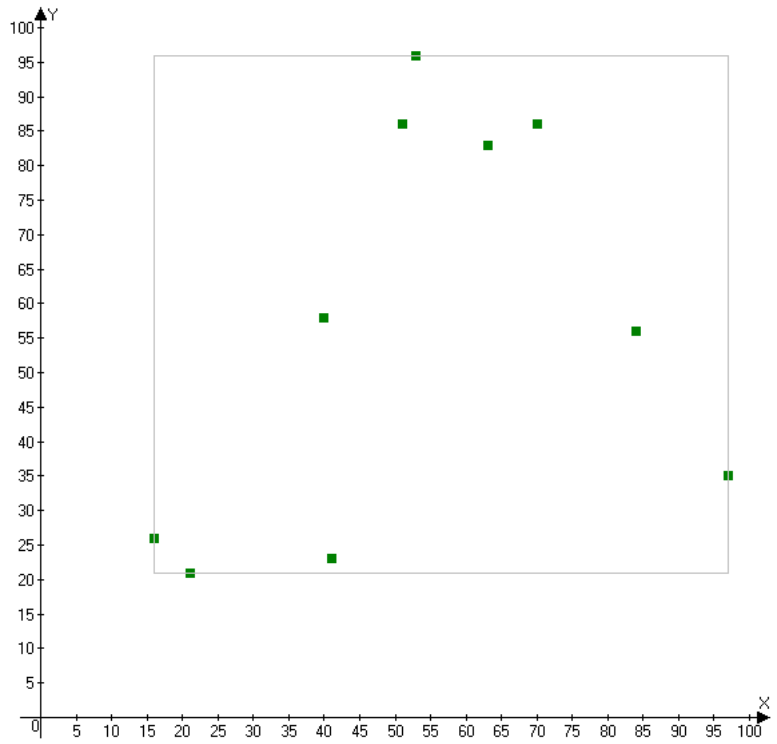

Fig. 5.2: Determinación de la región que ocupan los puntos de la tabla 5.1

El punto $(84, 56)$ divide $\mathcal{C}$, como lo muestra la fig. 5.3. Tenemos una partición inicial, lo que determina (esquemáticamente) el árbol en la fig. 5.4 (con el propósito de facilitar la comparación, las dos figuras comparten los mismos colores para identificar las regiones).

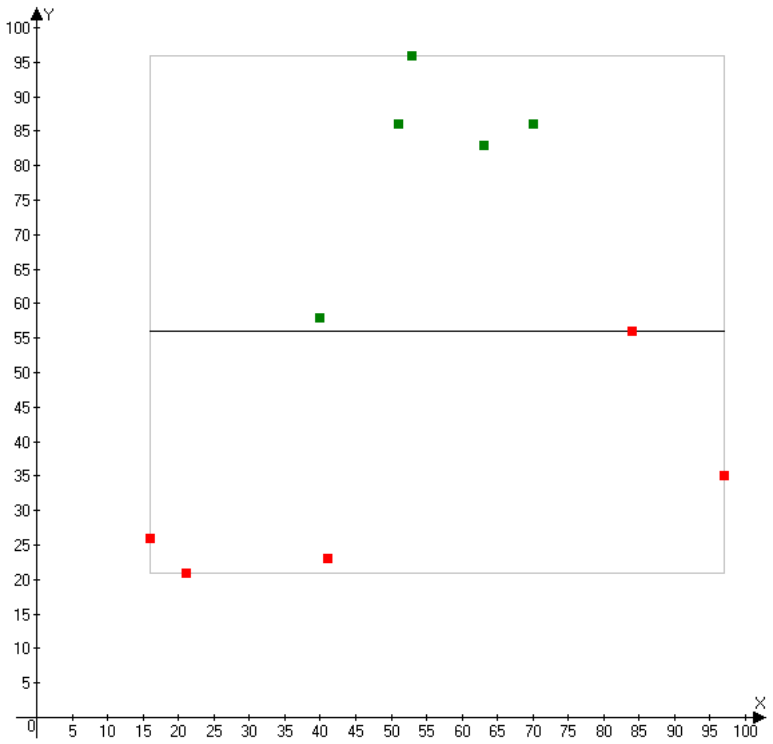

Fig. 5.3: El conjunto es partido en el punto $(84, 56)$, ya que 56 es la mediana de Y, la dimensión de mayor varianza.

Ahora debemos analizar separadamente cada una de las mitades de $\mathcal{C}$, obtenidas en el paso anterior. Calculamos la varianza en cada dimensión nuevamente y, luego, la mediana (ver tab. 5.2). La partición y el árbol resultantes pueden verse en fig. 5.5 y fig. 5.6

Nuevamente, analizamos separadamente cada una de las partes de $\mathcal{C}$, obtenidas en el paso anterior. Calculamos la varianza en cada dimensión y, luego, la mediana (ver tab. 5.3, la mediana está resaltada en negrita). La partición y el árbol resultantes pueden verse en fig. 5.7 y fig. 5.8

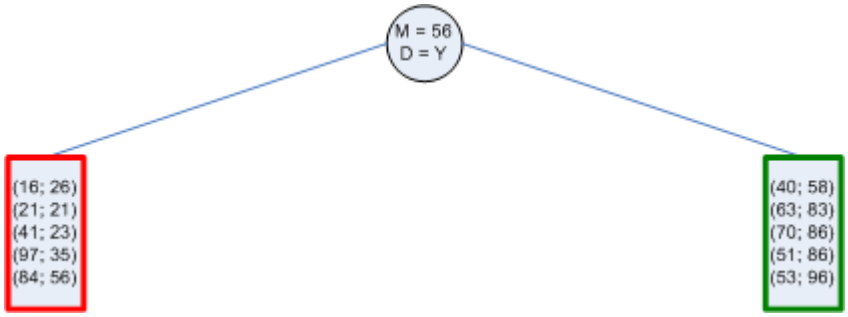

Fig. 5.4: Vista esquemática del árbol resultante de la primera partición, ver fig. 5.3.

a.

$\mathcal{C}_X$	16	21	**41**	84	97	$\mathrm{Var}(\mathcal{C}_X) = 1356,7$
$\mathcal{C}_Y$	26	21	23	56	35	$\mathrm{Var}(\mathcal{C}_Y) = 205,7$

b.

$\mathcal{C}_X$	40	63	51	70	53	$\mathrm{Var}(\mathcal{C}_X) = 169,66$
$\mathcal{C}_Y$	58	83	**86**	86	96	$\mathrm{Var}(\mathcal{C}_Y) = 201,2$

Tab. 5.2: Las dos mitades del conjunto de la tabla 5.1, resultantes de la primera partición del mismo (ver fig. 5.3).

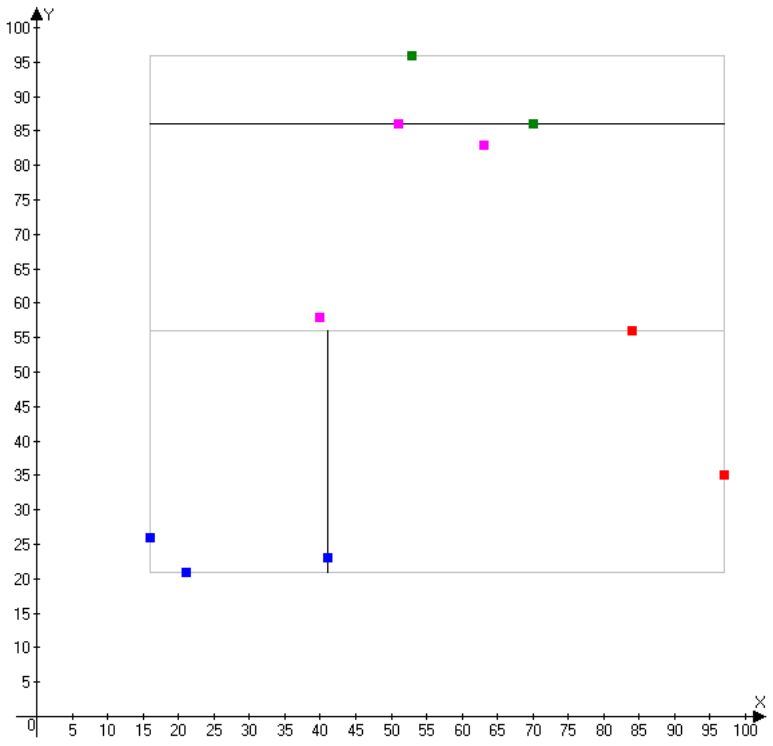

Fig. 5.5: El conjunto es partido en los puntos $(41, 23)$ y $(51, 86)$.

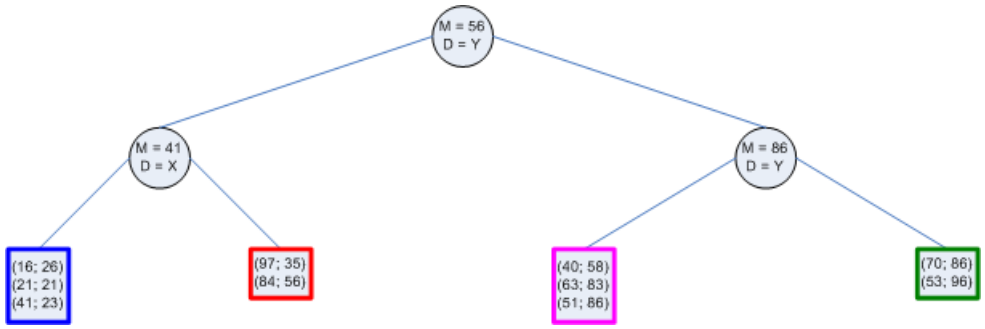

Fig. 5.6: Vista esquemática del árbol resultante de la segunda partición, ver fig. 5.5.

a.

$\mathcal{C}_X$	16	**21**	41
$\mathcal{C}_Y$	26	21	23

$\text{Var}(\mathcal{C}_X) = 175$
$\text{Var}(\mathcal{C}_Y) = 6,33$

b.

$\mathcal{C}_X$	84	97
$\mathcal{C}_Y$	56	**35**

$\text{Var}(\mathcal{C}_X) = 84,5$
$\text{Var}(\mathcal{C}_Y) = 220,5$

c.

$\mathcal{C}_X$	40	63	51
$\mathcal{C}_Y$	58	**83**	86

$\text{Var}(\mathcal{C}_X) = 75,66$
$\text{Var}(\mathcal{C}_Y) = 236,33$

d.

$\mathcal{C}_X$	70	**53**
$\mathcal{C}_Y$	86	96

$\text{Var}(\mathcal{C}_X) = 144,5$
$\text{Var}(\mathcal{C}_Y) = 50$

Tab. 5.3: Las cuatro partes del conjunto de las tablas 5.2, resultantes de la segunda partición (ver fig. 5.5).

a.

$\mathcal{C}_X$	**16**	21
$\mathcal{C}_Y$	26	21

$\text{Var}(\mathcal{C}_X) = 12,5$
$\text{Var}(\mathcal{C}_Y) = 12,5$

b.

$\mathcal{C}_X$	40	63
$\mathcal{C}_Y$	**58**	83

$\text{Var}(\mathcal{C}_X) = 264,5$
$\text{Var}(\mathcal{C}_Y) = 312,5$

Tab. 5.4: Las partes restantes del conjunto de las tablas 5.3, resultantes de la tercera partición (ver fig. 5.7).

