UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: PSO Binario con control de velocidad. Aplicación al reconocimiento de Rostros

Autores: Juan Andrés Maulini

Director: Prof. Lic. Laura C. Lanzarini

Carrera: Licenciatura en Sistemas

Resumen

El reconocimiento de rostros es una técnica biométrica muy utilizada en diversas áreas como seguridad y control de accesos, medicina forense y controles policiales. El procedimiento consiste en determinar si la imagen del rostro de una persona se corresponde o no con alguna de las imágenes existentes en una base de datos.
Este problema es difícil de resolver automáticamente debido a los cambios que distintos factores producen en la imagen, como por ejemplo la expresión facial, el envejecimiento e incluso la iluminación.

En esta tesina se presenta una variante original del método de Optimización mediante Cúmulos de Partículas o PSO que ofrece buenos resultados en la optimización de funciones.

Finalmente la variante del método de PSO propuesta es aplicada en la resolución del problema de reconocimiento de rostros donde se describen y detallan los resultados obtenidos y se comparan contra otros métodos alternativos.

Palabras Claves

Computación Evolutiva, Inteligencia de Cúmulo, Optimización mediante Cúmulo de Partículas, Reconocimiento de Rostros.

Trabajos Realizados

Se propuso una nueva variante de PSO Binario y se comparó su desempeño contra dos soluciones alternativas. Se realizaron 30 ejecuciones independientes de cada uno de los tres métodos aplicándolos en cada caso a la resolución de 4 funciones de prueba. Se utilizó un test estadístico para medir la significación de los resultados obtenidos.

Se utilizó el método propuesto para seleccionar los vectores SIFT más representativos de cada imagen.

Conclusiones

Se presentó un nueva variante del método de PSO Binario logrando una mejora en su funcionamiento mediante una adecuada modificación del vector velocidad.
Como caso de aplicación se utilizó el método propuesto para resolver el problema de reconocimiento de rostros logrando mejorar la tasa de acierto a través de la reducción de falsos positivos. En cuanto al tamaño de la BBDD, para los casos analizados, se logró una reducción entre el 25% y el 50%.

Trabajos Futuros

Resultaría interesante repetir el caso de aplicación utilizando imágenes color o en 3D.

Otro aspecto necesario a la hora de realizar un reconocimiento es considerar una clase de rechazo si se espera transferir esta propuesta a una situación real.

Fecha de la presentación: Octubre 2012

PSO Binario con control de velocidad
Aplicación al reconocimiento de Rostros

Autor: Juan Andrés Maulini

Directora: Prof. Lic. Laura C. Lanzarini

Tesina de Licenciatura en Sistemas

Facultad de Informática

UNIVERSIDAD NACIONAL DE LA PLATA

10 de octubre de 2012

Resumen

El reconocimiento de rostros es una técnica biométrica muy utilizada en diversas áreas como seguridad y control de accesos, medicina forense y controles policiales. Se trata de identificar si la imagen del rostro de una persona se corresponde o no con alguna de las imágenes existentes en una base de datos. Este problema es difícil de resolver automáticamente debido a los cambios que distintos factores, como la expresión facial, el envejecimiento e incluso la iluminación, producen en la imagen.

Esta tesina cubre varios aspectos de la solución de este problema:

- En primer lugar se presenta la Optimización mediante Cúmulos de Partículas o PSO (Particle Swarm Optimization), una metaheurística que ha sido utilizada exitosamente en la resolución de una amplia gama de problemas de optimización, incluyendo el entrenamiento de redes neuronales y la optimización de funciones. Para esta técnica se describe una variante original que ofrece buenos resultados.

- Luego se analizan algunos aspectos del reconocimiento de objetos en imágenes y se describe el método utilizado para establecer una representación del rostro a partir de un conjunto de vectores n-dimensionales.

- Finalmente se describen y detallan los resultados obtenidos de la aplicación de la variante de PSO propuesta en la resolución del problema de reconocimiento de rostros y se comparan los resultados obtenidos contra otros métodos alternativos.

Palabras Claves: Computación Evolutiva, Inteligencia de Cúmulo, Optimización mediante Cúmulo de Partículas, Reconocimiento de Rostros.

Índice general

Capítulo 1

Técnicas de Optimización

1.1. Introducción

La optimización es el concepto de encontrar la mejor solución, o al menos una solución lo adecuadamente buena. Debido a la gran importancia de los problemas de optimización en la Informática se han desarrollado diversos métodos para resolverlos denominados técnicas de optimización.
En este capítulo se realiza una introducción a las técnicas de optimización, realizando una clasificación y poder describir aquellas basadas en la Trayectoria y aquellas basadas en la Población.

1.2. Clasificación

Primeramente, las técnicas de optimización las podemos clasificar en exactas (o enumerativas, exhaustivas, etc.) y técnicas aproximadas. Las técnicas exactas garantizan encontrar la solución óptima para cualquier instancia de cualquier problema en un tiempo acotado. El inconveniente de estos métodos es que el tiempo necesario para llevarlos a cabo, aunque acotado, crece exponencialmente con el tamaño del problema, ya que la mayoría de éstos son NP-Completos. Esto provoca en muchos casos que el tiempo necesario para la resolución del problema sea inaceptable. Debido a estas limitaciones surgen los algoritmos aproximados o heurísticas. Estos métodos sacrifican la garantía de encontrar el óptimo a cambio de encontrar una buena solución en un tiempo razonable. En algunos casos, ni siquiera pueden determinar qué tan cerca del óptimo se encuentra una solución factible en particular.

Dentro de los algoritmos no exactos existen muchos métodos heurísticos de naturaleza muy diferente, por lo que es complicado dar una clasificación completa de los mismos.

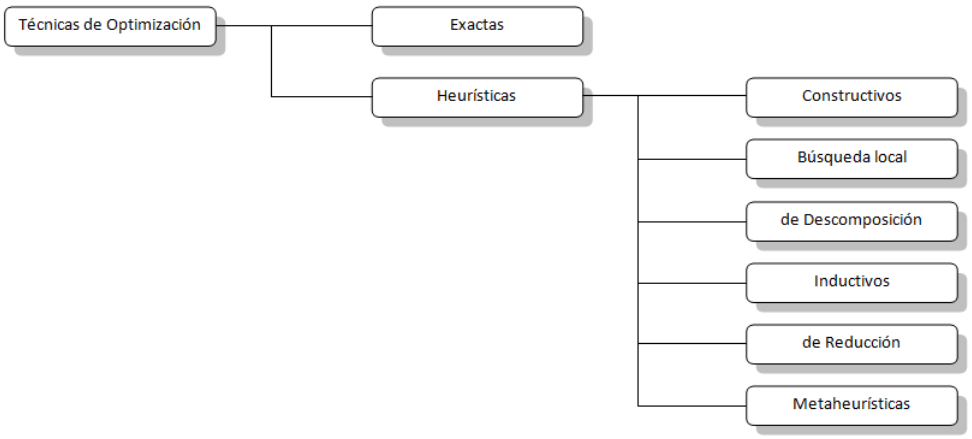

Figura 1.1: Clasificación de las técnicas de optimización.

Además, muchos de ellos han sido diseñados para un problema específico sin posibilidad de generalización o aplicación a otros problemas similares. De acuerdo a lo dicho anteriormente, como se observa en la figura 1.1, el siguiente esquema intenta realizar una categorización general para ubicar a los métodos heurísticos más conocidos según la forma de buscar y construir la solución [1]:

- Métodos constructivos: también llamados voraces. Los heurísticos constructivos suelen ser los métodos más rápidos. Consisten en construir literalmente paso a paso una solución del problema. Generan una solución partiendo de una vacía a la que se les va añadiendo componentes hasta tener una solución completa, que es el resultado del algoritmo. Usualmente son métodos deterministas y suelen estar basados en la mejor elección en cada iteración.

- Métodos de búsqueda local: también llamados de seguimiento del gradiente. Parten de una solución ya completa junto con el uso del concepto de vecindario, recorren parte del espacio de búsqueda hasta encontrar un óptimo local. En esa definición han surgido diferentes conceptos, como el de vecindario y óptimo local. El vecindario de una solución s, que notamos como N(s), es el conjunto de soluciones que se pueden construir a partir de s aplicando un operador específico de modificación (generalmente denominado movimiento). Un óptimo local es una solución mejor o igual que cualquier otra solución de su vecindario. Estos métodos, partiendo de una solución inicial, examinan su vecindario y eligen el mejor vecino continuando el proceso hasta que encuentran un óptimo local. El procedimiento realiza en cada paso un movimiento de una solución a otra con mejor valor. En muchos casos, la exploración completa del vecindario es inabordable y se siguen diversas estrategias, dando lugar a diferentes variaciones del esquema genérico. Según el operador de

movimiento elegido, el vecindario cambia y el modo de explorar el espacio de búsqueda también, pudiendo simplificarse o complicarse el proceso de búsqueda. El método finaliza cuando, para una solución, no existe ninguna solución accesible que la mejore.

- Métodos de descomposición: El problema original se descompone en subproblemas mas sencillos de resolver, siendo la salida de uno la entrada del siguiente.
- Métodos inductivos: La idea de estos métodos es generalizar de versiones pequeñas o más sencillas al caso completo. Propiedades o técnicas identificadas en estos casos más fáciles de analizar pueden ser aplicadas al problema completo.
- Métodos de reducción: Consiste en identificar propiedades que se cumplen mayoritariamente por las buenas soluciones e introducirlas como restricciones del problema. El objeto es restringir el espacio de soluciones simplificando el problema. El riesgo obvio es dejar fuera las soluciones óptimas del problema original.
- Metaheurísticas: En los años setenta surgió una nueva clase de algoritmos no exactos, cuya idea básica era combinar diferentes métodos heurísticos a un nivel más alto para conseguir una exploración del espacio de búsqueda de forma eficiente y efectiva. Estas técnicas se han denominado metaheurísticas, término utilizado por primera vez por Glover en [2]. Antes de que este término fuese aceptado completamente por la comunidad científica estos métodos eran denominados como heurísticos modernos [3].

Podemos describir las propiedades principales que caracterizan las metaheurísticas como [4] [5] [6]:

- Estrategias o plantillas generales que guían el proceso de búsqueda.
- Exploración del espacio de búsqueda eficiente con el objetivo de encontrar soluciones (casi) óptimas.
- Son algoritmos no exactos y generalmente son no deterministas.
- Pueden incorporar mecanismos para evitar las áreas del espacio de búsqueda no óptimas.
- El esquema básico de cualquier metaheurística es general y no depende del problema a resolver.
- Hacen uso de conocimiento del problema que se trata de resolver en forma de heurísticos específicos que son controlados de manera estructurada por una estrategia de más alto nivel.

- Utilizan funciones de aptitud para cuantificar el grado de adecuación de una determinada solución.

Haciendo un resumen de los puntos anteriores, se puede decir que una metaheurística es una estrategia de alto nivel que explora el espacio de búsqueda por medio de diferentes métodos. En otras palabras, una metaheurística es una plantilla general no determinista que debe ser rellenada con datos específicos del problema adaptados para la misma: representación de las soluciones, operadores para manipularlas, función de adecuación para el proceso de optimización, etc. Permite abordar problemas con espacios de búsqueda de gran tamaño, donde el número de posibles soluciones es extremadamente grande. Por lo tanto es de especial interés el correcto equilibrio que haya entre diversificación e intensificación.El termino diversificación se refiere a la exploración del espacio de búsqueda, mientras que intensificación se refiere a la explotación de algún área concreta de ese espacio. El equilibrio entre estos dos aspectos contrapuestos es de gran importancia, ya que por un lado deben identificarse rápidamente la regiones prometedoras del espacio de búsqueda global y por el otro lado no se debe malgastar tiempo en las regiones que ya han sido exploradas o que no contienen soluciones de alta calidad.

Existen diversas formas de clasificar y describir las técnicas metaheurísticas [7]. Dependiendo de las características que se seleccionen se pueden obtener diferentes taxonomías: basadas en la naturaleza o no basadas en la naturaleza, basadas en memoria o sin memoria, con función objetivo estática o dinámica, etc. En lo que continúa del capítulo hemos elegido clasificarlas de acuerdo a si en cada paso manipulan un único punto del espacio de búsqueda o trabajan sobre un conjunto (población) de ellos, es decir, detallaremos la agrupación que divide a las metaheuísticas en aquellas basadas en la trayectoria y aquellas basadas en la población, como se muestra en la Figura 1.2 en la que se pueden observar las principales metaheurísticas que pasamos a describir brevemente en las siguientes subsecciones.

1.2.1. Metaheurísticas Basadas en Trayectoria

La principal característica de estos métodos es que parten de un punto y mediante la exploración del vecindario van actualizando la solución actual, formando una trayectoria. La mayoría de estos algoritmos surgen como extensiones de los métodos de búsqueda local simples a los que se les añade alguna característica para escapar de los mínimos locales. Esto implica la necesidad de una condición de parada más compleja que la de encontrar un mínimo local. Normalmente se termina la búsqueda cuando se alcanza un número máximo predefinido de iteraciones, se encuentra una solución con una calidad aceptable, o se detecta un estancamiento del proceso.