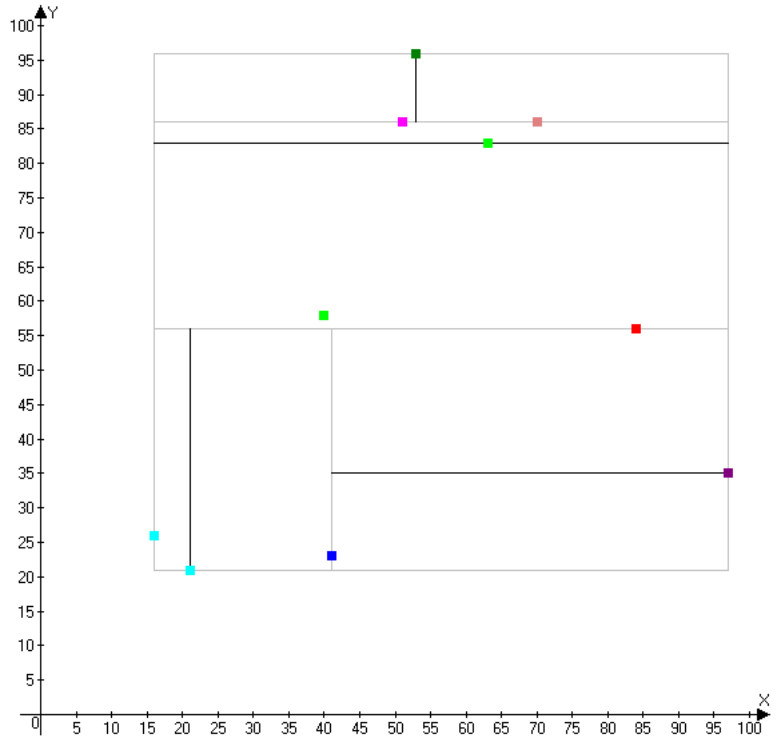

Fig. 5.7: El conjunto es partido en los puntos $(21, 21)$, $(97, 35)$, $(63, 83)$ y $(53, 96)$.

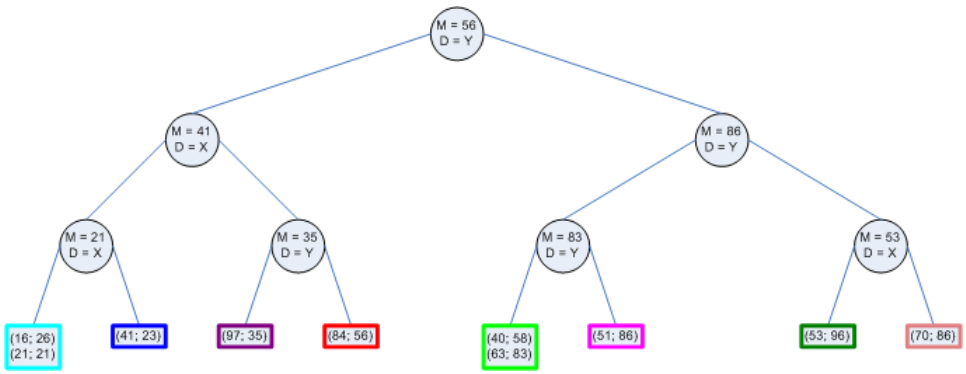

Fig. 5.8: Vista esquemática del árbol resultante de la tercera partición, ver fig. 5.7.

Por último, analizamos separadamente cada una de las partes de $\mathcal{C}$, obtenidas en el paso anterior. Calculamos la varianza en cada dimensión y, luego, la mediana (ver tab. 5.3, la mediana está resaltada en negrita). La partición y el árbol resultantes pueden verse en fig. 5.7 y fig. 5.8

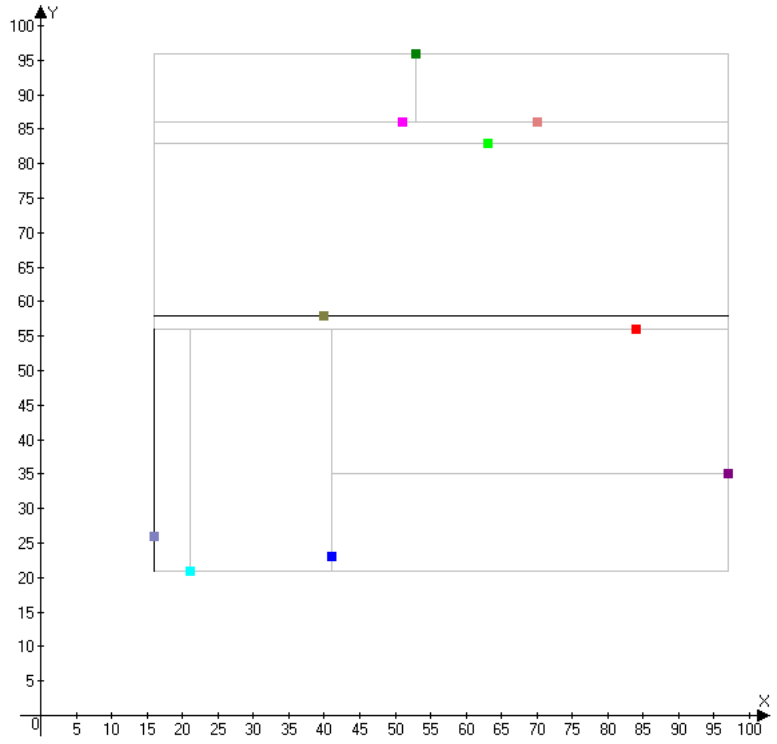

Fig. 5.9: El conjunto es partido en los puntos $(16, 26)$ y $(40, 58)$.

Búsqueda

Para buscar el VC a un punto dado q en un conjunto k dimensional se construye el árbol k-d A y se utiliza una técnica de backtracking. Primero, A es atravesado para encontrar la región R que contiene a q. Esto requiere d (ver eq. 5.3) comparaciones, y en general, el punto $q' \in R$ es una buena aproximación al VC buscado. En la etapa de backtracking, ramas enteras del árbol pueden ser *podadas* si la región del espacio que representan tiene una distancia a q mayor a la distancia entre q' y q. Si la distancia es menor,

se busca dentro de la región para ver si hay otro punto q'' más cercano a q que q'. La búsqueda termina cuando se han cotejado todas las ramas que inicialmente no fueron exploradas.

Cuando se baja por un subárbol, se agrega el otro a una cola de prioridad de ramas no exploradas. Esta cola ordena las ramas en orden creciente de distancia a q.

Este proceso es realmente eficiente en espacios de baja dimensionalidad, perdiendo rápidamente eficiencia a medida que se incrementa la dimensionalidad. Esto se debe a que el número de ramas a cotejar crece velozmente. Es interesante notar que, potencialmente, el número de puntos de una región que pueden ser el VC a q es pequeño, dado que su ubicación está limitada una región muy estrecha en la periferia de la región (ver fig. 5.11).

Se analiza un gran número de regiones cuando, en realidad es muy poco probable que contengan mejores aproximaciones que la estimación inicial q'.

Ejemplo

Supongamos que tenemos el subárbol del ejemplo anterior y deseamos buscar el VC al punto $p = (60, 65)$. La búsqueda se inicia en la raíz del árbol (ver fig. 5.12). Como la dimensión de mayor varianza del nodo es Y, comparamos la mediana 56 con $p_Y = 65$. Bajamos entonces por el subárbol derecho y encolamos el izquierdo a la cola de subárboles no explorados.

Un proceso similar se muestra en las figuras 5.13, 5.14, 5.15 y 5.16 hasta encontrar el punto de la región a la que pertence p. En este caso, la región es R19 y el punto $p' = (63, 83)$ (ver fig. 5.17). La cola de ramas no visitadas es $Q = [R2, R18, R13, R7]$.

Como la distancia de p a R2 es menor que la de p a p' (ver fig. 5.18), recorremos R2 con el mismo proceso que para encontrar p' y llegamos a $p'' = (84, 56)$ (ver fig. 5.19). Se fueron encolando otras ramas y finalmente $Q = [R4, R18, R13, R7, R10]$. Tenemos que $d(p, p') < d(p, p'')$ entonces descartamos p'' y nuestro VC sigue siendo p' (ver fig. 5.20) . Luego analizamos el resto de la cola Q:

1. $d(p, p') < d(p, R4)$ en la fig. 5.21
2. $d(p, p') < d(p, R18)$ en la fig. 5.22

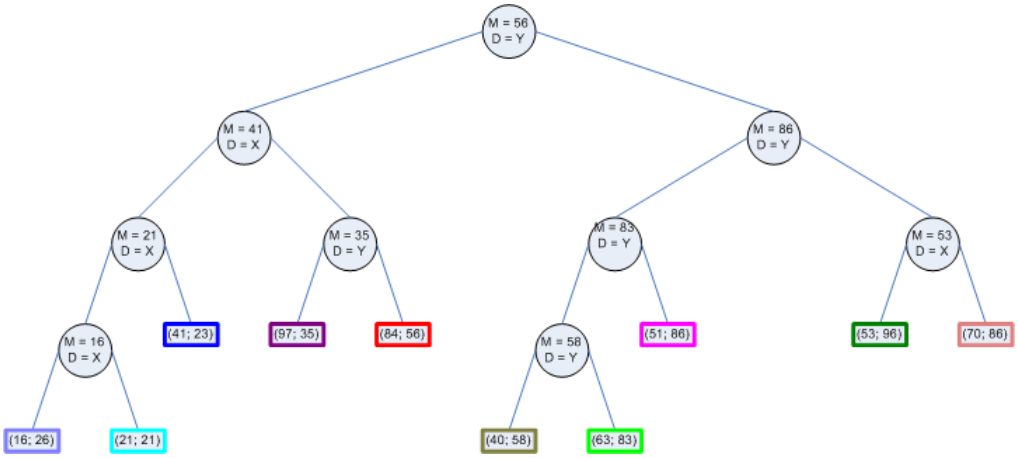

Fig. 5.10: Vista esquemática del árbol resultante de la cuarta partición, ver fig. 5.9.

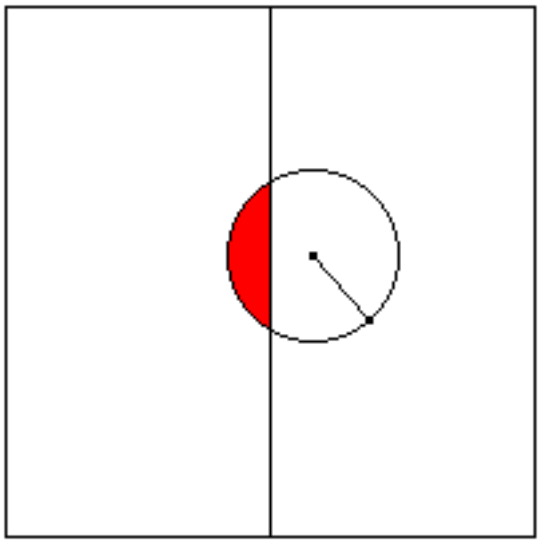

Fig. 5.11: El punto de entrada es el centro del círculo y el VC es el que se encuentra en la misma región, sobre el círculo. La región en rojo es la única en la que pueden estar los puntos de la región izquierda que mejoran la aproximación actual del VC.

3. $d(p, p') < d(p, R13)$ en la fig. 5.23

4. $d(p, p') < d(p, R10)$ en la fig. 5.24

El algoritmo no se adentra en ninguna de estas regiones porque no pueden contener puntos cuyas distancias a p sean menores que $d(q, q')$. El VC es finalmente $q' = (63, 83)$, que había sido encontrado antes de la etapa de backtracking.

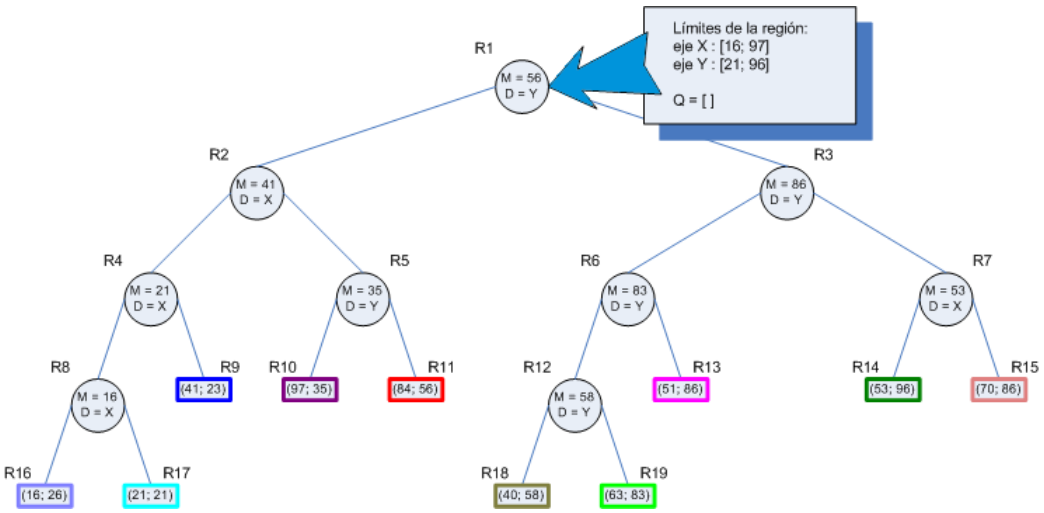

Fig. 5.12: Inicio de la búsqueda del VC al punto $p = (60, 65)$. M es la mediana y D es dimensión soreb la que se toma esa mediana

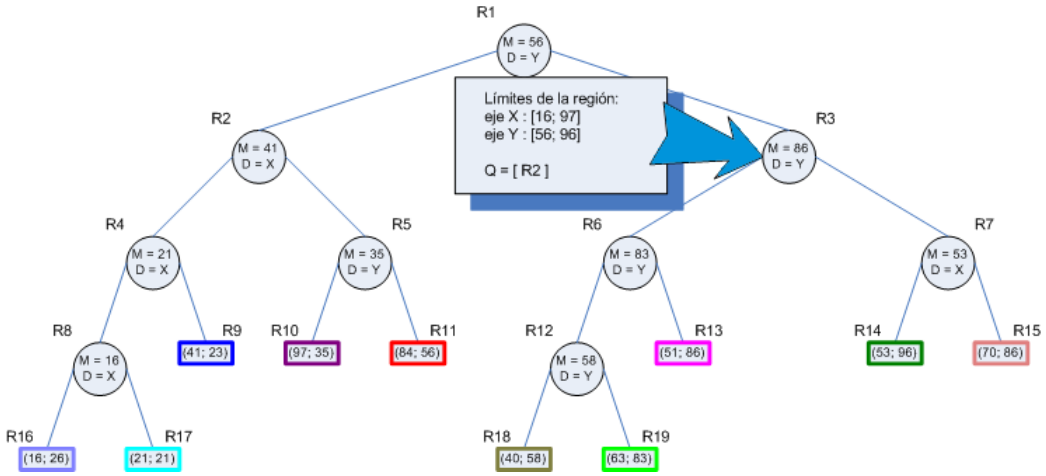

Fig. 5.13: Búsqueda en la región R3 del VC a $p = (60, 65)$. R2 fue agregada a la cola Q de ramas no visitadas.

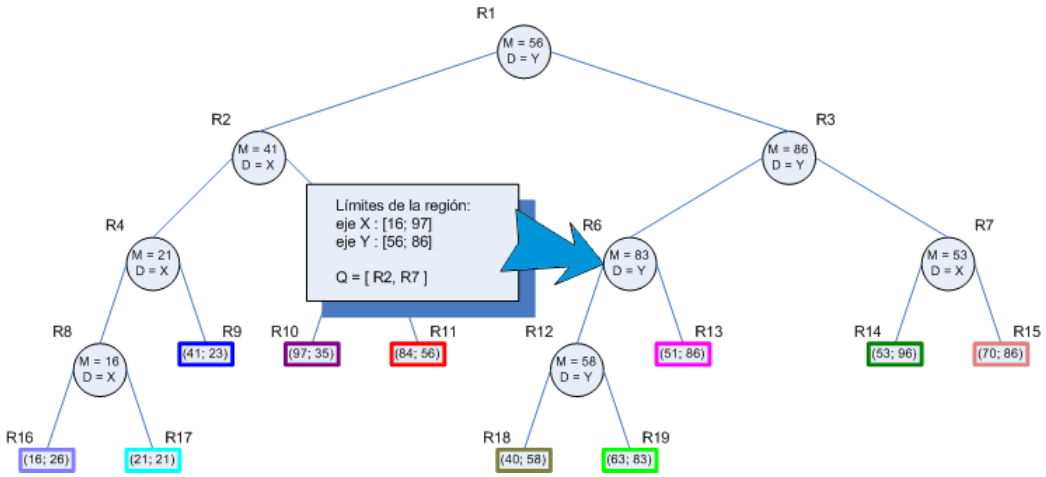

Fig. 5.14: Búsqueda en la región R6 del VC a $p = (60, 65)$. R6 fue agregada a la cola Q de ramas no visitadas.

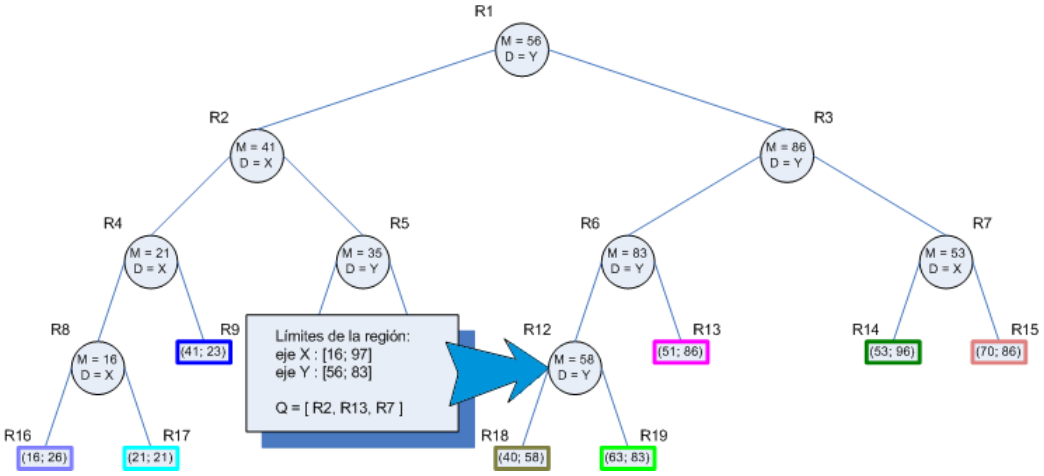

Fig. 5.15: Búsqueda en la región R12 del VC a $p = (60, 65)$. R13 fue agregada a la cola Q de ramas no visitadas.

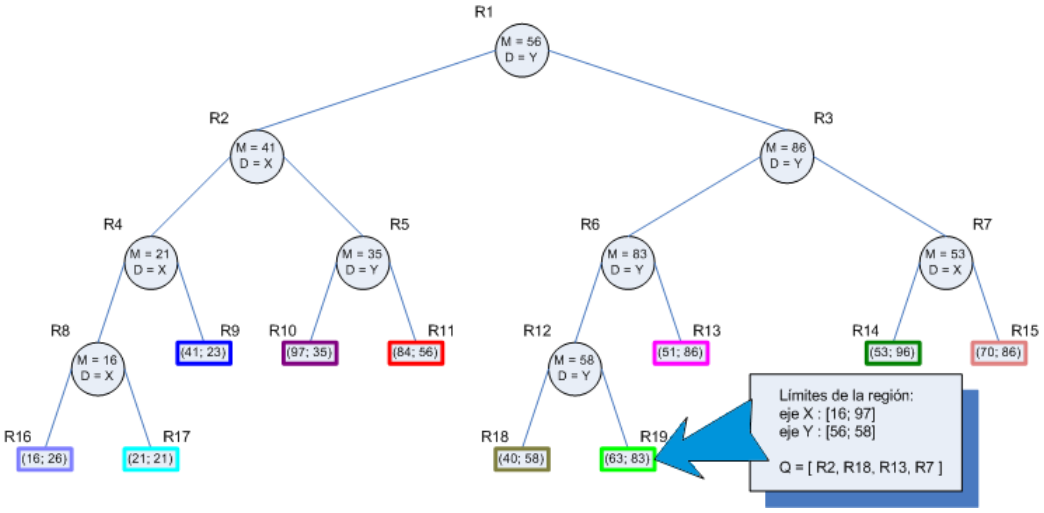

Fig. 5.16: Búsqueda en la región R19 del VC a $p = (60, 65)$. R18 fue agregada a la cola Q de ramas no visitadas. Tenemos que $p \in R19$ y $p' = (63, 83)$.

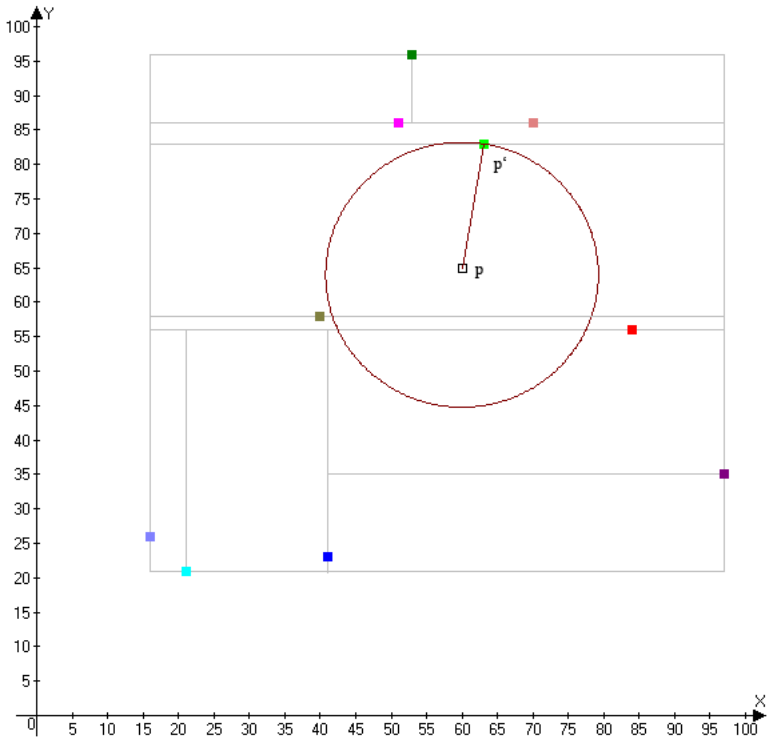

Fig. 5.17: El punto blanco es $p = (60, 65)$ y el verde sobre el círculo es $p' = (63, 83)$.

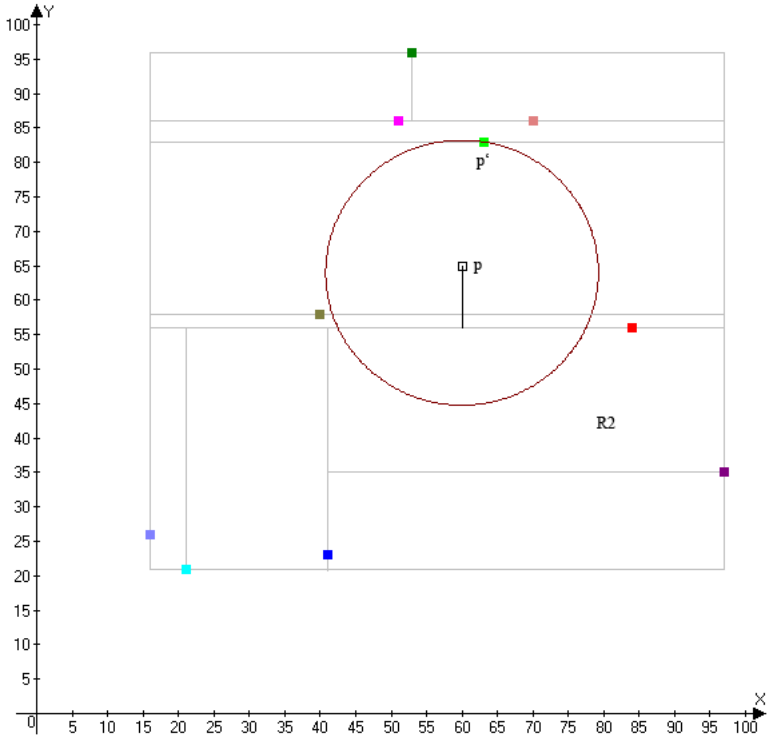

Fig. 5.18: Comparación de $d(p, p')$ y $d(p, R2)$.