- El Enfriamiento Simulado o Simulated Annealing (SA) es una de las más antiguas

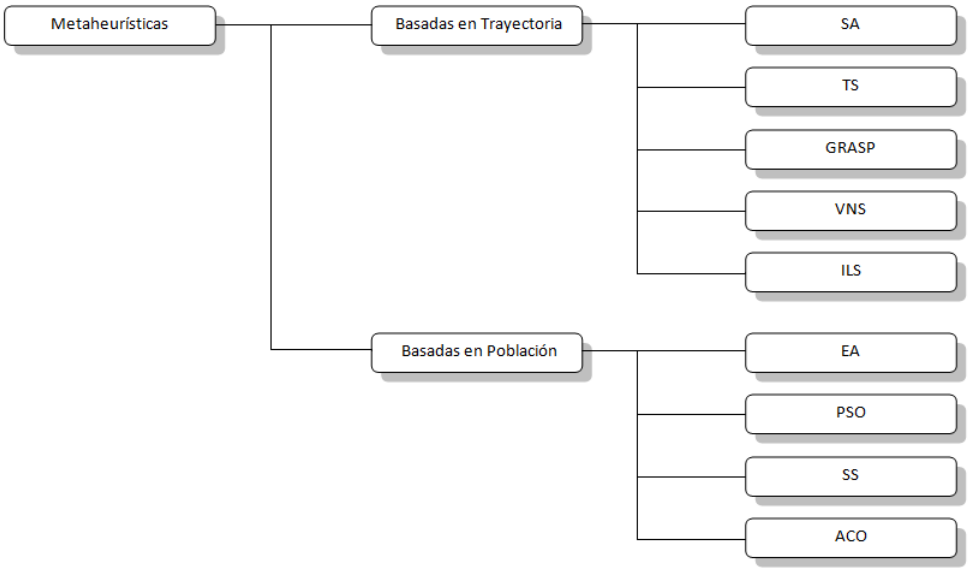

Figura 1.2: Clasificación de las metaheurísticas.

entre las metaheurísticas y seguramente es el primer algoritmo con una estrategia explícita para escapar de los óptimos locales. El SA fue inicialmente presentado en [8]. La idea del SA es simular el proceso de recocido del metal y del cristal. Para evitar quedar atrapado en un óptimo local, el algoritmo permite elegir una solución peor que la solución actual. En cada iteración se elige, a partir de la solución actual s, una solución s' del vecindario N(s). Si s' es mejor que s (es decir, tiene un mejor valor en la función de fitness), se sustituye s por s' como solución actual. Si la solución s' es peor, entonces es aceptada con una determinada probabilidad que depende de la temperatura actual T y de la variación en la función de fitness, f(s')-f(s) (caso de minimización). Esta probabilidad generalmente se calcula siguiendo la distribución de Boltzmann:

$$p(s'|T, s) = e^{\frac{f(s')-f(s)}{T}}$$

- La Búsqueda Tabú o Tabu Search (TS) es una de las metaheurísticas que se han aplicado con más éxito a la hora de resolver problemas de optimización combinatoria. Los fundamentos de este método fueron introducidos en [2]. La idea básica de la búsqueda tabú es el uso explícito de un historial de la búsqueda (una memoria de corto plazo), tanto para escapar de los óptimos locales como para implementar su estrategia de exploración y evitar buscar varias veces en la misma región. Esta memoria de corto plazo es implementada en esta técnica como una lista tabú, donde se mantienen las soluciones visitadas más recientemente para excluirlas

de los próximos movimientos. En cada iteración se elige la mejor solución entre las permitidas y la solución es añadida a la lista tabú.

- El Procedimiento de Búsqueda Miope Aleatorizado y Adaptativo o The Greedy Randomized Adaptive Search Procedure (GRASP) [9] es una metaheurística simple que combina heurísticos constructivos con búsqueda local. GRASP es un procedimiento iterativo compuesto de dos fases: primero una construcción de una solución y después un proceso de mejora. La solución mejorada es el resultado del proceso de búsqueda.

- La Búsqueda en Vecindario Variable o Variable Neighborhood Search (VNS) es una metaheurística propuesta en [10], que aplica explícitamente una estrategia para cambiar entre diferentes estructuras de vecindario de entre un conjunto de ellas definidas al inicio del algoritmo. Este algoritmo es muy general y con muchos grados de libertad a la hora de diseñar variaciones e instanciaciones particulares.

- La Búsqueda Local Iterada o Iterated Local Search (ILS) [11] es una metaheurística basada en un concepto simple pero muy efectivo. En cada iteración, la solución actual es perturbada y a esta nueva solución se le aplica un método de búsqueda local para mejorarla. Este nuevo óptimo local obtenido por el método de mejora puede ser aceptado como nueva solución actual si pasa un test de aceptación.

1.2.2. Metaheurísticas Basadas en Población

Los métodos basados en población se caracterizan por trabajar con un conjunto de soluciones (población) en cada iteración, a diferencia de los métodos observados anteriormente que únicamente utilizan un punto del espacio de búsqueda por iteración. El resultado final proporcionado por este tipo de algoritmos depende fuertemente de la forma en que manipula la población.

- Los Algoritmos Evolutivos o Evolutionary Algorithms (EA) [12] están inspirados en la capacidad de la naturaleza para evolucionar seres adaptándolos a los cambios de su entorno. Esta familia de técnicas siguen un proceso iterativo y estocástico que opera sobre una población de individuos. Cada individuo representa una solución potencial al problema que se está resolviendo. Inicialmente, la población es generada aleatoriamente (quizás con ayuda de un heurístico de construcción). Cada individuo en la población tiene asignado, por medio de una función de aptitud (fitness), una medida de su bondad con respecto al problema bajo consideración. Este valor es la información cuantitativa que el algoritmo usa para guiar su búsqueda. En los métodos que siguen el esquema de los algoritmos evolutivos, la modificación de la población se lleva a cabo

mediante tres operadores: selección, recombinación y mutación. Estos algoritmos establecen un equilibrio entre la explotación de buenas soluciones (fase de selección) y la exploración de nuevas zonas del espacio de búsqueda (fase de reproducción), basados sobre el hecho de que la política de reemplazo permite la aceptación de nuevas soluciones que no mejoran necesariamente la existentes. En la literatura se han propuesto diferentes algoritmos basados en este esquema general. Básicamente, estas propuestas se pueden clasificar en tres categorías que fueron desarrolladas de forma independiente. Estas categorías son la Programación Evolutiva o Evolutionary Programming (EP) desarrollada por Fogel [13], las Estrategias Evolutivas o Evolution Strategies (ES) propuestas por Rechenberg en [14], y los Algoritmos Genéticos o Genetic Algorithms (GA) introducidos por Holland en [15].

- La Búsqueda Dispersa o Scatter Search (SS) [6] es una metaheurística cuyos principios fueron presentados en [16] y que actualmente está recibiendo una gran atención por parte de la comunidad científica. El algoritmo se basa en mantener un conjunto relativamente pequeño de soluciones tentativas (llamado conjunto de referencia) que se caracteriza tanto por contener buenas soluciones como soluciones diversas. Este conjunto se divide en subconjuntos de soluciones a las cuales se les aplica una operación de recombinación y mejora. Para realizar la mejora o refinamiento de soluciones se suelen utilizar mecanismos de búsqueda local.

- Los sistemas basados en Colonias de Hormigas o Ant Colony Optimization (ACO) [17] son unas metaheurísticas inspiradas en el comportamiento de las hormigas reales cuando realizan la búsqueda de comida. Este comportamiento es el siguiente: inicialmente, las hormigas exploran el área cercana a su nido de forma aleatoria. Tan pronto como una hormiga encuentra la comida, la lleva al nido. Mientras que realiza este camino, la hormiga va depositando una sustancia química denominada feromona. Esta sustancia ayudará al resto de las hormigas a encontrar la comida. Esta comunicación indirecta entre las hormigas mediante el rastro de feromona las capacita para encontrar el camino más corto entre el nido y la comida. Esta funcionalidad es la que intenta simular este método para resolver problemas de optimización. En esta técnica, el rastro de feromona es simulado mediante un modelo probabilístico.

- Los Algoritmos Basados en Cúmulos de Partículas o Particle Swarm Optimization (PSO) [18] son técnicas metaheurísticas inspiradas en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces. Se fundamenta en los factores que influyen en la toma de decisión de un agente que forma parte de un conjunto de agentes similares. La toma de decisión por parte de cada agente se realiza conforme a una componente social y una componente individual, mediante las que se determina el movimiento (dirección) de este agente

para alcanzar una nueva posición en el espacio de soluciones. Simulando este modelo de comportamiento se obtiene un método para resolver problemas de optimización.

Capítulo 2

PSO - Particle Swarm Optimization

2.1. Introducción

La optimización mediante el cúmulo de partículas o PSO (Particle Swarm Optimization) es una metaheurística poblacional propuesta por Kennedy y Eberhart [18]. Está basada en la simulación de modelos sociales simples e inspirada en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces (Figura 2.1).

PSO utiliza lo que se conoce como *Inteligencia de Cúmulo* y plantea una "metáfora social" que se puede resumir de la siguiente forma: los individuos que conviven en una sociedad tienen una opinión que es parte de un "conjunto de creencias" (el espacio de búsqueda) compartido por todos los posibles individuos.

El libro de Kennedy y Eberhart [19] describe muchos aspectos filosóficos de PSO y la inteligencia de cúmulo. Poli hizo una cobertura exhaustiva de las aplicaciones de PSO en [20] y [21].

Cada individuo puede modificar su propia opinión basándose en tres factores:

- Su conocimiento sobre el entorno (su valor de aptitud o fitness).

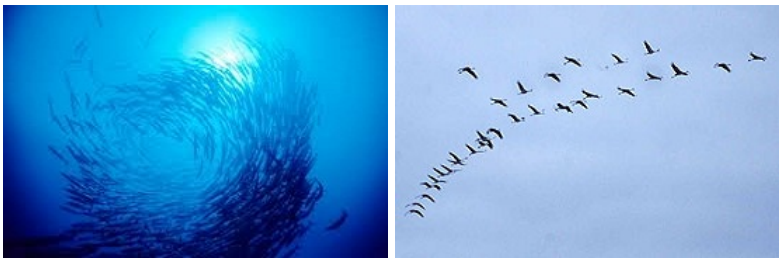

Figura 2.1: Ejemplos de *Inteligencia de Cúmulo* en la naturaleza

- Su conocimiento histórico o experiencias anteriores (su memoria o conocimiento cognitivo).
- El conocimiento histórico o experiencias anteriores de los individuos situados en su vecindario (su conocimiento social).

es decir que intervienen aspectos sociales, culturales y de imitación.

Los seres humanos también utilizan la *inteligencia de cúmulo* en la resolución de problemas de optimización. Por ejemplo, en un incendio, ante la imposibilidad de encontrar una vía de escape rápida, los humanos tienden a seguir el comportamiento de la mayoría; por tal motivo, si alguien advierte que todos corren en determinada dirección, lo natural es que adopte el mismo comportamiento sin saber efectivamente si esa dirección es la correcta. El conocimiento actual de cada individuo se ve afectado por el entorno.

Según se define en [18], PSO utiliza una población de tamaño fijo. Cada elemento de la población, denominado partícula, es una solución del problema y sus movimientos se encuentran acotados al espacio de búsqueda. Este espacio se encuentra definido de antemano y no se permite que las partículas se desplacen fuera de él.

En PSO, cada individo o partícula está siempre en continuo movimiento explorando el espacio de búsqueda y nunca muere.

Cada partícula esta compuesta por tres vectores y dos valores de fitness:

- Vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ almacena la posición actual de la partícula en el espacio de búsqueda.
- Vector $pBest_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ almacena la mejor solución encontrada por la partícula hasta el momento.
- Vector velocidad $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ almacena el gradiente (dirección) según el cual se moverá la partícula.
- El valor fitness $fitness\_x_i$ almacena el valor de aptitud de la solución actual (vector X_i).
- El valor fitness $fitness\_pBest_i$ almacena el valor de aptitud de la mejor solución local encontrada hasta el momento (vector $pBest_i$)

En la figura 2.2 se detalla una representación gráfica del movimiento de una partícula en el espacio de soluciones. Las flechas de línea punteada representan la dirección de los vectores de velocidad actual. La flecha de línea completa representa la dirección que toma la partícula para moverse desde la posición $X_i(t)$ hasta la posición $X_i(t + 1)$. El cambio de dirección de esta flecha depende de la influencia de las demás direcciones que intervienen en el movimiento.

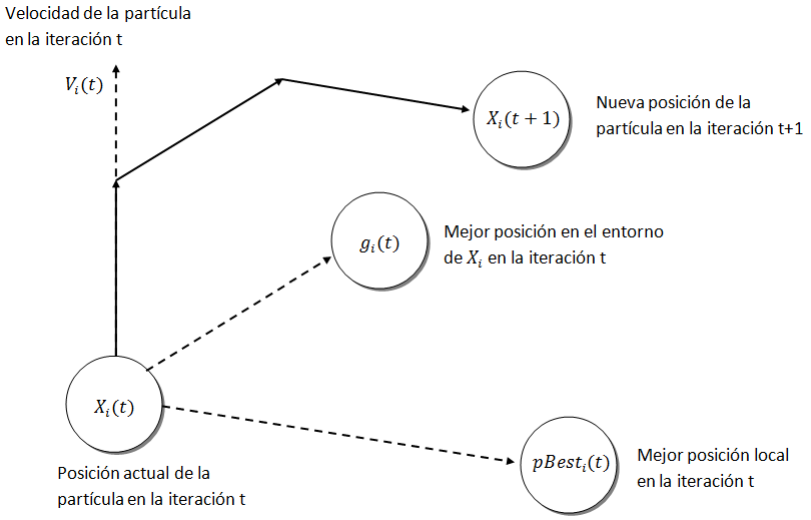

Figura 2.2: Movimiento de una partícula en el espacio de soluciones.

La población se inicializa generando las posiciones y las velocidades iniciales de las particulas aleatoriamente. Una vez que la población está inicializada, los individuos comienzan a moverse por el espacio de búsqueda por medio del proceso iterativo.

Con la nueva posición del individuo, se calcula y actualiza su fitness ($fitness\_x_i$). Además, si el nuevo fitness del individuo es el mejor encontrado hasta el momento, se actualizan los valores de mejor posición $pBest_i$ y fitness $fitness\_pBest_i$.

Como se explicó anteriormente, el vector velocidad es modificado tomando en cuenta su experiencia y su entorno.

La expresión es:

$$V_i(t+1) = w.V_i(t) + \varphi_1.rand_1.(pBest_i - X_i(t)) + \varphi_2.rand_2.(g_i - X_i(t)) \tag{2.1}$$

donde w representa el factor inercia [22], φ_1 y φ_2 las constantes de aceleración, $rand_1$ y $rand_2$ son valores aleatorios pertenecientes al intervalo (0,1) y g_i representa la posición de la particula con el mejor $pBest$ en el entorno de X_i ($lBest$ o $localbest$) o del todo el cúmulo ($gBest$ o $globalbest$). Los valores de w, φ_1 y φ_2 son importantes para asegurar que converja el algortimo. Para más detalles sobre la elección de estos valores consultar [23] y [24].

Finalmente, se actualiza la posición de la partícula de la siguiente forma:

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{2.2}$$

El algoritmo 1 describe el pseudocódigo del algoritmo de búsqueda de PSO

Algorithm 1: Algoritmo PSO Básico

```
Pop = CrearPoblacion(N);
while no se alcance la condición de terminación do
    for i = 1 to size(Pop) do
        Evaluar Partícula X_i del cúmulo Pop;
        if fitness(X_i) es mejor que fitness(pBest_i) then
            pBest_i ← X_i;
            fitness(pBest_i) ← fitness(X_i);
    for i = 1 to size(Pop) do
        Elegir g_i de acuerdo al criterio de vecindario usado;
        V_i ← w.V_i + (φ1.rand1.(pBest_i − X_i) + φ2.rand2.(g_i − X_i);
        X_i ← X_i + V_i;
```

Result: la mejor solución encontrada

Es importante notar que PSO es una metaheurística que hace muy pocas suposiciones con respecto al problema que se busca resolver y es capaz de buscar en espacios de soluciones muy grandes.