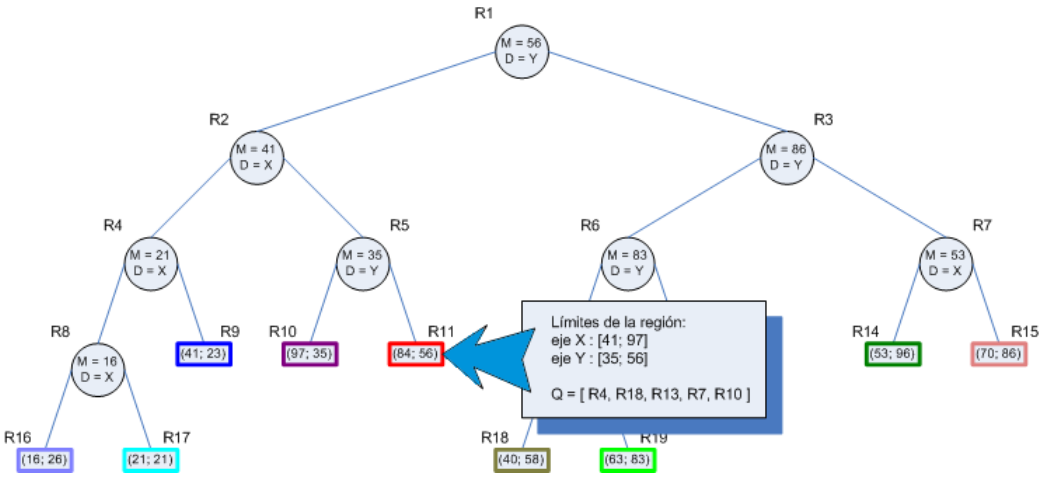

Fig. 5.19: Buscamos p'' en R2 y llegamos a $p'' = (84, 56)$. Fueron agregadas a Q las regiones R4 y R10.

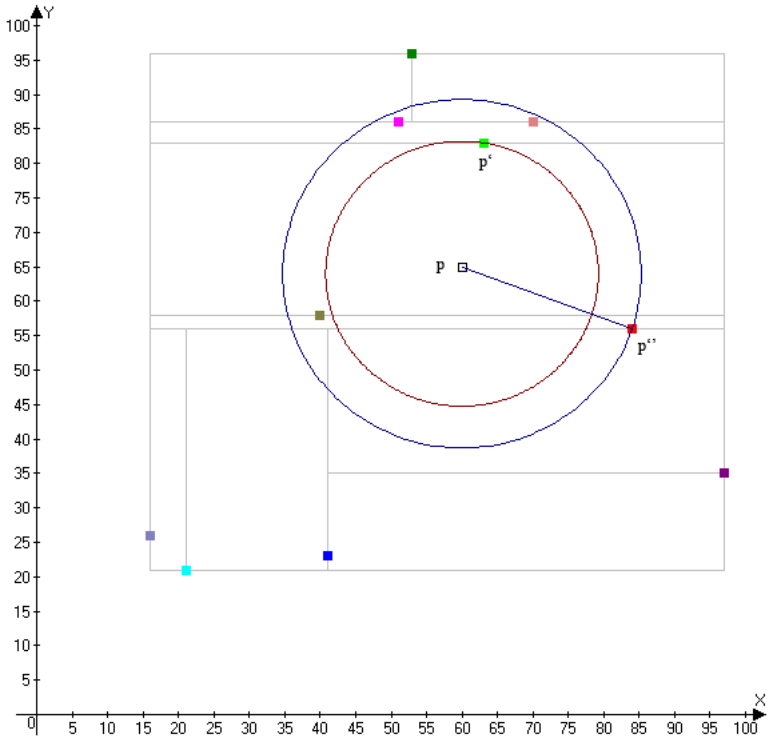

Fig. 5.20: Comparación de $d(p, p')$ y $d(p, p'')$.

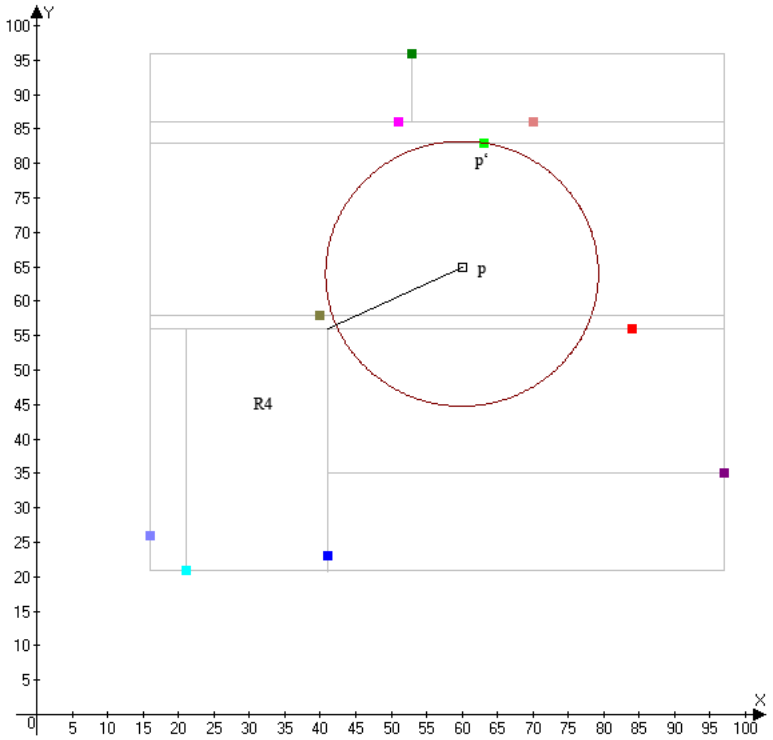

Fig. 5.21: Comparación de $d(p, p')$ y $d(p, R4)$.

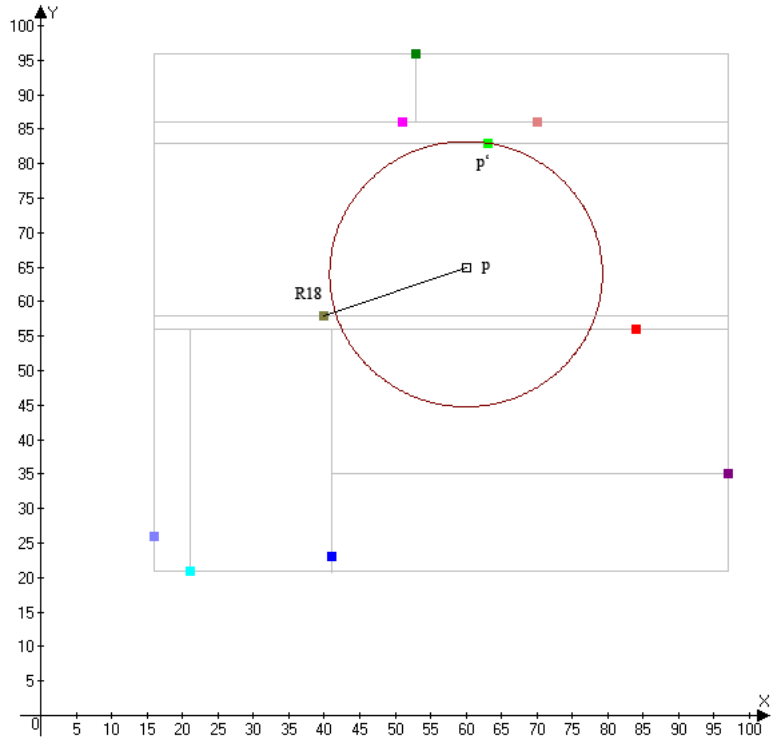

Fig. 5.22: Comparación de $d(p, p')$ y $d(p, R18)$. $R18$ es una hoja en el árbol y, por lo tanto, un punto.

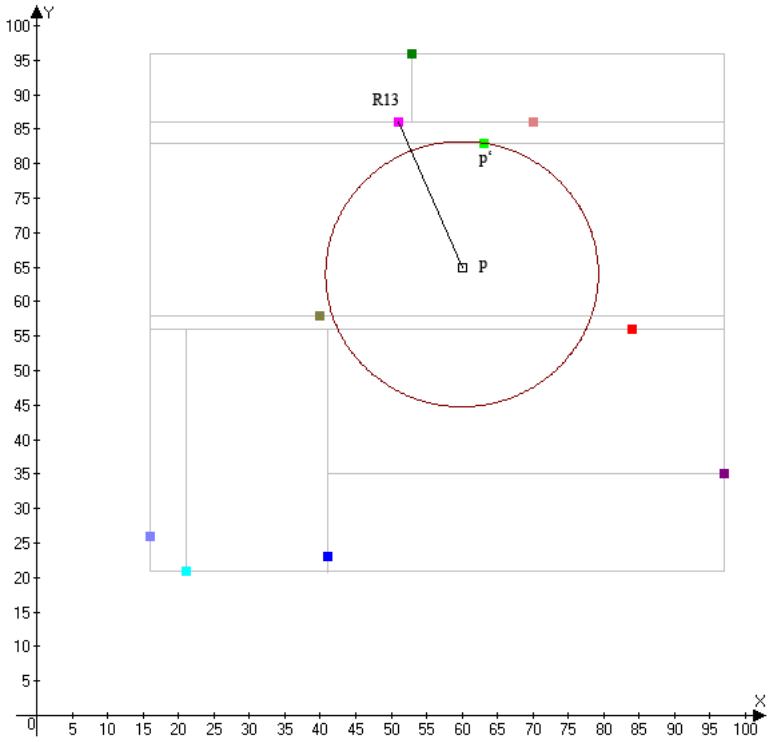

Fig. 5.23: Comparación de $d(p, p')$ y $d(p, R13)$. $R13$ es una hoja en el árbol y, por lo tanto, un punto.

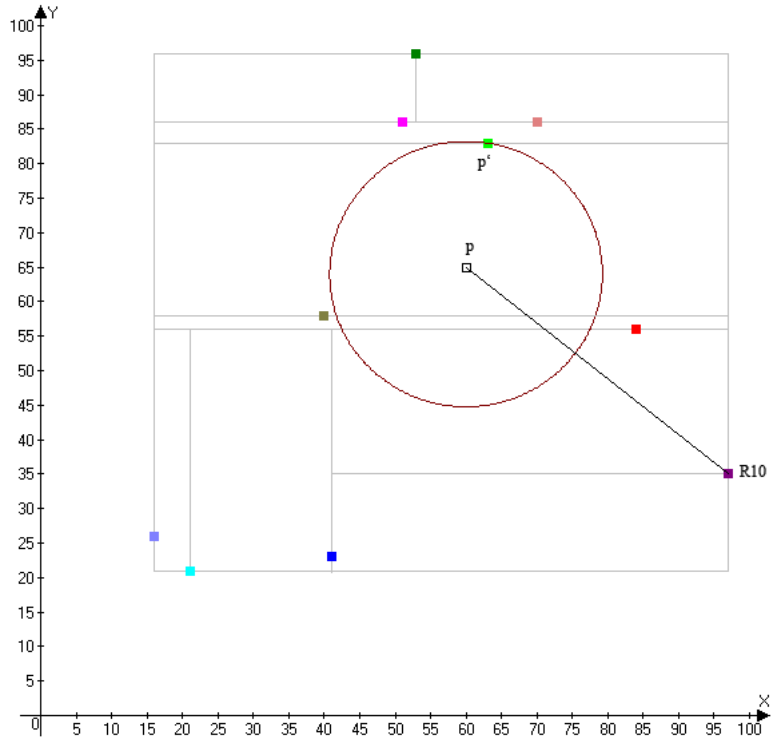

Fig. 5.24: Comparación de $d(p, p')$ y $d(p, R10)$. $R10$ es una hoja en el árbol y, por lo tanto, un punto.

5.2. Conjuntos de correspondencias

El armado del árbol k-d de un conjunto de descriptores vectoriales, junto al algoritmo de búsqueda del VC, permite la construcción de un conjunto de correspondencias a partir de dos fuentes de descriptores. Estas fuentes pueden ser dos imágenes o una imagen y una base de datos, por ejemplo.

Con los descriptores serializados (ver §4.4) de una de las fuentes, construímos un árbol k-d A. Luego tomamos cada uno de los descriptores D de la otra fuente y buscamos el VC D' de cada uno de ellos en A. Cada par (D, D') es agregado al conjunto de correspondencias. El algoritmo resultante es el 5.2.

Algoritmo 5.2 Armado de un conjunto de correspondencias C a partir de dos conjuntos de vectores S_1 y S_2.

$A \leftarrow KD(S_1)$ // construye el árbol k-d
$C \leftarrow \emptyset$
for all $D \in S_2$ **do**
  $D' \leftarrow VC(D)$ // devuelve el VC de D
  $C \leftarrow C \cup (D, D')$
end for
return C

6. ESTIMADORES ROBUSTOS

En el §5.2 hemos mostrado cómo es posible armar un conjunto de correspondencias $\{a_i \leftrightarrow b_i\}$ entre puntos de dos fuentes distintas (por ej. entre dos cuadros o entre un cuadro y una base de datos). La intención es estimar una transformación que haga que cada a_i se corresponda con cada b_i. Existe una fuente de error que proviene de la medición de la posición de los puntos, por lo tanto podemos escribir:

$$b_i = Ta_i + \varepsilon_p \tag{6.1}$$

donde T es la transformación que se desea estimar y ε_p es un vector aleatorio que modela el error, con distribución $N(0, \sigma)$.

Para modelar en forma completa los errores existentes en el conjunto de correspondencias, no basta con suponer que la única fuente de error es la de la fórmula 6.1. Es necesario considerar también los errores en el armado del conjunto de correspondencias. En otras palabras, dentro del conjunto de correspondencias, hay elementos que son *outliers* (poseen errores adicionales, diferentes y posiblemente no modelados) como lo indica la ecuación siguiente:

$$b_j = Ta_j + \varepsilon_p + \varepsilon_c$$

donde ε_c representa el error en la construcción del conjunto de correspondencias.

Los *outliers* pueden afectar severamente la transformación estimada y, por ende, deben ser encontrados y descartados. Es necesario determinar qué correspondencias, dentro del conjunto, responden a la ecuación 6.1 (*inliers*) y cuáles no (*outliers*). Luego, podremos estimar T utilizando únicamente los *inliers*. Ésta estimación es llamada *robusta* ya que no es susceptible al efecto de los *outliers*.

La mayoría de las técnicas para estimación de parámetros, como mínimos cuadrados, optimizan (con algún criterio elegido a priori) el ajuste de un modelo, o descripción funcional, a la *totalidad* de los datos. Estas técnicas no tienen mecanismos internos para detectar y rechazar errores groseros. Se basan, por el contrario, en la suposición de que la cantidad de valores aberrantes es muy pequeña comparada a la de valores "satisfactorios". La diferencia en la cardinalidad de los conjuntos de *inliers* y de *outliers* será suficientemente grande para lograr suavizar cualquier desviación.

A continuación presentaremos dos métodos que incorporan técnicas para manejar errores groseros. Uno, utilizando la transformación de Hough generalizada [2] y otro, utilizando un algoritmo llamado Random Sample Consensus (RANSAC) [5]. Ambos se basan en el establecimiento de grupos de consenso a partir de los cuales poder realizar las estimaciones. En §7 se mostrarán algunos ejemplos del comportamiento de ambos métodos.

6.1. Transformación de Hough

En la transformación de Hough se busca estimar un modelo T con g grados de libertad a partir de un conjunto de correspondencias. Si llamamos t_i a cada uno de los parámetros correspondientes a los g grados de libertad ($1 \leq i \leq g$), la idea básica es discretizar el espacio de valores que pueden tomar los t_i. Si se discretiza el espacio de cada t_i en k_i valores distintos, el espacio g-dimensional discretizado conforma una hipermatriz H de $k_1 \times \cdots \times \ k_i \times \cdots \times k_g$.

Por ejemplo, supongamos que tenemos un conjunto S de correspondencias $\{x \leftrightarrow y\}$ de puntos en $\mathbb{R}$ y queremos estimar una recta que pasa por ellas. El espacio de rectas puede ser parametrizado, con dos parámetros θ y r, teniendo en cuenta que cada recta puede expresarse, en coordenadas polares, como $r = x\cos\theta + y\sin\theta$. Estos parámetros se discretizan en k_θ y k_r valores, respectivamente, formando la matriz H de $k_\theta \times k_r$. En cada posición (θ_i, r_j) de la matriz, se almacenan las correspondencias de S que se encuentran en la recta $r_j = x\cos\theta_i + y\sin\theta_i$.

$$H(i,j) = \{(x \leftrightarrow y) \in S \mid r_j = x\cos\theta_i + y\sin\theta_i + \epsilon\}$$

donde $1 \leq i \leq k_\theta$, $1 \leq j \leq k_r$ y ϵ es la tolerancia aceptada para la divergencia

de un punto a la recta. En cada una de estas posiciones queda almacenado el conjunto *de consenso* para el modelo (θ_i, r_j). Finalmente, elegimos el modelo con el mayor conjunto de consenso para nuestra estimación de T.

Esta misma idea es usada para estimar cualquier modelo parametrizable, adaptándola a los g grados de libertad propios de cada modelo. Sea M el modelo determinado por los valores de cada uno de los g parámetros en la posición $(p_1, \ldots, p_i, \ldots, p_g)$ de la hipermatriz H resultante de la discretización del espacio g-dimensional. En esa posición, se almacenan las correspondencias que soportan M. Luego se toma el mayor conjunto de consenso de H y se estima con sus puntos el modelo.

Este acercamiento presenta una dificultad notoria: la discretización del espacio. Elegir una forma para dividir el espacio no es un problema trivial. Cerca de los límites de cada región es donde se producen anomalías. Imaginemos, por ejemplo, que la gran mayoría de los puntos del ejemplo anterior estén sobre la recta formada por $(\theta_i = \frac{\pi}{4}, r_j = 0)$. Las correspondencias, ya lo dijimos, tienen errores. Si partimos groseramente el espacio de θ de la siguiente forma: cada valor discreto es $c\frac{\pi}{2}$ con $0 \leq c \leq 3$. Para $\mu \approx 0 \wedge \mu > 0$, las correspondencias que están en una recta $(\frac{\pi}{4} + \mu, 0)$ podrían caer en $\frac{\pi}{2}$ y las correspondencias que están en $(\frac{\pi}{4} - \mu, 0)$ podrían caer en 0. Esto produce claramente una partición defectuosa de las correspondencias.

El problema reside realmente en que el espacio debe ser discretizado a priori, es decir que la discretización no depende realmente del conjunto de correspondencias S que tenemos. En cambio, el siguiente método tiene un comportamiento adaptativo al conjunto S.

6.2. RANSAC

Fischler y Bolles, en [5], introdujeron un nuevo método para ajustar un modelo a un conjunto experimental de datos que contenga, potencialmente, un porcentaje significativo de errores groseros. El procedimiento RANSAC, es el opuesto al de las técnicas convencionales: en lugar de utilizar tantos datos como es posible para obtener una solución inicial y luego eliminar los datos inválidos, se parte de un conjunto inicial mínimo agrandándolo con datos consistentes.

Inicialmente, se toma entre las correspondencias S un subconjunto minimal s. En este caso, minimal significa que el conjunto contiene la cantidad mínima de puntos necesaria para poder estimar un modelo determinado. Por ejemplo, para una transformación afín, son necesarias tres correspondencias (seis puntos) ya que ésta tiene seis grados de libertad. La elección de este subconjunto se realiza mediante un proceso aleatorio, aunque puede, alternativamente, utilizarse otro método que aproveche alguna información a priori que se tenga sobre el conjunto inicial de correspondencias.

El funcionamiento del método es iterativo. En la iteración i, se estima el modelo T_i a partir del subconjunto s. Luego, procedemos a verificar qué correspondencias de S son *inliers* al modelo T_i y con ellas formamos un subconjunto S_i. Reestimamos T_i, utilizando S_i en lugar de s (obviamente, $s \subseteq S$). Realizando este proceso iterativamente, obtenemos N subconjuntos S_i y N modelos T_i. Finalmente elegimos el modelo T_j tal que el conjunto *de consenso* S_j correspondiente sea el de máximo cardinal entre los N.

Más formalmente, el algoritmo 6.1 explica la forma en la que trabaja el paradigma.

Algoritmo 6.1 RANSAC: ajuste, en forma robusta, un conjunto de correspondencias S que contiene *outliers*

1: **for all** $0 \leq i < N$ **do**
2: elegir al azar una muestra s de correspondencias de S
3: instanciar el modelo T_i a partir de s
4: determinar el conjunto de consenso S_i de las correspondencias que soportan el modelo T_i
5: reestimar el modelo T_i usando S_i
6: **if** $\#S_i > u$ **then**
7: **return** T_i
8: **end if**
9: **end for**
10: **return** T_j tal que $j = \arg\max_{0 \leq i \leq N}(\#S_i)$

El método contiene tres parámetros sin especificar (variables libres):

1. la tolerancia usada para determinar si un punto es compatible con el modelo.
2. el número de iteraciones
3. el umbral u, que es el número de puntos necesarios para considerar que el modelo correcto fue encontrado.

En las secciones siguientes se explica como computar valores razonables para estos parámetros.

Tolerancia para establecer la compatibilidad entre un dato y un modelo

El error asociado a un modelo puede describirse, en parte, como una función de los errores de los datos utilizados para instanciar el modelo. Este error, junto al de un dato particular, componen la desviación de un dato a un modelo.

Si los datos forman un modelo con una función simple, es posible establecer analíticamente cotas para el error. Sin embargo, este acercamiento es generalmente inutilizable; para estos casos es necesario utilizar cotas experimentales. Por ejemplo, se pueden generar desviaciones muestrales perturbando los datos, computando el modelo y midiendo los errores obtenidos. Luego, la tolerancia puede fijarse a una o dos desviaciones estándar del error medio medido.

Como dijimos anteriormente la desviación depende del error particular del dato que estamos analizando, por lo que la tolerancia utilizada debería depender también de este error. Sin embargo, las variaciones en la tolerancia varían muy sutilmente de dato en dato, comparándolas con el resto de los errores presentes; podemos entonces despreciarlas y utilizar una tolerancia global.

Cantidad de iteraciones

La cantidad de intentos necesarios para encontrar un subconjunto s de n puntos no aberrantes (ver línea 2 del algoritmo 6.1) puede estimarse según

el siguiente modelo. Sean w la probabilidad de que un punto esté dentro de la tolerancia al error del modelo y N la cantidad de intentos que queremos estimar. La probabilidad de que todos los puntos de s estén en el modelo es $p = w^n$ (observar que esto implica que puede haber repeticiones en los puntos de s). La probabilidad de encontrar un subconjunto en N intentos está dada por:

$$P(N) = (q)^{N-1}p \quad \text{con } q = 1 - p.$$

Luego,

$$\begin{aligned} E_P(N) &= \sum_{i=1}^{\infty} i(q)^{i-1}p \\ &= p + 2qp + 3q^2p + \cdots + iq^{i-1}p + \dots \\ &= p(1 + 2q + 3q^2 + \cdots + iq^{i-1} + \dots) \end{aligned} \tag{6.2}$$

$E_P(N)$ representa la cantidad esperada de intentos necesarios para obtener un subconjunto s.

Se sabe que, para una suma geométrica,

$$\frac{x}{1-x} = x + x^2 + x^3 + \cdots + x^i + \dots$$

y derivando ambos términos con respecto a x, obtenemos

$$\frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + \cdots + ix^{i-1} + \dots \tag{6.3}$$

Combinando 6.2 y 6.3,

$$E_P(N) = \frac{p}{(1-p)^2} = \frac{1}{p} = w^{-n} \tag{6.4}$$

En general queremos, con el fin de tener mayores certezas, realizar un exceso de una o dos desviaciones estándar suplementarias, además de los $E_p(N)$ intentos.