Como contrapartida, debe mencionarse que, metaheurísticas como PSO no garantizan que la solución óptima sea alcanzada. PSO no utiliza la información de ningún gradiente para realizar la búsqueda, esto implica que la función de aptitud a utilizar no necesita ser diferenciable como es requerido por los métodos de optimización clásicos, como descenso de gradiente y los métodos cuasi-Newton.

Esto último permite que PSO sea utilizado en problemas de optimización cuya función de aptitud sea ruidosa o se encuentre parcialmente definida.

Existen diferentes versiones que fueron desarrolladas a partir de la idea original, la mayoría de ellas relacionadas con la variación de diversos parámetros del algoritmo o a la combinación de varias topologías, tamaños y número de poblaciones [25][26][27][28].

También se han propuesto cambios en la velocidad de las partículas con el fin de lograr diversidad y evitar el estancamiento en el proceso de búsqueda [29] [30] [31].

2.2. Tipos de Algoritmos basados en PSO

El algoritmo PSO original puede alterarse modificando su configuración original y, de esta manera, dar origen a nuevas aproximaciones. A continuación se detallan variaciones clásicas basadas en alteraciones sobre el tamaño de la vecindad y otras basadas en alteraciones de los pesos del conocimiento cognitivo y social.

Dependiendo de la influencia de los pesos cognitivo y social (valores φ_1 y φ_2 respectivamente) sobre la dirección de la velocidad que toma una partícula en el movimiento (ecuación 2.1), Kennedy [32] se pueden distinguir cuatro tipo de algoritmos:

- Modelo Completo: $\varphi 1 > 0$ y $\varphi 2 > 0$. Tanto el componente cognitivo como el social intervienen en el movimiento.
- Modelo sólo Cognitivo: $\varphi 1 > 0$ y $\varphi 2 = 0$. Únicamente el componente cognitivo interviene en el movimiento.
- Modelo sólo Social: $\varphi 1 = 0$ y $\varphi 2 > 0$. Únicamente el componente social interviene en el movimiento.
- Modelo sólo Social exclusivo: $\varphi_1 = 0$, $\varphi_2 > 0$ y $g_i \neq x_i$. La posición de la partícula en sí no puede ser la mejor de su entorno.

Los parámetros $\varphi 1$ y $\varphi 2$ ponderan la importancia que se le otorga al conocimiento adquirido por la partícula o por el mejor del entorno respectivamente. A mayor valor de $\varphi 1$ o $\varphi 2$ mayor presión se realiza para que la partícula sea atraída por su mejor solución encontrada o por el mejor del entorno, respectivamente.

Por otro lado, desde el punto de vista del vecindario, el algoritmo puede ser clasificado en dos tipos de algoritmos: *PSO Local* y *PSO Global*.

En la variación *PSOLocal*, el conocimiento social de la partícula es evaluado sólo en el entorno inmediatamente próximo a la misma. En cambio, para la variación *PSOGlobal*, el conocimiento social para cada partícula es evaluado en la población completa.

El tamaño de la vecindad influye en la velocidad de convergencia del algoritmo así como en la diversidad de los individuos de la población. A mayor tamaño de vecindad, la convergencia del algoritmo es más rápida pero la diversidad de los individuos es menor.

2.3. Topologías de Cúmulos de Partículas

Un aspecto muy importante a considerar es la manera en la que una partícula interacciona con las demás partículas de su vecindario. Las topologías definen el entorno de interacción de una partícula individual con su vecindario, por lo que los entornos pueden ser de dos tipos [33]:

- Geográficos: se calcula la distancia de la partícula actual al resto y se toman las más cercanas para componer su entorno.
- Sociales: se define a priori una lista de vecinas para cada partícula, independientemente de su posición en el espacio.

En la figura 2.3 se muestran gráficamente ejemplos de entornos geográficos y sociales

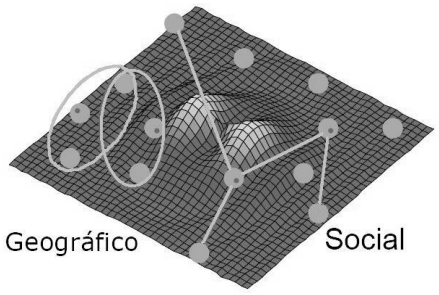

Figura 2.3: Ejemplos de entornos sociales y geográficos en un espacio de soluciones

2.4. Aspectos Avanzados de los Algoritmos basados en PSO

Pueden encontrarse en la literatura distintos variantes de PSO que buscan resolver sus problemas habituales generalmente relacionados con acotar la velocidad de la partícula o el tamaño de la población. En el primer caso, el vector velocidad es lo único que impide que la partícula se detenga. Lamentablemente, si toma un valor excesivamente fuera de rango, la partícula tiende a no estabilizarse generando oscilaciones. Por su parte, el tamaño de la población determina la capacidad exploratoria inicial de la población. Si es muy chica, es posible que no logre cubrir adecuadamente el espacio de entrada y si es muy grande, su evolución consumirá un tiempo computacional muy importante.

A continuación se idican algunas soluciones existentes referidas a estos aspectos.

2.4.1. Mecanismos de control de velocidad

Un problema de rendimiento de los algoritmos basados en PSO es que la magnitud de la velocidad suele llegar a ser muy grande durante la ejecución, provocando que las partículas se muevan demasiado rápido por el espacio de búsqueda si no se fija adecuadamente un valor de velocidad máximo. En [34] se comparan dos métodos para controlar el excesivo crecimiento de las velocidades: un factor de inercia ajustado dinámicamente y un coeficiente de constricción.

La idea de tener un factor de inercia w que influya en la velocidad dinámicamente indica que la inercia se reduce gradualmente a lo largo del tiempo (iteraciones del algoritmo) [35], como muestra la ecuación (2.3):

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}}.iter \tag{2.3}$$

donde w_{max} es el peso inicial y w_{min} el peso final, $iter_{max}$ es el número máximo de

iteraciones y *iter* es la iteración actual. w debe mantenerse entre 0.9 y 1.2. Valores altos provocan una búsqueda exhaustiva (más diversificación) y valores bajos una búsqueda más localizada (más intensificación).

Por otra parte, el coeficiente de constricción introduce una nueva ecuación para la actualización de la velocidad (ecuaciones 2.4 y 2.5). Este coeficiente asegura la convergencia [34].

$$v_i(t+1) = K[w.v_i(t) + \varphi_1.rand_1.(pBest_i - x_i(t)) + \varphi_2.rand_2.(g_i - x_i(t))] \tag{2.4}$$

$$K = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \tag{2.5}$$

2.4.2. PSO de población variable

Otro aspecto a tener en consideración es el tamaño del cúmulo de partículas, pues determina el equilibro entre la calidad de las soluciones obtenidas y el costo computacional.

En el algoritmo PSO descripto en [36], cada individuo permanece en continuo movimiento dentro del espacio de búsqueda y nunca muere. Este algoritmo necesita experiencia previa para la definición de la cantidad de soluciones que se van a manejar en cada iteración, de manera de no condicionar el resultado a obtener.

En [28] se propone una versión de PSO, denominada varPSO, que permite variar el tamaño de la población durante el proceso adaptativo con la intención de mejorar la relación de compromiso existente entre la velocidad de convergencia y la diversidad de la población. La variación de la población hace innecesario definir a priori el tamaño del cúmulo, evitando así el problema planteado anteriormente.

El algoritmo varPSO se basa en una modificación del proceso adaptativo que permite que las partículas se reproduzcan y mueran en función de su aptitud para resolver el problema planteado. Esto se realiza principalmente a través del concepto de edad que permite determinar el tiempo de permanencia de cada elemento dentro de la población. Además, dado que PSO tiende a poblar rápidamente las zonas exploradas con buen fitness, para no poblar excesivamente un mismo lugar del espacio de soluciones, se analiza el entorno de cada individuo y se eliminan las peores soluciones de las zonas muy pobladas.

Finalmente, la inserción de partículas tiene dos objetivos: incrementar la velocidad de convergencia incorporando individuos en las zonas menos pobladas y compensar la eliminación de partículas provocada por el cumplimiento de los respectivos tiempos de vida. Determinar los lugares convenientes, dentro del espacio de búsqueda, donde deben insertarse los nuevos individuos no es una tarea trivial. En realidad, se trata de una situación de compromiso entre la identificación de las zonas óptimas y la velocidad del proceso de inserción de los nuevos individuos.

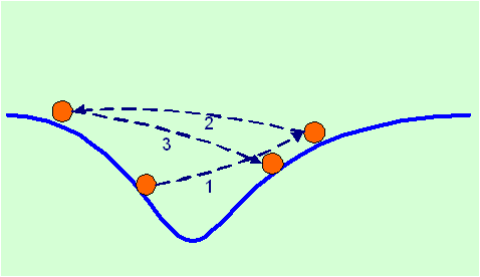

Figura 2.4: Detector de oscilación

Para más detalles consultar [28].

2.4.3. PSO con control de oscilaciones

PSO es una técnica de optimización que ha mostrado ser simple de implementar, estable, escalable y con la cual se han obtenido buenos resultados en optimización de funciones, incluso cuando se ha comparado con otras técnicas más utilizadas como los algoritmos genéticos. No obstante, esta técnica no está exenta de problemas, ya que tiene cierta tendencia a quedar atrapada en mínimos locales. La estrategia de exploración que utiliza alrededor de óptimos locales se muestra ineficiente, dado que tiende a oscilar alrededor de los mismos en repetidas ocasiones [37].

Teniendo en cuenta este último punto, en [38] se presenta una variante de PSO denominada oscPSO que cambia la estrategia de acercamiento a los óptimos locales, cuando se detecta una oscilación alrededor de ellos.

Para lograr este objetivo, se utiliza un procedimiento de búsqueda determinista que minimiza la cantidad de pasos para alcanzar los puntos óptimos.

Es importante considerar que, generalmente dichas oscilaciones se presentan sólo en algunas dimensiones de una misma partícula. Por lo tanto, en oscPSO, una misma partícula podrá utilizar dos estrategias distintas para modificar las dimensiones de su vector velocidad. Las dimensiones que se encuentren en estado de oscilación utilizarán un proceso de búsqueda local, mientras que las dimensiones que no se encuentren en estado de oscilación seguirán lo indicado en el algoritmo PSO convencional.

Una dimensión de una partícula será considerada en estado de oscilación si evidencia un comportamiento que alterna avances y retrocesos de forma sucesiva. Es decir que, para cada dimensión de una misma partícula, se analiza el signo de la velocidad resultante en cada iteración. Cuando se perciba que existe una secuencia de signos de velocidad contrarios en una sucesión de iteraciones, se define que la partícula, para esa dimensión en particular, se encuentra en estado de oscilación. La figura 2.4 ilustra esta situación.

El procedimiento de búsqueda local es utilizado para actualizar el vector velocidad de una

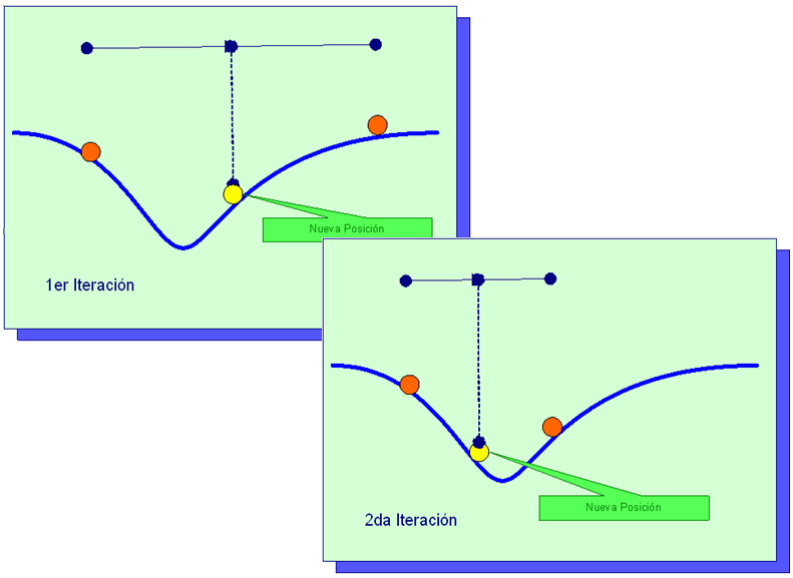

Figura 2.5: Búsqueda Local

partícula, en las dimensiones que se encuentran en estado de oscilación, reemplazando a lo indicado en (2.1) por un Procedimiento de Búsqueda Local Determinista. Este procedimiento busca las dos mejores posiciones con mayor fitness dentro de las ubicaciones memorizadas por esa partícula, con signos de velocidad opuestos en la dimensión a considerar, y calcula el punto medio entre ellas. Este procedimiento se repite en las sucesivas iteraciones, hasta encontrar una ubicación con fitness peor que el anterior. Requiere del almacenamiento de las últimas N posiciones de la partícula dentro del espacio de soluciones junto con sus valores de fitness correspondientes. La figura 2.5 muestra el funcionamiento del Procedimiento de Búsqueda Local.

Capítulo 3

PSO Binario

3.1. PSO Binario original

En diversas aplicaciones prácticas cada vez es más frecuente la presencia de problemas de optimización que involucran variables que deben tomar valores discretos. Se trata de problemas de naturaleza combinatoria donde una de las formas de resolverlos es la discretización de valores continuos. Considerando que cualquier problema de optimización, discreto o continuo, puede ser expresado en notación binaria, Kennedy y Eberthart en [36], consideraron de utilidad el poder disponer de una versión binaria discreta de PSO para problemas de optimización discretos que denominaron *PSO Binario*.

En un espacio binario, una partícula se mueve en las esquinas de un hipercubo, cambiando los valores de los bits que determinan su posición. En este contexto, la velocidad de una particula puede ser vista como la cantidad de cambios ocurridos por iteración o la distancia de Hamming entre las posiciones de la partículas en el instante t y el $t + 1$. Una partícula con 0 bits cambiados, de una iteración a la otra, no se mueve mientras que otra partícula que cambia por el valor contrario todos sus bits, se desplaza a velocidad máxima.

Aún no se ha dicho en que consiste el vector velocidad. Repitiendo la notación del capítulo anterior, la partícula de PSO binario está formada por

- Vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ almacena la posición actual de la partícula en el espacio de búsqueda. Es decir que se trata de un vector binario.
- Vector $pBest_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ almacena la mejor solución encontrada por la partícula hasta el momento. Por lo tanto, también es binario.
- Vector velocidad $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ es un vector de valores reales.
- El valor fitness $fitness\_x_i$ almacena el valor de aptitud de la solución actual (vector x_i).

- El valor fitness $fitness\_pBest_i$ almacena el valor de aptitud de la mejor solución local encontrada hasta el momento (vector $pBest_i$)

El valor de la velocidad para la partícula i en la dimensión d se actualiza según (3.1)

$$v_{id}(t+1) = w.v_{id}(t) + \varphi_1.rand_1.(p_{id} - x_{id}(t)) + \varphi_2.rand_2.(g_{id} - x_{id}(t)) \tag{3.1}$$

Nótese que si bien las ecuaciones (3.1) y (2.1) coinciden, ahora p_{id} y x_{id} son enteros en $\{0, 1\}$ y v_{id} es utilizada como argumento de la función sigmoide de (3.2) a fin de obtener un valor acotado en $[0, 1]$ equivalente a la probabilidad de que la posición de la partícula tome el valor 1.

$$sig(v_{id}(t)) = \frac{1}{1 + e^{-v_{id}(t)}} \tag{3.2}$$

Luego, el vector posición de la partícula se actualiza de la siguiente forma

$$x_{id}(t+1) = \begin{cases} 1 & if\ rand() < sig(v_{id}(t+1)) \\ 0 & if\ not \end{cases} \tag{3.3}$$

donde *rand*() es un número random con distribución uniforme en [0,1] distinto para cada dimensión.

El algoritmo 2 detalla el pseudocódigo correspondiente.

En la versión continua de PSO el valor de v_{id} también se encontraba acotado a un intervalo cuyos valores eran parámetros del algoritmo.

En el caso discreto, el valor de la velocidad es acotado según(3.4)

$$|v_{id}| < Vmax \tag{3.4}$$

siendo $Vmax$ un parámetro del algoritmo. Es importante notar que $Vmax$ debe tomar valores adecuados para permitir que las partículas sigan explorando mínimamente alrededor del óptimo. Por ejemplo, si $Vmax = 6$ las probabilidades de cambio $sig(v_{id})$ estarán limitadas a 0,9975 y 0,0025. Esto permite una reducida probabilidad de cambio. Pero si $Vmax$ tomara un valor extremo, por ejemplo 50, sería muy poco probable que una partícula cambie su posición luego de alcanzar un óptimo (que tal vez sea local).

Es importante remarcar que la incorporación de la función sigmoide cambia radicalmente la manera de utilizar el vector velocidad para actualizar la posición de la partícula. En PSO continuo, el vector velocidad toma valores mayores al inicio para facilitar la exploración del espacio de soluciones y al final se reduce para permitir que la partícula se estabilice.

En PSO binario ocurre precisamente todo lo contrario. Los valores extremos, al ser mapeados por la función sigmoide, producen valores de probabilidad similares, cercanos a 0 o a 1, reduciendo la chance de cambio en los valores de la partícula. Por otro lado,

Algorithm 2: Algoritmo PSO Binario

```
Pop = CrearPoblacion(N);
while no se alcance la condición de terminación do
    for i = 1 to size(Pop) do
        Evaluar Partícula X_i del cúmulo Pop;
        if fitness(X_i) es mejor que fitness(pBest_i) then
            pBest_i ← X_i;
            fitness(pBest_i) ← fitness(X_i);
    for i = 1 to size(Pop) do
        Elegir g_i de acuerdo al criterio de vecindario usado;
        V_i ← w.V_i + (φ1.rand1.(pBest_i − X_i) + φ2.rand2.(g_i − X_i);
        for d = 1 to n do
            if rand() < sig(V_id) then
                X_id = 1
            else
                X_id = 0
```

Result: la mejor solución encontrada

valores del vector velocidad cercanos a cero incrementan la probabilidad de cambio. Es importante considerar que si la velocidad de una partícula es el vector nulo, cada uno de los dígitos binarios que determina su posición tiene probabilidad 0.5 de cambiar a 1. Es la situación más aleatoria que puede ocurrir.

Cada partícula incrementa su capacidad exploratoria a medida que el vector velocidad reduce su valor; es decir que , cuando v_{id} tiende a cero, $\lim_{t\to\infty} sig(v_{id}(t)) = 0{,}5$, permitiendo que cada dígito binario tome el valor 1 con probabilidad 0,5. Es decir que puede tomar cualquiera de los dos valores.

Por el contrario, cuando los valores del vector velocidad se incrementan, $\lim_{t\to\infty} sig(v_{id}(t)) = 1$, y por lo tanto todos los bits tienden a 1, mientras que cuando el vector velocidad decrece, tomando valores negativos, $\lim_{t\to\infty} sig(v_{id}(t)) = 0$ y todos los bits cambian a 0. Es importante remarcar que limitando los valores del vector velocidad entre -3 y 3, $sig(v_{id}) \in [0{,}0474, 0{,}9526]$, mientras que para valores superiores a 5, $sig(v_{id}) \simeq 1$ y para valores inferior a -5, $sig(v_{ij}) \simeq 0$.

3.2. Variante de PSO Binario

Si bien se han definido distintas alternativas al algoritmo PSO Binario descripto en la sección anterior, como por ejemplo [39], [40] y [41], en esta tesina se ha analizado en detalle el artículo [42] por su buen desempeño y facilidad de implementación. Esta variante de PSO será utilizada para comparar los resultados obtenidos por el método propuesto en esta tesina.

En la versión de PSO Binario definida en [42], se cuestiona la dificultad de selección de parámetros del PSO Binario. En especial se menciona la importancia que el parámetro de inercia tiene en la capacidad exploratoria del método.

Por este motivo, en [42] la velocidad de una partícula ya no incide en la probabilidad de cambiar a 1, sino en la probabilidad de cambiar desde su estado previo a su valor complementario.

En esta nueva definición, la velocidad de partícula como también sus parámetros tienen el mismo rol que en la versiones de PSO descriptas previamente. La mejor posición que visitó la partícula $PBest_i$ y la mejor posición global $gBest$ son actualizadas como en la versión continua o binaria de PSO.

Sin embargo, por cada partícula se introducen dos vectores: V_i^0 es la probabilidad que los bits de la partícula cambien a cero y V_i^1 es la probabilidad que los bits de la partícula cambien a uno. Dado que en la ecuación de actualización de estas velocidades el término de inercia es utilizado, estas velocidades no son complementarias. La probabilidad de cambio en el j-esimo bit de la i-esima partícula es definido como sigue:

$$v_{ij}^c = \begin{cases} v_{ij}^1 & if\ x_{ij} = 1 \\ v_{ij}^0 & if\ x_{ij} = 0 \end{cases} \tag{3.5}$$

La manera en que estos vectores son actualizados es la siguiente: Si el j-ésimo bit en la mejor posición global es 0 (cero) o si el j-ésimo bit en la mejor solución hallada por la partícula es cero, la velocidad v_{ij}^0 es incrementada y la probabilidad de cambiar a uno decrece. Por el contrario, si el j-ésimo bit en la mejor posición global es 1 o si el j-ésimo bit en la mejor solución hallada por la partícula es 1, v_{ij}^1 es incrementada y v_{ij}^0 decrece.

Usando este concepto, se extraen las siguientes reglas:

$$if\ PBest_{ij} = 1\ then\ d_{ij,1}^1 = c_1 r_1\ and\ d_{ij,1}^0 = -c_1 r_1$$

$$if\ PBest_{ij} = 0\ then\ d_{ij,1}^0 = c_1 r_1\ and\ d_{ij,1}^1 = -c_1 r_1$$

$$if\ gBest_j = 1\ then\ d_{ij,2}^1 = c_2 r_2\ and\ d_{ij,2}^0 = -c_2 r_2$$

$$if\ gBest_j = 0\ then\ d_{ij,2}^0 = c_2 r_2\ and\ d_{ij,2}^1 = -c_2 r_2$$

donde d^1_{ij} y d^0_{ij} son dos valores temporarios, r_1 y r_2 son dos valores aleatorios con distribución uniforme en (0,1) que se actualizan en cada iteración. Las constantes c_1 y c_2 son parámetros del algoritmo y se definen a priori.

Luego, los vectores velocidad se actualizan según (3.6) y (3.7)

$$v^1_{ij} = wv^1_{ij} + d^1_{ij,1} + d^1_{ij,2} \tag{3.6}$$

$$v^0_{ij} = wv^0_{ij} + d^0_{ij,1} + d^0_{ij,2} \tag{3.7}$$

donde w es un término de inercia. Desde este enfoque la dirección de cambio, hacia 1 o hacia 0, para cada bit, es considerada por separado.

Luego de actualizar la velocidad de las partículas, se obtiene la velocidad de cambio, como se indicó en (3.5).

Finalmente, para obtener la nueva posición de la partícula, se utiliza la función sigmoide definida en (3.2) y se calcula la nueva posición siguiendo lo indicado en (3.8)

$$x_{ij}(t+1) = \begin{cases} \overline{x_{ij}}(t) & if\ r_{ij} < sig(v_{ij}(t+1)) \\ x_{ij}(t) & if\ not \end{cases} \tag{3.8}$$

donde $\overline{x_{ij}}$ es el complemento a 2 de x_{ij}. Es decir, si $x_{ij} = 0$ entonces $\overline{x_{ij}} = 1$ y si $x_{ij} = 1$ entonces $\overline{x_{ij}} = 0$. De la misma forma que en el PSO Binario convencional, r_{ij} es un valor aleatorio con distribución uniforme entre 0 y 1.

3.3. PSO Binario con control de velocidad - Método propuesto

Tanto el método PSO Binario original como la variante descripta en la sección anterior dejan en evidencia la importancia que tiene una adecuada modificación del vector velocidad. En el caso del PSO Binario original, el problema central se encuentra en el escaso control que se realiza a la hora de acotarlo. Si el vector velocidad toma valores excesivos, el aplicarle la función sigmoide hace que la probabilidad de cambio sea casi nula. Como forma de compensar este efecto, el método definido en [42] buscó descomponer el vector velocidad en dos partes para poder tener una opinión de cambio por el valor contrario al que actualmente tiene la partícula.

Ambos métodos trabajan sobre partículas cuyas posiciones actuales se expresan de manera binaria.

Esta tesina propone modificar el vector velocidad utilizando una versión modificada del algoritmo gBest PSO continuo.

Es decir que, bajo esta propuesta cada partícula tendrá dos vectores velocidad, V1 y V2. El primero se actualiza según (3.9).

$$V1_i(t+1) = w.V1_i(t) + \varphi_1.rand_1.(2 * pBesti - 1) + \varphi_2.rand_2.(2 * gBest - 1) \quad (3.9)$$

donde las variables $rand_1$, $rand_2$, φ_1 y φ_2 funcionan de la misma forma que en (3.1). Los valores p_i y g_i corresponden al i-ésimo dígito binario de los vectores $pBest_i$ y $gBest$, respectivamente.
La diferencia más importante entre (3.1) y (3.9) es que en la segunda, el desplazamiento del vector $V1$ en las direcciones correspondientes a la mejor solución encontrada por la partícula y al mejor global no dependen de la posición actual de dicha partícula.
Luego, cada elemento del vector velocidad $V1$ es controlado según (3.10)

$$v1_{ij}(t) = \begin{cases} \delta 1_j & if\ v1_{ij}(t) > \delta 1_j \\ -\delta 1_j & if\ v1_{ij}(t) \leq -\delta 1_j \\ v1_{ij}(t) & if\ not \end{cases} \quad (3.10)$$

donde

$$\delta 1_j = \frac{limit1_{upper_j} - limit1_{lower_j}}{2} \quad (3.11)$$

Es decir que, el vector velocidad $V1$ se calcula según (3.9) y se controla según (3.10). Su valor se utiliza para actualizar el valor del vector velocidad $V2$, como se indica en (3.12).

$$V2(t+1) = V2(t) + V1(t+1) \quad (3.12)$$

El vector $V2$ también se controla de manera similar al vector $V1$ cambiando $limit1_{upper_j}$ y $limit1_{lower_j}$ por $limit2_{upper_j}$ y $limit2_{lower_j}$ respectivamente. Esto dará lugar a $\delta 2_j$ que será utilizado como en (3.10) para acotar los valores de $V2$. Luego se le aplica la función sigmoide y se calcula la nueva posición de la partícula según (3.3).

El concepto de control de velocidad se basa en el método descripto en [38] donde se utiliza un control similar para evitar las oscilaciones finales de las partículas alrededor del óptimo.

3.4. Comparación de Performance

En esta sección se comparará la performance de la variante de PSO binario propuesta en esta tesina con el método propuesto por Kennedy y Eberhart en [36] y el PSO binario definido en [42], en la minimización de un conocido conjunto de funciones de prueba N dimensionales las cuales se detallan a continuación

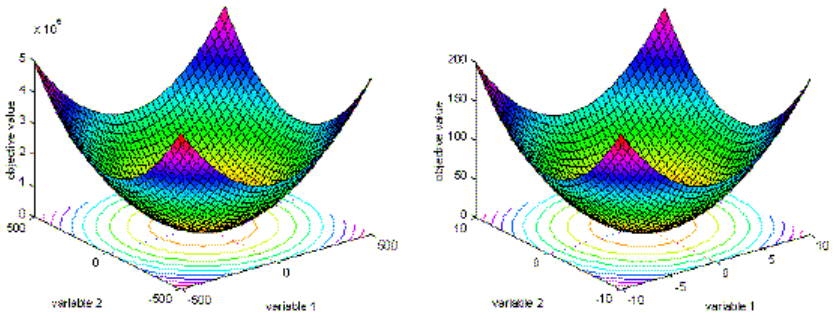

Figura 3.1: Función *Sphere* utilizando dos dominios distintos. Ambos gráficos se ven muy parecidos pese al cambio de escala. A izquierda se representa la función haciendo que cada variable tome valores entre -500 y 500 y a derecha se considera un área menor, entre -10 y 10.