Tenemos que:

$$V_P(N) = \sqrt{E_P(N^2) - E(N)^2} \tag{6.5}$$

y

$$\begin{aligned} E_P(N^2) &= \sum_{i=0}^{\infty} bi^2 a^{i-1} \\ &= \sum_{i=0}^{\infty} bi(i-1)a^{i-1} + \sum_{i=0}^{\infty} bia^{i-1} \end{aligned} \tag{6.6}$$

Derivando nuevamente los dos términos de la identidad 6.3,

$$\frac{2x}{(1-x)^3} = \sum_{i=0}^{\infty} \left(i(i-1)x^{i-1}\right) \tag{6.7}$$

Finalmente,

$$E_P(N^2) = \frac{2-b}{b^2}$$

y

$$V_P(N) = \frac{\sqrt{1-w^n}}{w^n} \tag{6.8}$$

Para facilitar el análisis es útil notar que $V(k) \approx E(k)$. Por ejemplo si $w = 0{,}5$ y $n = 4$ entonces $E(k) = 16$ y $V(k) = 15,5$. Esto implica que el número de intentos total sería dos o tres veces (una desviación estándar o dos, respectivamente) el número esperado implicado por $E(k)$.

Tamaño mínimo del conjunto de consenso

El umbral u, una variable libre dentro del paradigma RANSAC, es usado para determinar si un subconjunto S_i de S tiene suficiente consenso para ser aceptado por el algoritmo. La elección de u debe responder a dos motivos: encontrar el modelo correcto para los datos, y tener suficientes puntos como para poder estimar el modelo.

No hay métodos analíticos para determinar un valor de u, por lo que una buena estimación es terminar el algoritmo si el tamaño de S_i es similar a la cantidad de *inliers* que se cree que hay en el S. En otras palabras,

$$u = \beta n$$

donde β es la cantidad estimada de *inliers* en S.

Recordar que u debe ser suficientemente grande para poder estimar el modelo buscado.

6.3. Sistemas a estimar

Para concluir el capítulo, hablaremos de las transformaciones que buscamos estimar en este trabajo.

Toda imagen real está formada por la proyección en un plano de una escena determinada. La familia de transformaciones que modela este tipo de proyecciones se denomina proyectiva. Por lo tanto, cuando un objeto se mueve dentro de una escena, podemos modelar su posición final como una transformación proyectiva de la original.

Estas transformaciones son complejas e involucran un gran número de parámetros. Por suerte, es posible hacer una estimación de éstas mediante una subfamilia llamada transformaciones afines.

Conceptualmente, y visualmente, las transformaciones proyectivas involucran puntos de fuga; de lo que se desprende que algunas líneas que son en realidad paralelas se verán oblícuas. Las transformaciones afines, o simplemente afinidades, responden al modelo de la perspectiva axonométrica, donde todas las líneas mantienen su paralelismo.

Podemos escribir algebraicamente una afinidad A como

$$A = R(\theta)R(-\phi)KR(\phi) \tag{6.9}$$

donde $R(\theta)$, $R(-\phi)$ y $R(\phi)$ son matrices de rotación y K es una matriz diagonal, todas de 2×2. Sumando las traslaciones, la ecuación que la transformación afín de un punto $p \in \mathbb{R}^2$ en otro $p' \in \mathbb{R}^2$ es

$$\begin{aligned} p' &= Ap + t \\ \begin{pmatrix} x' \\ y' \end{pmatrix} &= \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \end{aligned} \tag{6.10}$$

El sistema puede rescribrirse para obtener:

$$\begin{pmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \\ & & \cdots & & & \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ \vdots \end{pmatrix} \tag{6.11}$$

que no es otra cosa que un sistema lineal $Ay = b$. Para que el sistema pueda ser resuelto, se necesitan al menos seis ecuaciones o, lo que es lo mismo, tres puntos. Si tenemos más de tres puntos, tenemos un sistema sobredeterminado.

Para resolver un sistema $Ay = b$ podemos utilizar el método de mínimos cuadrados que encuentra la solución mediante la ecuación

$$y = \left(A^T A\right)^{-1} A^T b \tag{6.12}$$

Una vez obtenida la matriz característica de la afinidad A, podemos encontrar su descomposición (ecuación 6.9) mediante el método *Singular Value Decomposition* (SVD). En el apéndice A se incluyen algunos detalles al respecto. La descomposición SVD de una matriz M es

$$M = UDV^T \tag{6.13}$$

donde U y V son matrices ortogonales y D es una matriz diagonal. Esta ecuación puede ser rescrita como

$$M = UV^T VDV^T. \tag{6.14}$$

Usando las ecuaciones 6.9 y 6.14 y estableciendo

$$\begin{aligned} R(\theta) &= UV^T \\ R(-\phi) &= V \\ K &= D \\ R(\phi) &= V^T \end{aligned}$$

Con esto podemos estimar los parámetros de una transformación afín.

Un caso particular de las transformaciones afines son las semejanzas. Éstas se dan cuando $K = sI$ con $s \in \mathbb{R}$. Tenemos entonces

$$\begin{aligned} A &= R(\theta)R(-\phi)KR(\phi) \\ A &= sR(\theta)R(-\phi)IR(\phi) \\ A &= sR(\theta) \end{aligned}$$

Las semejanzas combinan una traslación, una rotación y un cambio de escala isotrópico (homogéneo en todas las direcciones).

Todos estos métodos pueden combinarse con las herramientas provenientes de la geometría proyectiva y de la geometría epipolar para estimar asimismo posiciones 3D de puntos. Estos puntos provienen de dos imágenes obtenidas por dos cámaras que generan un sistema epipolar.

7. RESULTADOS Y DISCUSIÓN

En esta sección mostraremos los resultados obtenidos en la implementación de los métodos desarrollados que fueron expuestos en los capítulos anteriores.

El espacio-escala de una imagen, naturalmente, se ve como un suavizado gaussiano isotrópico (ver fig. 7.1). Una ventaja secundaria de construir cada nivel del espacio-escala independientemente, utilizando la propiedad de semi-grupo, es la menor pérdida de los bordes de la imagen. Si realizamos convoluciones sucesivas sobre una imagen iremos perdiendo cada vez más pixels en el borde de la misma, debido al efecto del zero-padding. Eso no sucede en el acercamiento elegido: todos los niveles pierden la misma cantidad de pixels en el borde.

A continuación se proveen algunos ejemplos de la representación que la función dDG, definida en §3, brinda de la estructura de una imagen (fig. 7.3). Es interesante observar que la imagen correspondiente a los primeros niveles de la función dDG presenta mucha textura y se pierden detalles estructurales importantes de la imagen. Esto es perfectamente natural y esperable: si miramos un objeto desde muy cerca, seremos capaces de ver con gran precisión los detalles de la textura de la superficie, pero no podremos observar con claridad la totalidad del mismo.

Para mostrar un ejemplo de las diferencias entre la representación de las funciones DG y dDG de una imagen (ver fig. 7.2), vemos una comparación entre estas dos funciones en la fig. 7.4. La mejora en la representación es, sobre todo, observable en la zona que dice "Casbah". En las imágenes correspondientes a la función DG (las de la izquierda) vemos que la zona no es realmente clara, presentando artefactos negros. Ésto no sucede con las imágenes de la derecha, correspondientes a la función dDG. Por otro lado, vale la pena hacer hincapié sobre el texto "IMPORTED FROM INDIA". La única

imagen que permite leerlo es la segunda de la derecha. Ninguna de las que corresponden a la función DG permite extraer esta información.

Como habíamos aclarado en el capítulo 3, hay dos puntos fundamentales para encontrar puntos clave en un determinado nivel del espacio-escala. El tamaño de la ventana utilizada para buscar extremos locales y el umbral k en la función potencia de un extremo (ver §3.4 en la pág. 16). La tabla 7.1 ilustra la cantidad de marcas encontradas para diferentes tamaños de ventana y umbrales. El criterio utlizado para seleccionar los parámetros fue que, para una imagen de 640×480 queden aproximadamente mil puntos. Si tenemos muchos más, la etapa de apareamiento se vuelve excesivamente costosa.

Para poder decidir entre las posibles elecciones que conducen a tener mil puntos en una imagen de 640×480 pixels, es necesario mirar los resultados que produce cada combinación en las imágenes más pequeñas.

Las figuras 7.7, 7.8, 7.9 y 7.10 muestran las marcas obtenidas para la imagen B de la tabla 7.1 (ver fig. 7.5).

En 7.7, podemos observar que las marcas obtenidas pertenecen en la inmensa mayoría a los niveles inferiores del espacio-escala (puntos magenta y azules). Ésto no es útil porque buscamos marcas que estén distribuídas en todos los niveles de escala discretizados. Esto se debe a que el parámetro $k = 6$ es demasiado alto para la pérdida de intensidad que ocurre en el espacio-escala. Esto muestra también que la suposición de iso-intensidad, de la que hablamos anteriormente, no es válida. Para la misma combinación de ventana de 3×3 y $k = 6$, podemos observar en la tabla que la cantidad de marcas encontradas para la imagen A es muy baja, comparada con las otras elecciones. Esto conduce que el conjunto de marcas constituya una descripción pobre de la misma.

La figura 7.10, presenta una baja cantidad de marcas detectadas (518). Podemos inferir que la elección de una ventana de 15×15 para la búsqueda de extremos locales es demasiado grande para imágenes más pequeñas, que requieren un análisis más minucioso ya que el rango que va desde su escala interior a la exterior es menor.

Podemos también observar cómo gran parte de las marcas están ubicadas en zonas aparentemente "suaves". Esta aparente falla se debe a la diversidad de texturas realmente presentes en la imagen en los niveles inferiores de escala

y que escapan al ojo.

En las figuras 7.11 y 7.12 se presentan algunos ejemplos de matchings de imágenes. Además de los cambios de posición y/o rotaciones, podemos observar un cambio de luminosidad respecto de la figura 7.2. Esto no afecta el apareamiento entre una y otras.

En la figura 7.13 tenemos un ejemplo de matching entre dos imágenes de una habitación tomadas una a 10 cm a la izquierda de la otra. Esto es una aproximación de lo que puede ser el sistema de cámaras de un robot. El apareamiento entre una imagen y la otra introduce claramente la posibilidad de usar las técnicas presentadas para un sistema de visión en robótica.

Para mostrar un ejemplo de los matchings obtenidos a partir de una imagen (ver fig. 7.14), le agregamos un marco negro a la misma y estimamos la semejanza (que en este acaso es sólo una traslación) entre la imagen original y ésta. Las figuras 7.15.a y 7.15.b muestran los matchings obtenidos al estimar. Vemos como RANSAC encuentra un apareamiento correcto, mientras que la transformación de Hough no. Esto refuerza lo planteado en §6.1, respecto de la dificultad de discretizar el espacio de parámetros a estimar. El método RANSAC tiene una forma adaptativa de armar conjuntos de consenso con los puntos que arroja resultados superiores.

Fig. 7.1: Espacio-escala de una imagen. La imagen superior izquierda es la orginal, es decir el nivel cero. La imagen inferior izquierda corresponde al primer nivel, la superior derecha, al segundo y la inferior derecha, al tercero.