3.4.1. Funciones utilizadas

Función Sphere o Función 1 de De Jong

La definición de la función es la siguiente:

$$f_1(X) = \sum_{i=1}^{n} x_i^2 \tag{3.13}$$

Esta es la función de prueba más sencilla. Es continua, convexa y unimodal. El mínimo global es 0 y se encuentra ubicado en $x(i) = 0$ para $i = 1 : n$

Función Rosenbrock o Función 2 de De Jong

En esta función el óptimo global está dentro de un largo y estrecho valle plano con forma parabólica. Si bien encontrar el valle es trivial, la convergencia hacia el óptimo global es difícil y por lo tanto, este problema se ha utilizado en varias ocasiones en evaluar el rendimiento de los algoritmos de optimización.
La definición de la función es

$$f_2(X) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i^2 \tag{3.14}$$

El mínimo global es 0 y se encuentra ubicado en $x(i) = 1$ para $i = 1 : n$

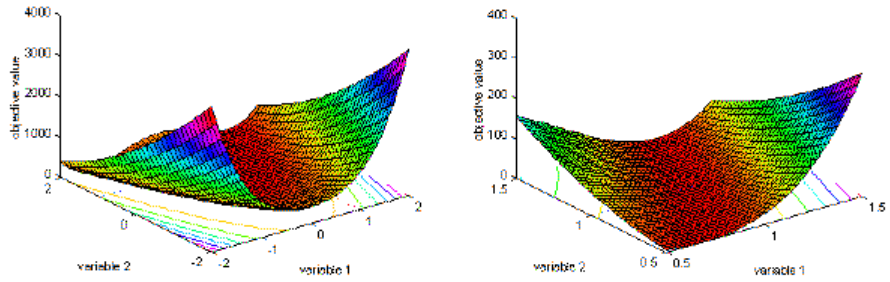

Figura 3.2: Función Rosenbrock. A izquierda se visualiza un área más grande que a la derecha donde se focaliza en el mínimo global

Función Griewangk

Se trata de una función con muchos óptimos locales y se define así

$$f_3(X) = \sum_{i}^{n} \frac{x(i)^2}{4000} - \Pi_{i=1}^{n}(cos(\frac{x(i)}{sqrt(i)}) + 1 \tag{3.15}$$

El mínimo global es 0 y se encuentra ubicado en $x(i) = 1$ para $i = 1 : n$

Función Rastrigin

La función se define de la siguiente forma

$$f_4(X) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i) \tag{3.16}$$

Esta función se basa en la función 1 y agrega la función coseno para producir muchos mínimos locales. Esto da lugar a una función multimodal. El mínimo global es 0 y se encuentra ubicado en $x(i) = 0$ para $i = 1 : n$

3.4.2. Pruebas realizadas y parámetros utilizados

Se realizaron 40 corridas independientes de cada uno de los métodos utilizando 2000 iteraciones máximas. Se trabajó con N=3, 5 y 10 variables. El tamaño de la población en todos los casos fue de 20 partículas. Los valores de *limite*1 y *limite*2 son iguales para todas las variables; estos son $[0; 1]$ y $[0; 6]$ respectivamente. Por lo tanto, los valores de los vectores velocidad $V1$ y $V2$ fueron limitados a los rangos $[-0{,}5, 0{,}5]$ y $[-3, 3]$ respectivamente. Es decir que pueden obtenerse probabilidades en el intervalo

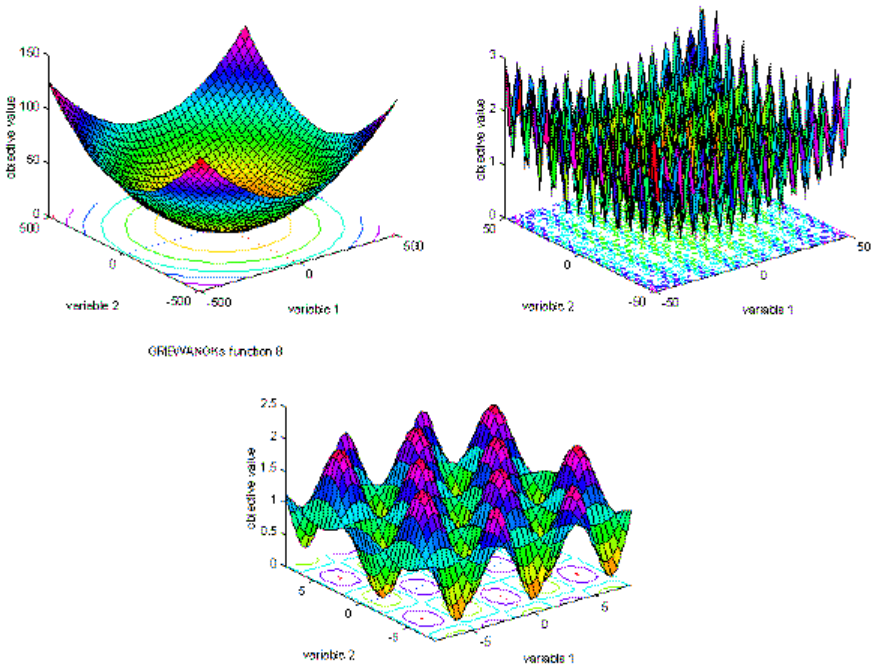

Figura 3.3: Función Griewangk utilizando tres dominios distintos a fin de mostrar las características que posee la función. Arriba a la izquierda se utiliza una área amplia para mostrar que la función es similar a la Función 1. Arriba a la derecha muestra que visto más de cerca posee muchos valles y picos. Sin embargo la figura inferior muestra que cerca del óptimo los valles y los picos son suaves.

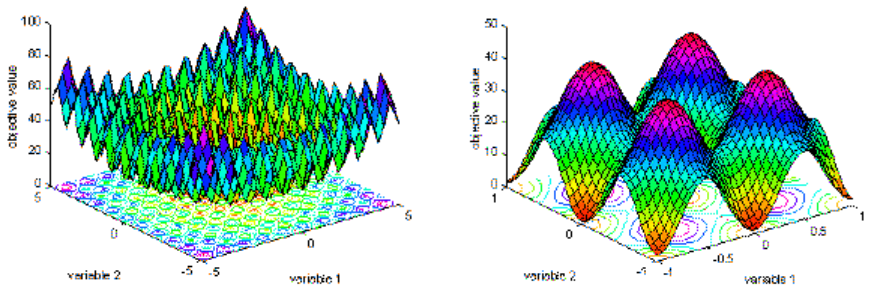

Figura 3.4: Función Rastrigin. A la izquierda sus variables toman valores entre -5 y 5 mientras que a la derecha lo hacen entre -1 y 1

Tabla 3.1: Resultados Obtenidos al utilizar el método propuesto y los métodos definidos en [36] y [42] para encontrar el mínimo de las funciones Sphere, Rosenbrock, Griewangk y Rastrigin numeradas en la tabla del 1 al 4 respectivamente

		Método propuesto		PSO Binario [36]		Variante de PSO Binario [42]	
Nro. Var	Nro. Func	Mejor Fitness	Mejor Fitness Promedio	Mejor Fitness	Mejor Fitness Promedio	Mejor Fitness	Mejor Fitness Promedio
3	1	0	**0**	0	1,2e-09	1,8e-08	6,3e-07
3	2	7,0e-04	**2,8**	1,1e-05	3,0	2,0e-03	5,6
3	3	2,1e-09	**3,3e-03**	2,1e-09	6,8e-03	4,2e-06	8,8e-03
3	4	5,4e-08	**5,4e-08**	5,4e-08	**5,4e-08**	8,9e-06	7,6e-04
5	1	0,0	**3,1e-09**	1,4e-07	1,3e-05	1,7e-04	1,2e-03
5	2	2,2	**28,7**	2,1	111,5	7,2	278,3
5	3	2,6e-09	**8,2e-03**	1,6e-03	2,0e-02	1,9e-02	6,6e-02
5	4	9,0e-08	**1,3e-07**	2,3e-04	5,1e-01	5,0e-01	3,7
10	1	8,2e-05	**9,8e-04**	1,3e-02	7,1e-02	9,7e-02	6,2e-01
10	2	7,3	**141,0**	334,0	2812,8	92013,0	613510,0
10	3	1,3e-02	**7,6e-02**	7,7e-02	1,9e-01	5,2e-01	7,3e-01
10	4	0,8	**4,3**	7,5	15,3	13,7	44,0
20	1	3,3e-01	**8,1e-01**	1,7e+00	3,9e+00	1,2e+01	1,9e+01
20	2	1865,9	**10105**	2,8e+05	1,2e+07	1,2e+08	5,0e+08
20	3	1,4e-01	**2,7e-01**	9,3e-01	1,0e+00	1,3e+00	1,4e+00
20	4	27,7	**43,0**	52,7	88,9	169,8	215,2

[0,0474, 0,9526]. Los valores para φ_1 y φ_2 fueron establecidos en 0,25. Con respecto a los métodos [36] y [42], se establecieron límites de velocidad entre $[-3, 3]$ a fin de mantener el mismo rango de probabilidades.

3.4.3. Resultados obtenidos

La tabla 3.1 muestra el fitness de la mejor solución encontrada por cada método, así como el valor del fitness promedio de las 40 ejecuciones. Las funciones de prueba utilizadas son las siguientes: Sphere, Rosenbrock, Griewangk y Rastrigin, las cuales aparecen numeradas del 1 al 4 respectivamente. Se recuerda que en todas las funciones se trata de un problema de minimización y por lo tanto se considera mejor a la solución que posea el menor valor de aptitud.

Tabla 3.2: Resultado del test ANOVA de un solo factor para decidir si existe una diferencia significativa entre los resultados promedio de los métodos utilizados para minimizar las funciones utilizando un nivel de significación de 0.05. La hipótesis nula sostiene que todas las medias son iguales. Para cada caso se indica el *p-valor* obtenido.

nro.Var	Función 1	Función 2	Función 3	Función 4
3	3,029718e-07	0,305974	0,000157	0,003812
5	0	0,000317	0	0
10	0	1,178835e-12	0	0
20	0	0	0	0

Puede observarse que el método propuesto encuentra las mejores soluciones y posee los menores valores de fitness promedio.
Para analizar si existen diferencias significativas entre los tres métodos analizados se realizó un test ANOVA de un solo factor conun nivle de significación de 0.05 para cada una de las dimensiones consideradas (N=3, 5, 10 y 10). La tabla 3.2 muestra el *p-valor* obtenido en cada caso. Nótese que salvo la función 2 en 3 variables, el resto posee un valor muy inferior al nivel de signifación indicando que se rechaza la hipótesis nula.

Para identificar la o las medias significativamente distintas se contruyeron los intervalos de confianza simultáneos para los tres métodos utilizando el mismo nivel de significación. Las figuras 3.5, 3.6, 3.7 y 3.8 grafican los resultados obtenidos.

Finalmente la tabla 3.3 resume las comparaciones de a pares de medias e indica si la diferencia es significativa o no. En cada caso la media corresponde al promedio de los 40 mejores fitness obtenidos por cada uno de los tres métodos. En dicha tabla se ha utilizado el símbolo ▲ para representar que el IC no contiene al 0, indicando que la hipótesis nula debe ser rechazada. El nivel de significación utilizado es 0.05. El símbolo ▽ indica que la hipótesis nula no se rechaza, es decir que las medias son iguales.

La figura 3.9 muestra los diagramas de caja calculados sobre los mejores resultados obtenidos en cada una de las 40 corridas. Cada columna corresponde a una función distinta. Los diagramas de una misma fila corresponden a la misma cantidad de variables. Considerando de arriba hacia abajo, las filas 1, 2, 3 y 4 corresponden a los resultados obtenidos al evaluar las funciones en 3, 5, 10 y 20 variables respectivamente. En cada figura, los métodos están numerados del 1 al 3 correspondiendo a: el método propuesto, PSO Binario [36] y PSO Binario [42] respectivamente.
Como puede observarse, efectivamente, el método propuesto ofrece mejores soluciones que las otras dos alternativas de PSO.
La figura 3.10 está organizada de la misma forma que la figura 3.9 y muestra los diagramas de caja correspondientes al fitness promedio de cada una de las 40 corridas realizadas para cada función y para cada cantidad de variables consideradas. En ella puede observarse la diversidad poblacional de cada método. En general, a partir de la altura de cada caja,

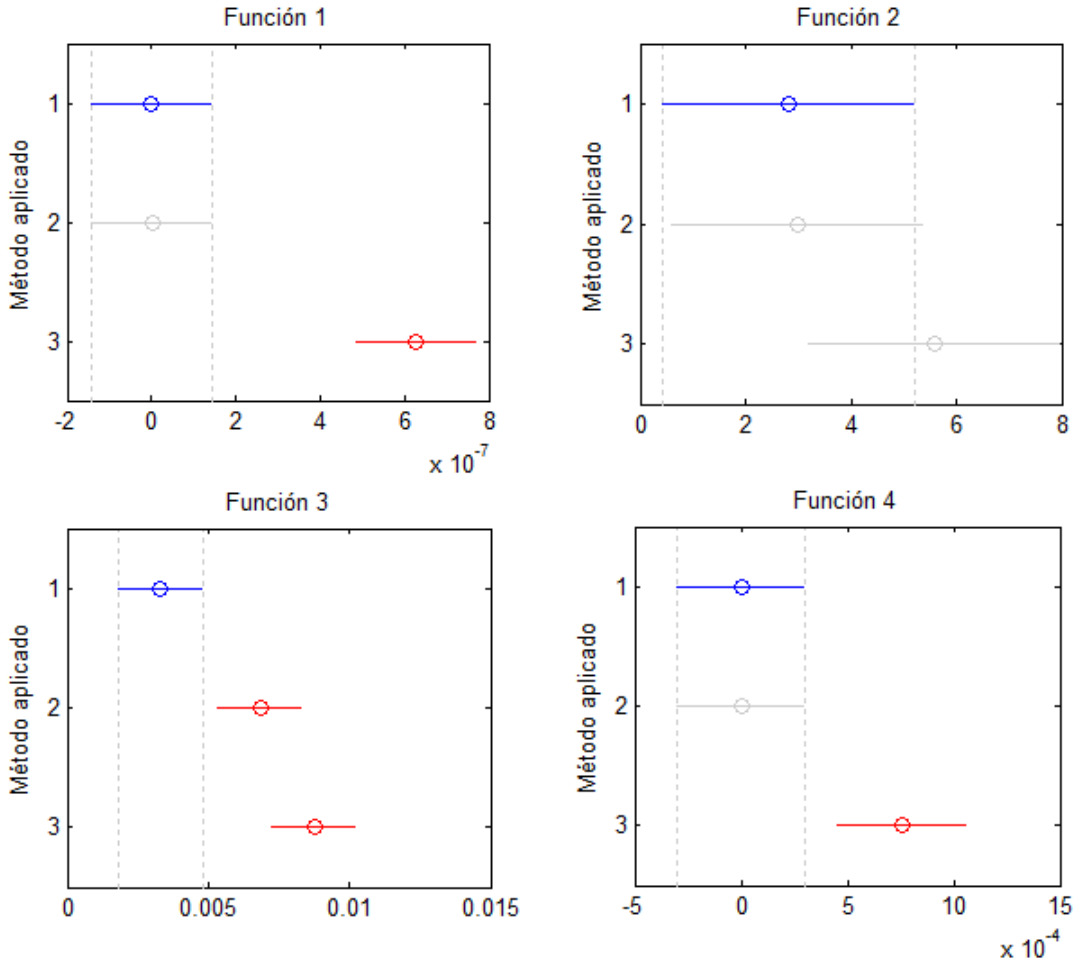

Figura 3.5: Intervalos de confianza simultáneos para el fitness promedio de la mejor solución encontrada por cada uno de los métodos utilizando 3 variables. La numeración asignada es: 1 para el método propuesto; 2 y 3 para los métodos definidos en [36] y [42] respectivamente.

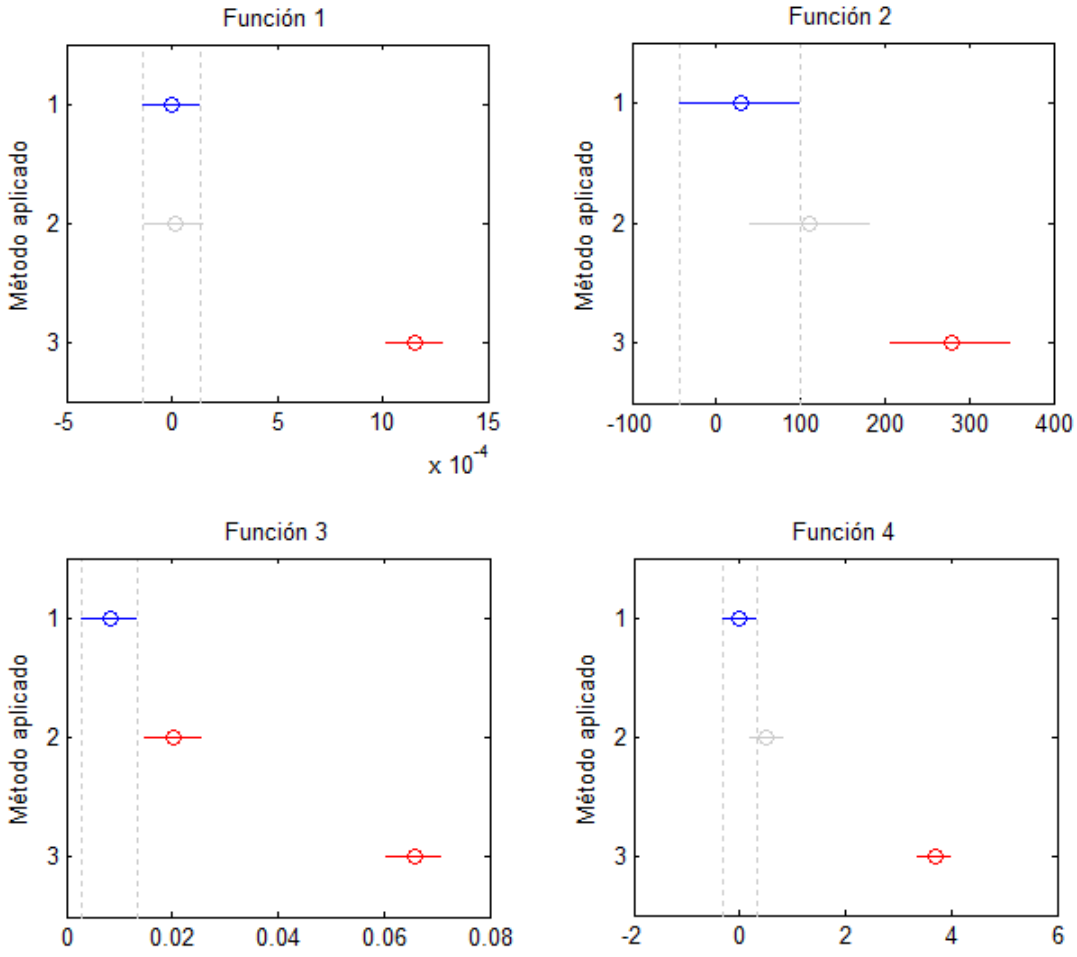

Figura 3.6: Intervalos de confianza simultáneos para el fitness promedio de la mejor solución encontrada por cada uno de los métodos utilizando 5 variables. La numeración asignada es: 1 para el método propuesto; 2 y 3 para los métodos definidos en [36] y [42] respectivamente.

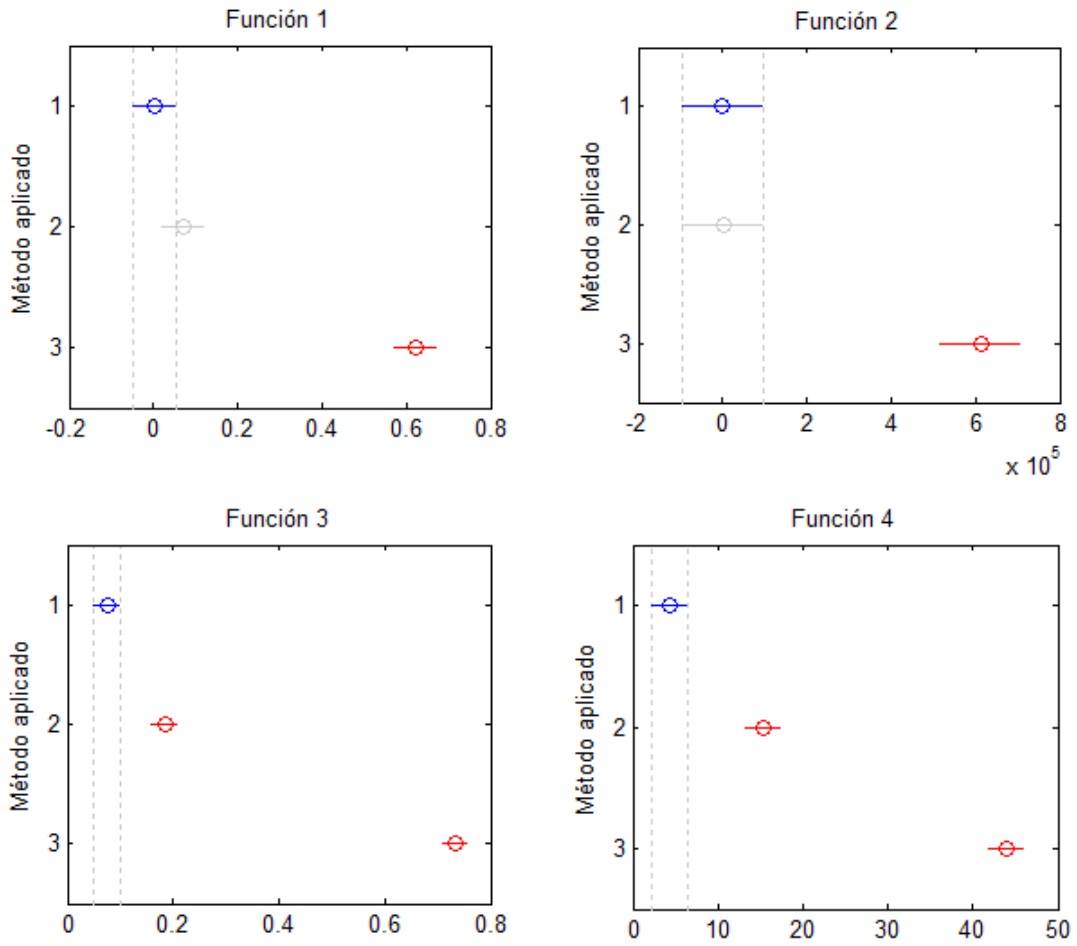

Figura 3.7: Intervalos de confianza simultáneos para el fitness promedio de la mejor solución encontrada por cada uno de los métodos utilizando 10 variables. La numeración asignada es: 1 para el método propuesto; 2 y 3 para los métodos definidos en [36] y [42] respectivamente.

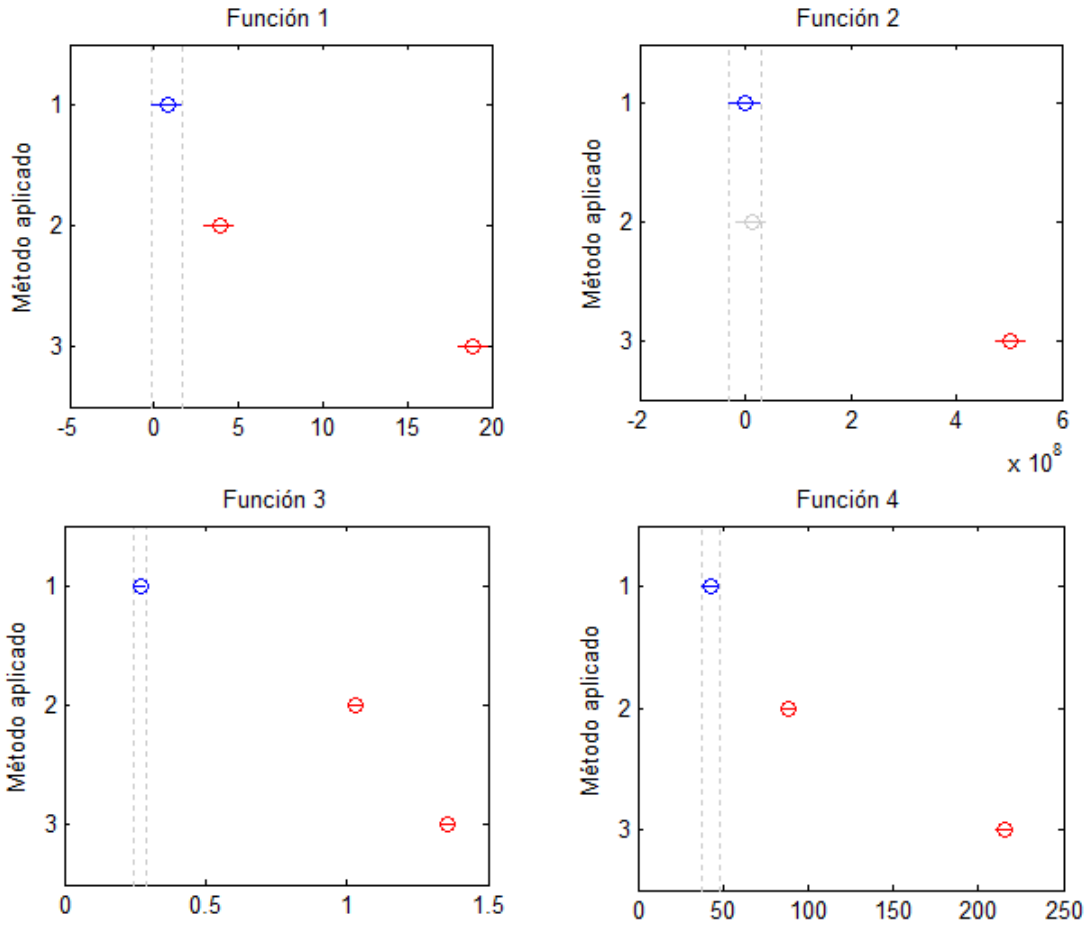

Figura 3.8: Intervalos de confianza simultáneos para el fitness promedio de la mejor solución encontrada por cada uno de los métodos utilizando 20 variables. La numeración asignada es: 1 para el método propuesto; 2 y 3 para los métodos definidos en [36] y [42] respectivamente.

Tabla 3.3: Resultados de las comparaciones de los promedios de los 40 mejores fitness obtenidos por cada uno de los métodos. Se han calculado los IC de a pares para un nivel de significación 0.05. El símbolo ▲ representa que el IC no contiene al 0 indicando que la hipótesis nula, que afirma que la media del método indicado en la fila es igual a la del método indicado en la columna, debe ser rechazada

nro Var.	Método	Binary PSO [36]				Binary PSO [42]			
3	Método propuesto	▽	▽	▲	▽	▲	▽	▲	▲
3	Binary PSO [36]					▲	▽	▽	▽
5	Método propuesto	▽	▽	▲	▽	▲	▲	▲	▲
5	Binary PSO [36]					▲	▽	▲	▲
10	Método propuesto	▽	▽	▲	▲	▲	▲	▲	▲
10	Binary PSO [36]					▲	▲	▲	▲
20	Método propuesto	▲	▽	▲	▲	▲	▲	▲	▲
20	Binary PSO [36]					▲	▲	▲	▲

puede decirse que si bien el método [42] presenta los mayores rangos intercuartiles, las soluciones que ofrece son las peores. En cuanto al método propuesto y el PSO Binario [36], los rangos son equivalentes.

3.5. Conclusiones

Se ha presentado una variante del método PSO binario propuesto originalmente en [36] que controla las modificaciones del vector velocidad utilizando una variante del método PSO continuo.
Los resultados obtenidos al minimizar un conjunto de funciones de prueba son mejores a los arrojados por los métodos definidos en [36] y [42].
La tabla 3.3 permite ver que a medida que aumenta la cantidad de variables utilizadas, la diferencia de medias se hace cada vez más significativa.
La figura 3.9 permite mostrar que en la mayoría de las ejecuciones, los resultados obtenidos por el método propuesto han sido superiores a los de los otros dos. Asimismo, en la figura 3.10 se observa que el método propuesto es el que genera los mejores resultados de fitness promedio de la población, por lo menos en las funciones de prueba evaluadas. Si se considera el rango intercuartil del fitness promedio (amplitud de los diagramas de caja de la figura 2) puede afirmarse que el método propuesto en este articulo y el PSO Binario de [36] son equivalentes, ofreciendo el primero las mejores soluciones.