Fig. 7.2: Imagen de prueba

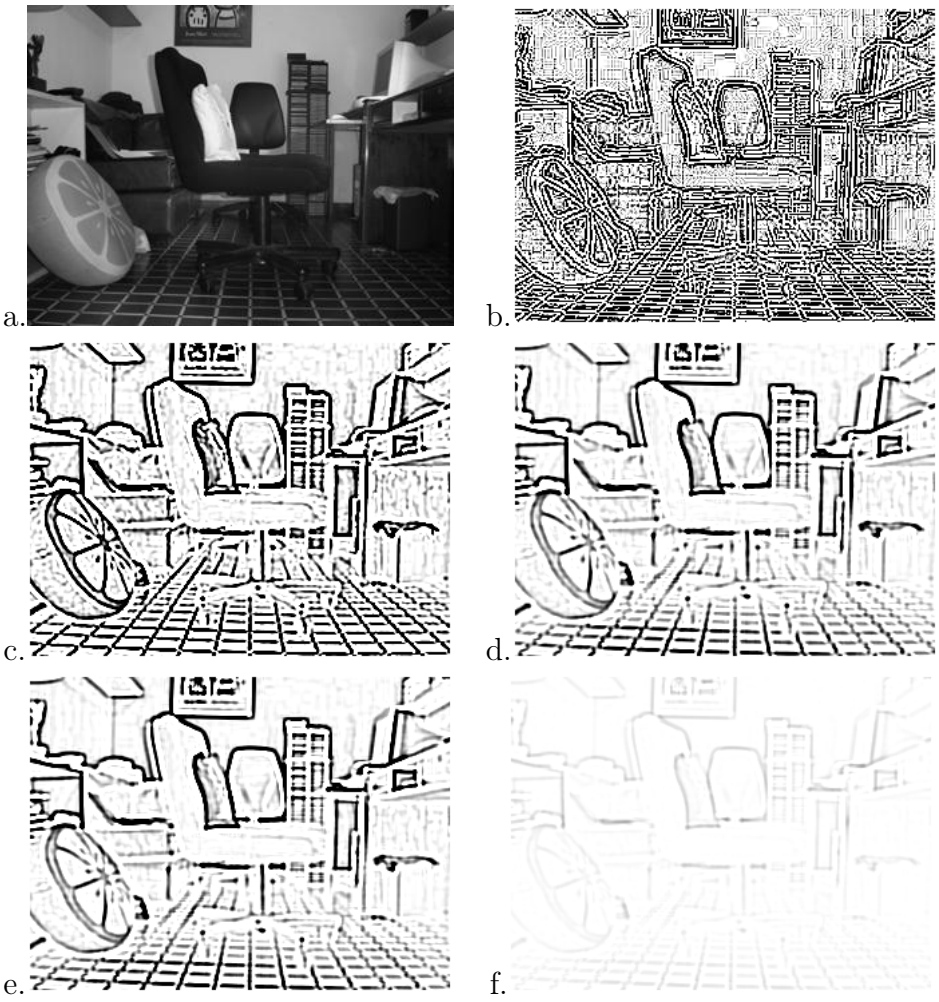

Fig. 7.3: Imagen original y representación de la función *dDG*. Las imágenes a,b,c,d,e,f constituyen la función *dDG* en niveles de escala creciente del espacio escala.

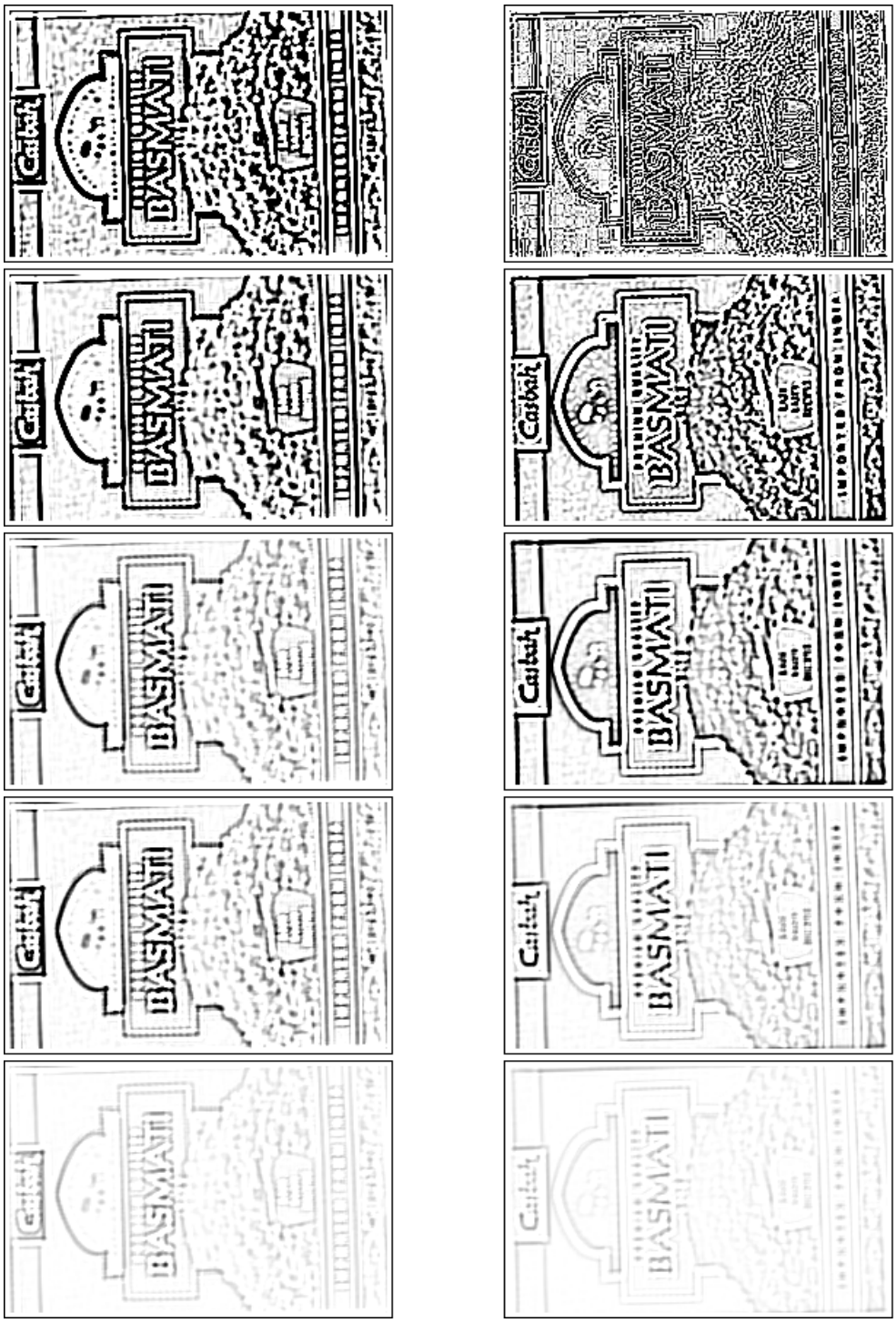

Fig. 7.4: Comparación entre la función DG (izquierda) y la función dDG (derecha). Los pares de imágenes horizontales corresponden a un mismo nivel del espacio-escala. De arriba hacia abajo se avanza en la escala.

Parámetros		**Imágenes** (tamaño en pixels)		
Ventana	**Umbral**	**A** (265 × 175)	**B** (512 × 384)	**C** (640 × 480)
3 × 3	4	125	1309	1613
3 × 3	6	81	**880**	**1071**
3 × 3	8	65	645	815
7 × 7	1	539	1909	2059
7 × 7	2	262	**980**	**1051**
7 × 7	3	167	648	701
11 × 11	0,5	384	985	1433
11 × 11	0,75	254	**640**	**976**
11 × 11	1	323	813	710
15 × 15	0,25	400	991	1835
15 × 15	0,5	203	**518**	**987**
15 × 15	0,75	136	358	660

Tab. 7.1: Cantidad de marcas en función del tamaño de la ventana y el valor del umbral para los extremos locales. La imagen A es la de la fig. 7.2, la B es la de la fig. 7.5 y la C es la de la fig. 7.6. Las combinaciones que producen aprox. mil puntos para imágenes de 640 × 480 son las que consideramos óptimas (ver números en negrita). Los números subrayados y en negrita corresponden a las figuras elegidas para completar el análisis (ver fig. 7.7, 7.8, 7.9, 7.10).

Fig. 7.5: Imagen de prueba correspondiente a la columna B en la tabla 7.1.

Fig. 7.6: Imagen de prueba correspondiente a la columna C en la tabla 7.1.

Fig. 7.7: Marcas de la imagen para una ventana de 3×3 y un umbral $k = 6$. Los puntos azules y magentas se corresponden con los niveles inferiores de escala.

Fig. 7.8: Marcas de la imagen para una ventana de 7×7 y un umbral $k = 2$. Los puntos azules y magentas se corresponden con los niveles inferiores de escala.

Fig. 7.9: Marcas de la imagen para una ventana de 11×11 y un umbral $k = 0,75$. Los puntos azules y magentas se corresponden con los niveles inferiores de escala.

Fig. 7.10: Marcas de la imagen para una ventana de 15×15 y un umbral $k = 0,5$. Los puntos azules y magentas se corresponden con los niveles inferiores de escala.

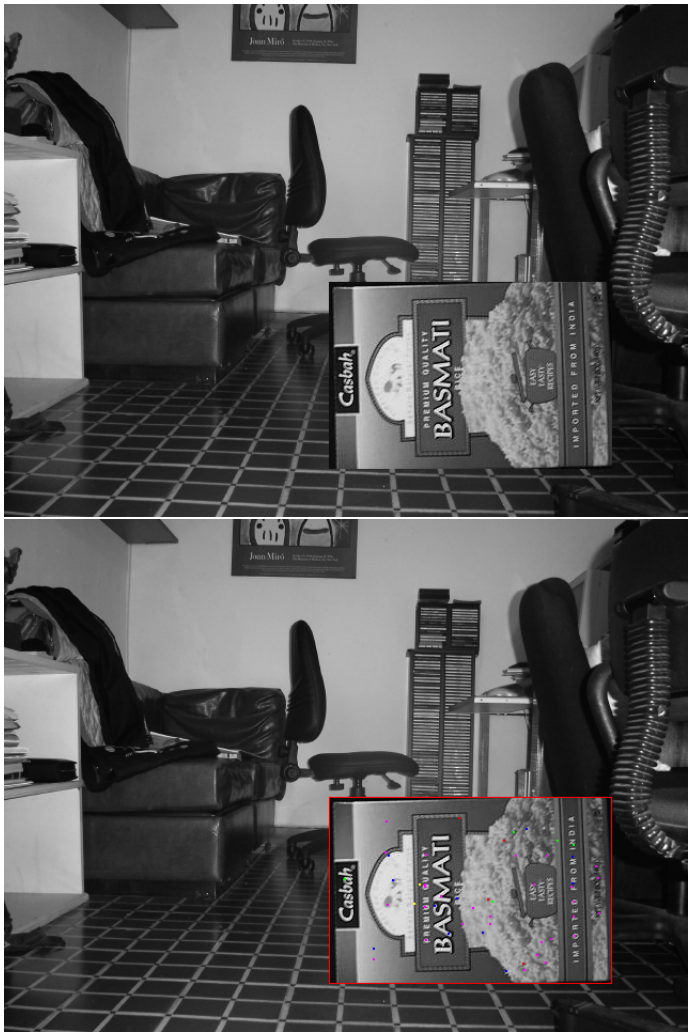

Fig. 7.11: Imagen original y matching encontrado de la fig. 7.2 en la misma.

Fig. 7.12: Imagen original y matching encontrado de la fig. 7.2 en la misma.

Fig. 7.13: Dos imágenes de la misma habitación tomadas una a 10 cm a la izquierda de la otra. En la tercera figura observamos el matching de la primera en la segunda.

Fig. 7.14: Imagen de prueba

a.

b.