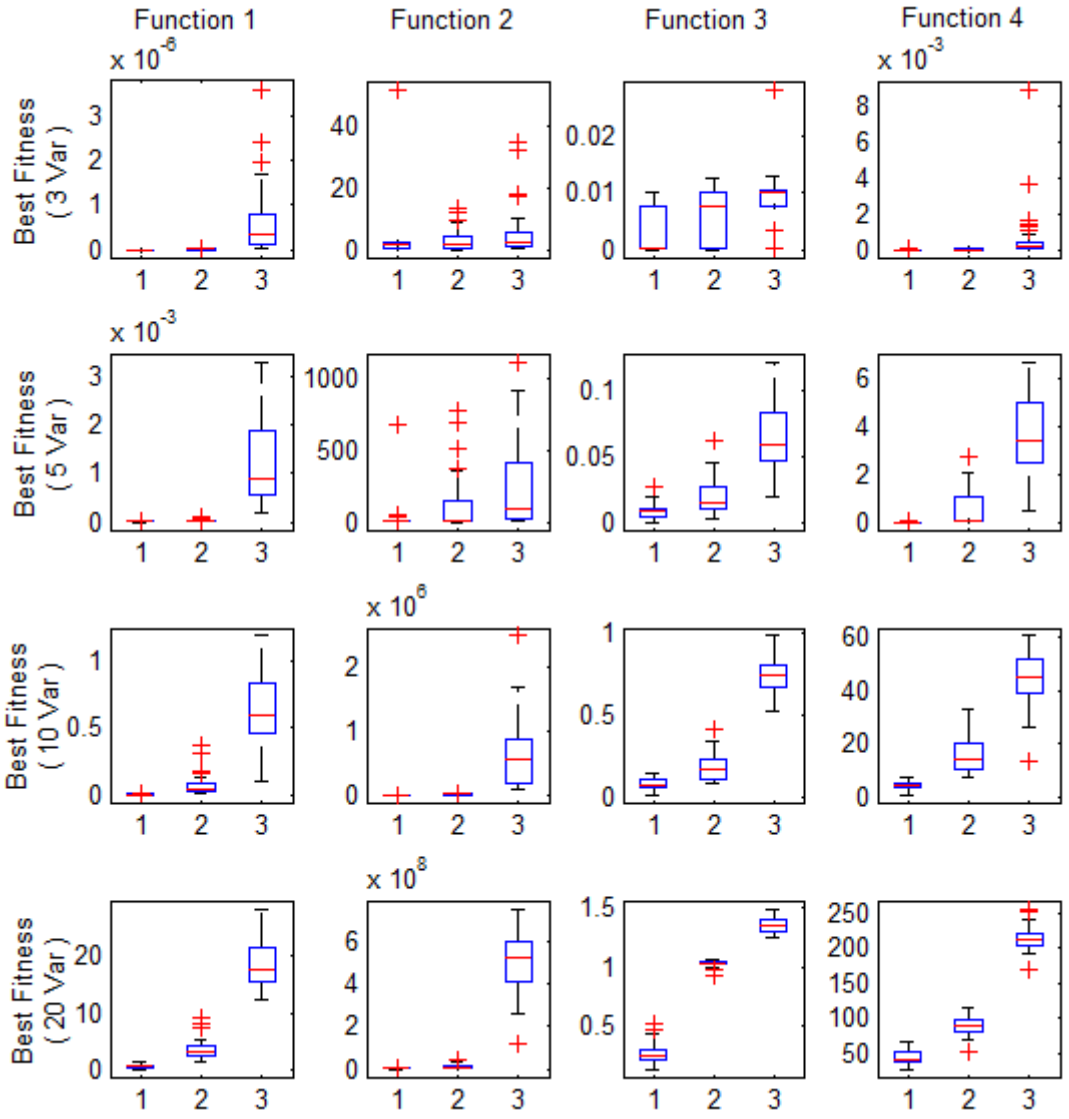

Figura 3.9: Diagramas de caja correspondientes a las mejores soluciones obtenidas en cada una de las 40 corridas independientes. Sobre el eje de las abscisas se indica el método: 1 = Método propuesto, 2 = PSO Binario [36] y 3 = PSO Binario [42]. Cada fila indica los resultados obtenidos con 3, 5, 10 y 20 variables

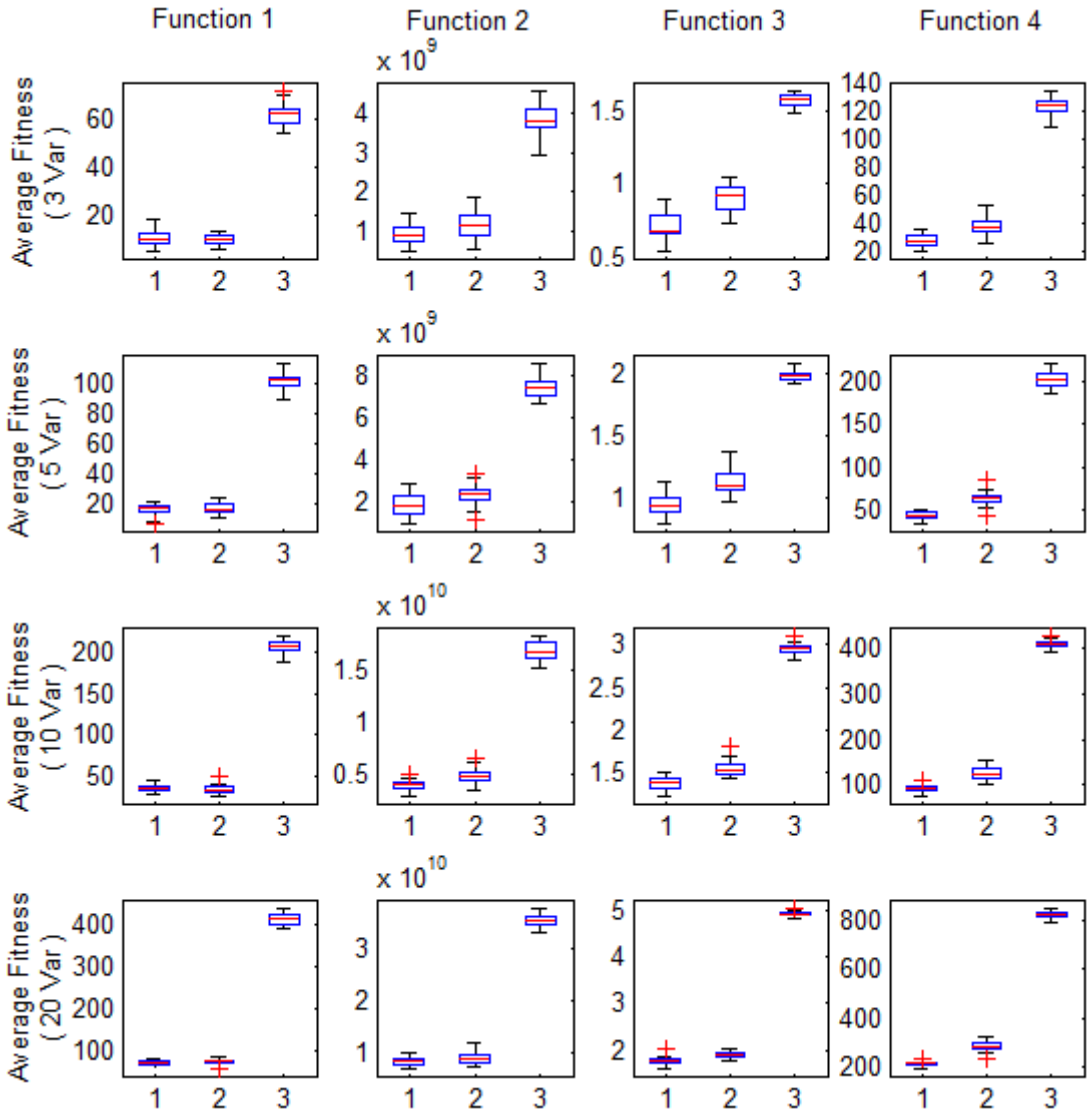

Figura 3.10: Diagramas de caja correspondientes al fitness promedio de cada una de las 40 corridas independientes. Sobre el eje de las abscisas se indica el método: 1 = Método propuesto, 2 = PSO Binario [36] y 3 = PSO Binario [42]. Cada fila indica los resultados obtenidos con 3, 5, 10 y 20 variables.

Actualmente, el énfasis de la investigación está puesto en medir la performance del algoritmo propuesto cuando el número de dimensiones del problema crece con el objetivo de aplicarlo a la resolución de problemas del mundo real.
También resulta de interés analizar su performance utilizando PSO de población variable [28]. Esto permitiría adaptar la cantidad de individuos del cúmulo (swarm) en función de la complejidad del problema a resolver.

Capítulo 4

Método SIFT

4.1. Introducción

La detección automática de objetos o patrones en imágenes tiene aplicaciones en temáticas tan diversas como la industria, seguridad, medicina, robótica y demás.

El reconocimiento de objetos o patrones es una técnica que se utiliza para lograr individualizar, reconocer y clasificar objetos, formas o estructuras en las imágenes analizadas.

Las investigaciones en visión por computadora abordaron el problema del reconocimiento de objetos basándose, en un principio, en la geometría de los mismos[43][44][45][46].

En la actualidad, se utilizan principalmente regiones concretas de la imagen en las que se considera que existe información suficientemente útil y repetible como para caracterizar dicho objeto. Esta información se almacena en vectores de características denominados descriptores.

La estrategia para llevar a cabo el reconocimiento de objetos, consiste en aplicar a una imagen un método automático de detección y extracción de características con el fin de obtener un conjunto de descriptores. Luego, este mismo método se aplica en diversas imágenes para detectar si contienen un conjunto de descriptores similares al del objeto que se quiere reconocer.
El proceso de reconocimiento consta de las siguientes etapas:

- Detección y obtención de los descriptores de la imagen que contiene el objeto a reconocer.
- Detección y obtención de los descriptores en una o varias imágenes correspondientes al objeto de interés.
- Comparación y cálculo de correspondencias entre los descriptores de la primera

imagen con las demás.

- Decidir a que imágenes corresponde el objeto a reconocer.

Este capítulo presenta una descripción detallada del método SIFT (Scale Invariant Feature Transform) el cual extrae características invariantes distintivas de un objeto presente en una imagen. Fue desarrollado por David Lowe en 1999 [47] [48] y patentado en 2004.

Las características que extrae el método SIFT son invariantes a la escala y rotación de la imagen. Además son robustas a cambios en la iluminación, oclusión parcial, ruido y pequeños cambios en el punto de vista. Son altamente distintivas, fáciles de extraer y permiten la correcta identificación de objetos con una baja probabilidad de error.

4.2. Descripción general del algoritmo

Los pasos principales que son llevados a cabo para generar los vectores de características, utilizando SIFT, son los siguientes:

- **Detección de puntos extremos en espacio-escala**: En primer lugar se buscan puntos de interés que sean invariantes a la escala. Para lograrlo se utiliza una función de diferencias de Gaussianas.
- **Localización de los puntos clave**: Se efectúa un análisis detallado de cada uno de los puntos obtenidos a fin de determinar su escala. Luego se seleccionan los puntos clave en base a medidas realizadas sobre su estabilidad.
- **Asignación de Orientaciones** Utilizando la información del gradiente se asigna una o varias direcciones a cada punto clave. De esta forma, todas las operaciones que se realicen sobre la imagen son expresadas en forma relativa a la orientación asignada, a la escala y la ubicación de cada característica lo cual otorga invarianza a las transformaciones.
- **Descriptor de cada punto clave**: Cada gradiente local (a un sector de la imagen) es medido en la escala seleccionada en una región cercana al punto clave correspondiente. Luego se realiza una cambio en la representación a fin de permitir distorsiones de forma y cambios de iluminación.

A continuación se describen estas etapas con mayor detalle.

4.3. Detección de puntos extremos en el espacio-escala

En el sentido más general, la escala en imágenes representa un grado de libertad expresado por una señal. La escala es simplemente una manifestación de un cambio en el tamaño espacial o escala, de un atributo, región o agrupamiento.

Poder reconocer a los objetos sin importar la cantidad de pixels que ocupan en una imagen es una característica deseable. De ello se ocupa el análisis de invariantes a la escala. Una forma de realizar un análisis invariable a la escala es muestrear el espacio-escala con la suficiente densidad de tal manera que sea posible rastrear la evolución de los detalles que van emergiendo cuando se pasa de una escala a otra.

Según Lindeberg [49], la representación espacio-escala es un tipo especial de representación multi-escala que incluye un parámetro continuo de escala y preserva el mismo muestreo espacial para todas las escalas.
El argumento principal para esta representación es que si no se dispone a priori de información referida a la escala adecuada para una señal entonces la única solución es representar la entrada de datos en múltiples escalas.
Esto lleva a que la representación espacio-escala de una señal es convertir la señal original en una familia de señales de un sólo parámetro construidas mediante la convolución con señales Gaussianas de varianza creciente.

Las propiedades deseables para el procesamiento multi-escala son:

- Invarianza a los desplazamientos: Isotropía espacial, todas las posiciones espaciales son tratadas en forma equitativa.
- Invarianza a la escala: Homogeneidad espacial, todas las escalas espaciales son tratadas por igual.
- Causalidad
  - No deben crearse nuevas curvas de nivel en los espacios-escala.
  - No deben crearse nuevos extremos locales.
  - No deben acentuarse los extremos locales, es decir, ningún extremo en una cierta escala se vuelve mayor en las escalas superior e inferior.

En resumen, para una señal n-dimensional $f : \mathbb{R}^n \to \mathbb{R}$ su representación en el espacio-escala es una función $L : \mathbb{R}^n x R_+ \to \mathbb{R}$ cuyo valor se obtiene convolucionando la señal inicial, $f(x)$, con un núcleo gaussiano de escala variable, $G : \mathbb{R}^n x R_+ \to \mathbb{R}$ como se indica en (4.1)

$$L(x;\sigma) = G(x;\sigma) * f(x) = \int_{\xi \in \mathbb{R}^n} f(x)\, G(x-\xi)\, d\xi \tag{4.1}$$

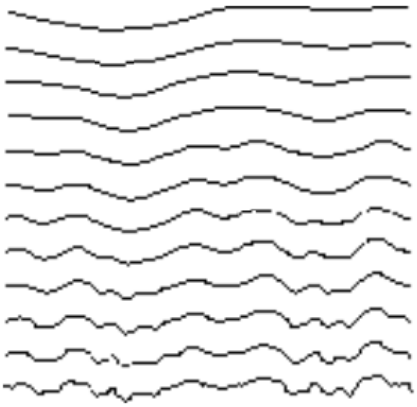

Figura 4.1: La representación espacio-escala transforma la señal de entrada en una familia de señales basadas en un único parámetro: la escala. La figura muestra una señal sucesivamente convolucionada con un kernel gaussiano de ancho creciente

donde $*$ es la operación de convolución en el vector n-dimensional x y $G(x;\sigma)$ se define como en (4.2)

$$G(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\|x\|^2}{2\sigma^2}} \tag{4.2}$$

La figura (4.1) muestra la familia de señales generadas al suprimir sucesivamente la información más detallada de la escala. La figura (4.2) repite este mismo proceso sobre una imagen.