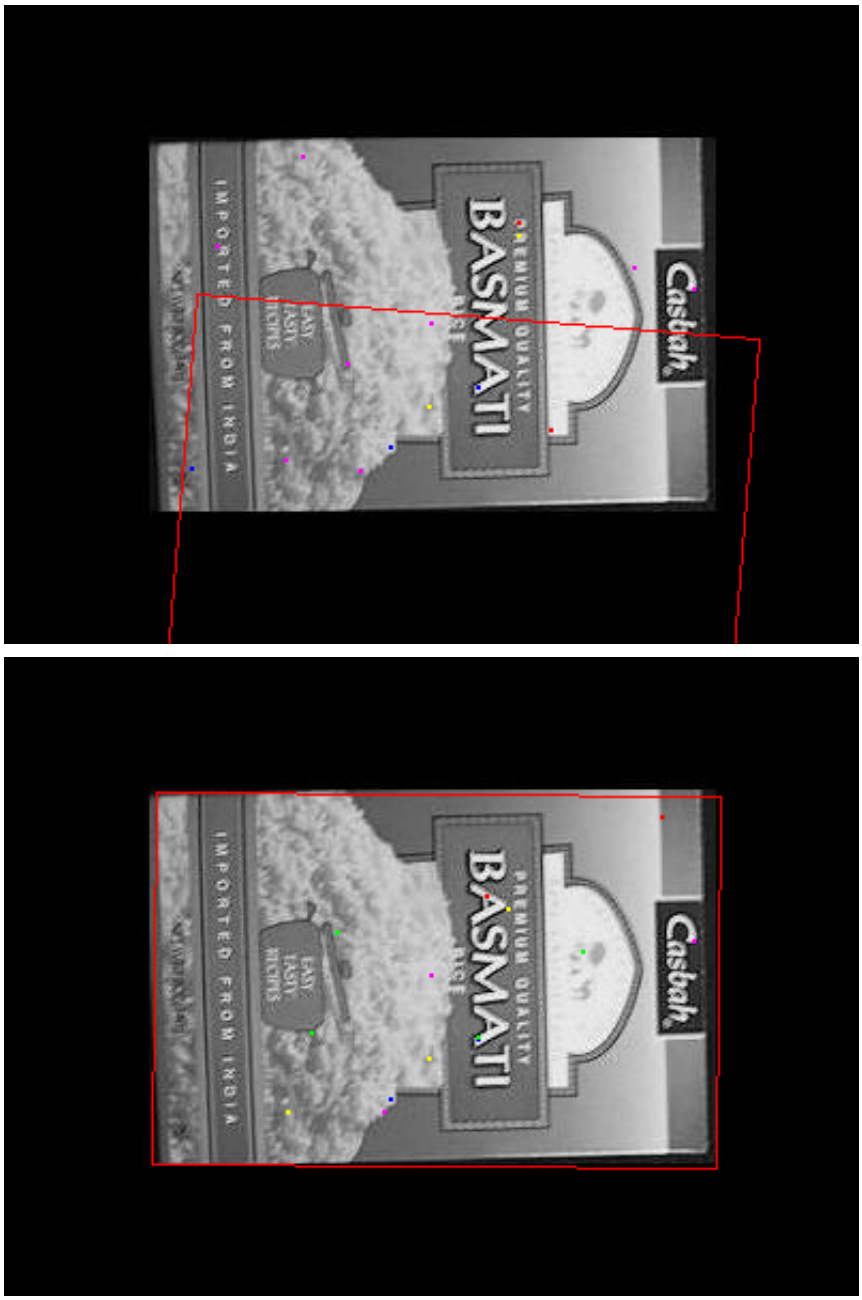

Fig. 7.15: Comparación de los resultados del apareamiento de una imagen (ver fig. 7.14) usando: (a) la transformación de Hough, y (b) RANSAC

8. CONCLUSIONES

En este trabajo hemos presentado un método para detectar marcas naturales en múltiples escalas. Estas marcas son puntos que consideramos claves en la estructura jerárquica de la imagen. Luego establecimos un método para decribir la imagen localmente en cada una de dichas marcas. Los descriptores utilizados están basados en el comportamiento de mecanismos naturales presentes en las neuronas del córtex primario.

El trabajo realizado en esta tesis no sólo presenta ventajas a nivel conceptual si no que las acompaña con ventajas prácticas. Está basado en la noción formal de espacio-escala para la detección de marcas y en mecanismos naturales de probada efectividad en el reino animal para la descripción de las mismas.

La combinación de estos métodos con los sistemas de almacenamiento de descriptores y con los estimadores robustos, dan pie para proveer aplicaciones prácticas. Entre ellas, la posibilidad de combinarlas con herramientas de geometría epipolar para utilizarlas en un sistema de localización y mapeo en robótica.

8.1. Problemas abiertos

El ruido presente en una imagen, intuitiva y prácticamente, genera un gran número de extremos locales. Éstos, si se encuentran cerca de un extremo más prominente son eliminados rápidamente y sólo se conservan en las primeras escalas. Sin embargo, si ocurren en una región con variaciones suaves, serán conservados en un rango mucho mayor de escalas. Pese a que su amplitud decrece rápidamente, no hay nada que nos permita dilucidar si puede establecerse un umbral con bases objetivas y "universalmente" válidas (es decir, que sean automáticas y no requieran ajustes manuales). Este

problema nos remite al de la estimación del nivel de ruido presente en una imagen, el cual es conocido como extremadamente difícil y es un tema actual de estudio en el campo del procesamiento digital de imágenes.

El problema de determinar la escala correcta a la que pertenece un punto es otro de los que suscita interés. Dado que la estructura en una imagen se establece jerárquicamente, la estabilidad de un punto en una escala determinada no es una propiedad de la imagen sino del objeto al que pertenece. En otras palabras la estabilidad de la escala es una propiedad local, lo que refuerza la idea del análisis local efectuado en este trabajo. Por otro lado, no existe evidencia de que el problema de la elección de la "mejor escala" para un punto tenga una única solución. Dado un punto, éste existe, o es significativo, en un cierto número de escalas y, por ende, la función de escala estable es una función multivaluada.

Los análisis basados en la vinculación de puntos a través de las escalas presuponen la iso-intensidad de la representación. Esto claramente no es verdadero ya que el nivel de gris de los puntos va uniformizándose con el suavizado en el espacio-escala. Sólo es posible realizar análisis cualitativos en base a características tales como bordes o extremos locales. Queda abierto el problema de encontrar un método que modele la variación de la intensidad en el espacio-escala y permita un análisis cuantitativo entre escalas.

8.2. Contribuciones

Los aportes de este trabajo consiten en los siguientes puntos:

1. en la detección de marcas naturales

    - trabajamos con la representación del espacio-escala, sin recurrir a pirámides, lo que brinda ventajas sustanciales en la ubicación espacial de las marcas y es invariante a traslación (§3).
    - elegimos trabajar con extremos locales en la tercera derivada de la función espacio-escala en lugar de sus ceros (extremos en la segunda derivada). Esto condujo a una representación más clara e intuitiva a nivel visual que las utilizadas anteriormente (§3).

- encontrar extremos en la tercera derivada permite incorporar al análisis información de escalas adyacentes sin sufrir los perjuicios que impone la suposición de la iso-intensidad cuando se comparan extremos entre escalas (§3).
- introducimos la posibilidad de realizar un análisis estadístico del criterio para el umbralado de los extremos locales sin pérdida significativa de rendimiento (§3).
- la complejidad algorítmica del análisis de cada candidato a marca natural es significativamente menor (§3).
- el algoritmo es totalmente paralelizable. Es posible ejecutar en paralelo el análisis de cada una de las escalas. (§3)

2. en la construcción de descriptores locales

   - propusimos un algoritmo para serializar el descriptor neurológico que robustece y completa en forma notable la invarianza frente a rotaciones que éste poseía (§4).

8.3. Trabajo a futuro

Como trabajo a futuro resta tratar la idea de utilizar ventanas adaptativas en cada escala. Se trata de agrandar la ventana sobre la que se construye el descriptor a medida que se avanza en la escala, con el fín de incorporar más información relevante (recordando que el suavizado va decrementando la información disponible), acompañando esto con un remuestreo local (restringido a la misma ventana). Creemos que la combinación de estas dos ideas tornará la descripción más precisa. Sería también interesante analizar posibles métodos heurísticos para elegir la escala correcta de una marca.

Por otro lado, el criterio para la selección de los extremos locales puede ser revisado, intentando utilizar un estadístico más certero que el usado actualmente.

Finalmente, la implementación en paralelo de los algoritmos para la detección de marcas naturales y para su descripción, sería un paso adelante significativo hacia la concreción de un sistema en tiempo real.

APÉNDICE

A. DESCOMPOSICIÓN EN VALORES SINGULARES

La descomposición en valores singulares (*Singular Value Decomposition*, SVD) es ampliamente usada en computación numérica, especialmente para resolver sistemas de ecuaciones sobredeterminados.

Dada una matriz cuadrada A, la SVD, es una factorización $A = UDV^T$ donde U y V son matrices ortogonales y D es una matriz diagonal con todos sus valores positivos. Como convención se escribe V^T en lugar de V y D tiene sus valores ordenados en forma descendente (es posible encontrar la SVD que cumpla esta restricción).

La SVD también existe para matrices no cuadradas de $m \times n$ y resulta particularmente útil cuando $m \leq n$. En este caso, A es factorizada como UDV^T donde U es una matriz de $m \times n$ con columnas ortogonales [1], D es una matriz diagonal de $n \times n$ y V^T es una matriz ortogonal de $n \times n$.

Valores singulares y autovalores Los valores en la diagonal de D son los valores singulares de A. Para ver la conexión entre autovalores y autovectores, examinemos las siguientes ecuaciones

$$\begin{aligned} A &= UDV^T \\ A^TA &= UDV^TUDV^T = VDU^TUDV^T = VD^2V^T \\ A^TA &= VD^2V^{-1} \quad \text{ya que } V \text{ es ortogonal} \end{aligned}$$

Esta es la ecuación que define los autovalores de A^TA, ubicados en la diagonal de D^2. Los autovectores son las columnas de V. En definitiva, los valores singulares de A son las raíces cuadradas de los autovalores de A^TA. (Notar que A^TA es simétrica semidefinida positiva, por lo que sus autovalores son reales positivos, los valores singulares son siempre reales y positivos.)

[1] una matriz ortogonal M es tal que $M^TM = I$. También se cumple como corolario que $\|Mx\| = \|x\|$ para cualquier vector x

BIBLIOGRAFíA

[1] R. Arnheim. *El pensamiento Visual*. Editorial Universitaria de Buenos Aires, Buenos Aires, Argentina, 1973.

[2] D. Ballard. Generalizing the hough transform to detect arbitrary shapes. In M. Fischler and O. Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*, pages 714–725. Kaufmann, Los Altos, CA., 1987.

[3] A. Baumberg. Reliable feature matching across widely separated views. *cvpr*, 01:1774, 2000.

[4] S. Edelman, N. Intrator, and T. Poggio. Complex cells and object recognition, 1997.

[5] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*, pages 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[6] J. Freidman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[7] J. Hadamard. *Lectures on Cauchy's Problem in Linear Differential Equations*. Yale University Press, New Haven, 1923.

[8] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[9] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Neurophysiology*, 160:106–154, 1962.

[10] T. Iijima. Pattern recognition (in japanese). *Corona-sha*, 1973.

[11] T. Iijima. Theory of pattern recognition (in japanese). *Series of Basic Information Technology*, 6, 1989.

[12] T. Iijima. Basic theory of pattern observation (in japanese). *Papers of the Technical Group on Automata and Automatic Control*, Dec. 1959.

[13] S. Ishikawa J. Weickert and A. Imiya. On the history of gaussian scale-space axiomatics. *Gaussian scale-space theory*, pages 45–59, 1997.

[14] J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[15] T. Lindeberg. *Discrete Scale-Space Theory and the Scale-Space Primal Sketch*. PhD thesis, Royal Inst. of Technology, Stockholm, Sweden, 1991.

[16] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *J. of Applied Statistics*, 21(2):224–270, 1994. (Supplement on Advances in Applied Statistics: Statistics and Images: 2).

[17] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[18] D. Lowe. Local feature view clustering for 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 682–688. Springer, 2001.

[19] D. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 60, pages 91–110. Springer, November 2004.

[20] D. Marr. *Vision*. W.H.Freeman & Co., 1982.

[21] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 128–142. Springer, 2002. Copenhagen.

[22] J. Weickert, S. Ishikawa, and A. Imiya. Scale-space has been discovered in japan. Technical Report 18, Department of Computer Science, University of Copenhagen, 1997. 162.

[23] A. Witkin. Scale-space filtering. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 1019–1022, Karlsruhe, Germany, 1983.