Cuando se trata de una imagen, es decir una señal en dos dimensiones, la imagen de $f(x,y)$ en el espacio-escala es $L(x,y;\sigma)$.

Para detectar eficientemente la ubicación de puntos estables en el espacio-escala se calculan los puntos extremos en la función diferencia de Gaussianas convolucionada con la imagen, $D(x,y,\sigma)$, la cual puede ser calculada a partir de las diferencias entre dos escalas vecinas separadas por un factor constante k. La función de detección es la indicada en la ecuación (4.3)

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * f(x,y) = L(x,y,k\sigma) - L(x,y,\sigma) \tag{4.3}$$

donde L es una imagen suavizada con un filtro gaussiano.

Para cada octava del espacio-escala, la imagen inicial es convolucionada repetidamente con Gaussianas, para producir el conjunto de imágenes del espacio-escala como se muestra a la izquierda de la figura 4.3. Las imágenes Gaussianas adyacentes son restadas para producir las imágenes de diferencias de Gaussianas como se muestra a la derecha de dicha figura. Luego de cada octava, la imagen Gaussiana es sub-muestreada a la cuarta parte y el proceso se repite nuevamente.

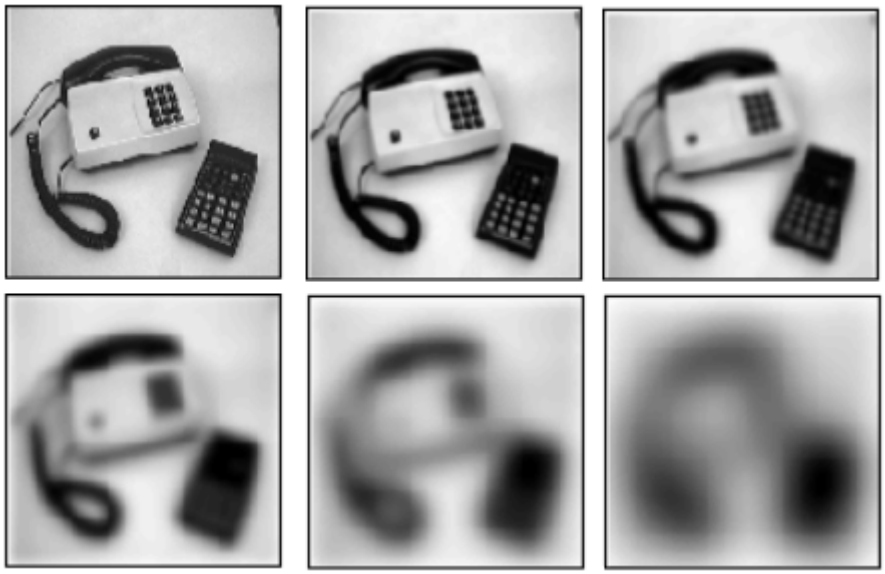

Figura 4.2: Distintos niveles en la representación espacio-escala de una imagen 2D en las escalas σ=0, 2, 8,32, 128 y 512

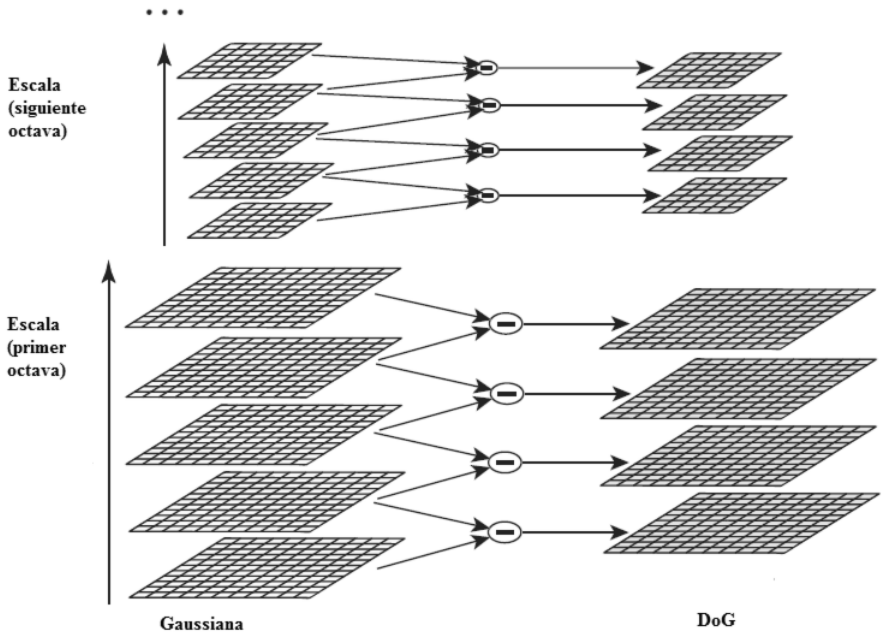

Figura 4.3: Pirámide Gaussiana y Diferencias de Gaussianas (DoG) del espacio-escala

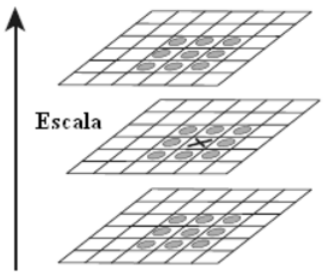

Figura 4.4: Detección de máximos y mínimos

4.3.1. Detección de extremo local

Para detectar el máximo y mínimo local de $D(x, y, \sigma)$, cada punto es comparado con sus ocho vecinos en la imagen actual y nueve vecinos en la escala anterior y posterior, como se muestra en la figura 4.4. Este punto es seleccionado solo si es más grande o más pequeño que todos sus vecinos. El costo de este chequeo es razonablemente bajo ya que antes de realizarlo se descartan los puntos de muestreo con bajo contraste y cercanos al borde, reduciendo la cantidad de puntos a analizar.

4.4. Localización exacta de puntos claves

Para cada punto extremo en el espacio-escala obtenido como resultado del paso anterior, se realiza un ajuste detallado de los datos de los vecinos en lo que se refiere a la posición, escala y proporción de las curvaturas principales.

Esta información permite rechazar los puntos con bajo contraste (sensibles al ruido) o que están mal localizados a lo largo de un borde.

La implementación inicial de este enfoque simplemente localiza puntos claves en la posición y escala del punto central de la muestra. Sin embargo, Brown ha desarrollado un método [50] para el ajuste de una función cuadrática 3D a los puntos de muestreo local para determinar la posición interpolada del máximo, y sus experimentos demostraron que esto proporciona una mejora sustancial en la correspondencia y la estabilidad. Su método utiliza el desarrollo de Taylor (hasta los términos cuadráticos) de la función del espacio-escala, $D(x, y, \sigma)$, modificado de manera que el origen está en el punto de muestreo (4.4)

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{4.4}$$

donde D y sus derivadas son evaluadas en el punto de muestra y $\mathbf{x} = (x, y, \sigma)^T$ es el desplazamiento desde ese punto. La posición del extremo, $\hat{\mathbf{x}}$, se determina tomando la

Figura 4.5: Imagen original

derivada de esta función con respecto a **x** e igualando a cero, obteniendo (4.5)

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}} \tag{4.5}$$

Según lo sugerido por Brown, el Hessiano y derivada de D se aproximan utilizando las diferencias de puntos vecinos de muestreo. El sistema lineal resultante de 3x3 se puede resolver con un costo mínimo. Si el desplazamiento $\hat{\mathbf{x}}$ es mayor que 0.5 en cualquier dimensión entonces significa que el extremo se encuentra más cercano a un punto de muestreo diferente. El desplazamiento final $\hat{\mathbf{x}}$ es agregado a la posición de ese punto de muestra para la interpolación estimada de la posición del extremo. El valor de la función en el extremo, $D(\hat{\mathbf{x}})$, es útil para rechazar extremos con bajo contraste. Esto se obtiene sustituyendo las ecuaciones (4.4) y (4.5), quedando (4.6)

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}} \tag{4.6}$$

La figura 4.6 muestra los efectos de la selección de puntos claves en una imagen de una persona. Se utiliza una imagen de 320 x 243 pixeles y se muestran los puntos claves como vectores dando la posición, la escala, y la orientación de cada punto clave. La figura 4.5.(a) muestra la imagen original de la que se obtienen los puntos claves. La figura 4.6.(a) muestra los 276 puntos claves máximos y mínimos detectados de la función de diferencias de gaussianas. En la figura 4.6.(b) se muestran los 172 puntos claves restantes removiendo aquellos con bajo contraste. La figura 4.6.(c) muestra los 228 puntos donde los cercanos al borde fueron descartados. La figura 4.6.(d) muestra los 139 puntos resultantes de haber eliminado los de bajo contraste y los cercanos al borde.

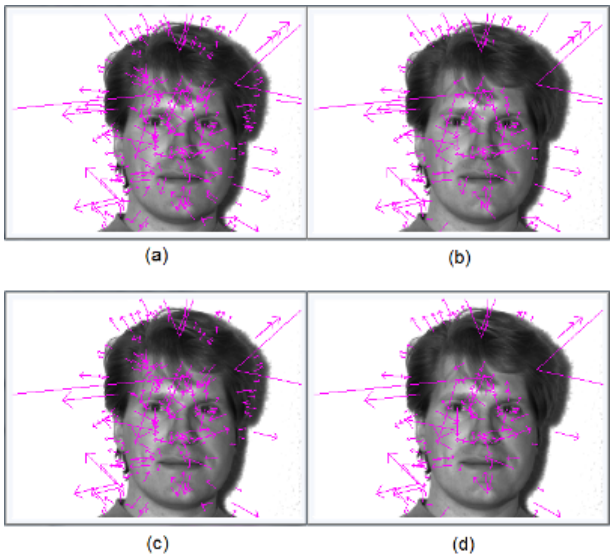

Figura 4.6: (a) Todos los puntos claves detectados. (b) Puntos claves luego de descartar los de bajo contraste. (c) Puntos claves luego de descartar los cercanos al borde. (d) Puntos claves luego de descartar los de bajo contraste y cercanos al borde

4.5. Eliminación de bordes

Para la estabilidad no es suficiente con rechazar los puntos con bajo contraste. La función de diferencia de Gaussianas tiene una respuesta firme a lo largo de los bordes, incluso si la posición a lo largo del borde no está bien determinada y por lo tanto inestable a pequeñas cantidades de ruido. Un pico mal definido en la función de diferencia de gaussianas tendrá una gran curvatura principal a través del borde, pero una pequeña en la dirección perpendicular. Las curvaturas principales pueden ser calculadas desde una matriz Hessiana, H, de 2x2 calculada en la posición y escala del punto clave como se muestra en (4.7):

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{4.7}$$

Las derivadas son estimadas tomando las diferencias entre puntos vecinos. Los valores propios de H son proporcionales a las curvaturas principales. Sea α el valor propio con la mayor magnitud y β el más pequeño, se puede calcular la suma de los valores propios a partir de la traza de H (4.8) y el producto a partir de los determinantes (4.9):

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{4.8}$$

$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \tag{4.9}$$

En el caso de que el determinante sea negativo, lo cual es poco probable, las curvaturas tienen signos diferentes de modo que el punto se descarta por no ser un extremo. Sea r la relación entre el valor propio de la magnitud más grande y la más pequeña, de modo que $\alpha = r\beta$, tomando las ecuaciones (4.8) y (4.9) se llega a la ecuación (4.10)

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{4.10}$$

La ecuación (4.10) solo depende de la relación de los valores propios en lugar de sus valores individuales. La cantidad $(r+1)^2/r$ es mínima cuando los dos valores propios son iguales y se incrementa con r. Por lo tanto, para comprobar que la relación de las curvaturas principales está por debajo de cierto umbral, r , sólo es preciso verificar que se cumpla la ecuación (11)

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{4.11}$$

Se utiliza $r = 10$, que elimina puntos claves que tienen una relación entre las curvaturas principales mayor que 10. La transición de la figura 4.6.(b) a la 4.6.(c) muestra los efectos de esta operación [51] [48].

4.6. Asignación de la orientación

En base a las propiedades de la imagen local se asigna una orientación a cada punto clave de manera que el descriptor correspondiente pueda ser representado en relación a dicha orientación. De esta forma, el descriptor será invariante a la rotación de la imagen.

La escala del punto clave es usada para seleccionar la imagen suavizada Gaussiana, L, cuya escala esté más cerca, para que todos los cálculos sean realizados de manera invariante a la escala. Para cada muestra de imagen, $L(x, y)$, en esa escala, la magnitud del gradiente, $m(x, y)$, y la orientación $\theta(x, y)$ se calculan usando diferencia de pixeles como se muestra en las ecuaciones (4.12) y (4.13) respectivamente:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{4.12}$$

$$\theta(x,y) = tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \tag{4.13}$$

Luego se considera una región alrededor del punto clave y se construye un histograma con las orientaciones de los puntos de dicha región. El histograma tiene 36 niveles de discretización con el objetivo de cubrir todas las posibles orientaciones (360 grados).

Las direcciones dominantes correspondientes a los gradientes locales forman picos dentro del histograma. Para las direcciones correspondientes al pico más alto del histograma y todas aquellas que se superen el 80 % de su valor, se crean puntos claves con sus respectivas orientaciones. De esta forma, las ubicaciones que en el histograma den lugar a picos de magnitud similar darán lugar a la creación de puntos clave con igual ubicación y escala pero distinta orientación. Según Lowe, sólo el 15 % de los puntos reciben orientaciones múltiples contribuyendo de esta forma a la estabilidad del proceso de reconocimiento [48].

4.7. Descriptor de la imagen local

Las operaciones anteriores han asignado una posición, una escala y una orientación a cada punto clave de la imagen. El próximo paso es calcular un descriptor para la región de imagen local que sea altamente distintivo, tan invariante como sea posible a las variaciones restantes, tales como cambios en la iluminación o punto de vista 3D.

4.7.1. Representación del descriptor

La figura 4.7 ilustra el cálculo del descriptor de puntos claves. Primero las magnitudes del gradiente de la imagen y orientaciones son probadas alrededor de la posición del punto clave, usando la escala del mismo para seleccionar el nivel de borroneado Gaussiano para la imagen. A fin de lograr la invariancia de orientación, las coordenadas del descriptor y las orientaciones del gradiente se rotan en relación con la orientación del punto clave. Por eficiencia, los gradientes son precalculados para todos los niveles de la pirámide. Estos están ilustrados con pequeñas flechas en cada posición de la figura 4.7.(a).

Un descriptor se crea mediante el cálculo de la magnitud del gradiente y la orientación en cada punto de muestreo de la imagen en una región alrededor de la ubicación del punto clave, como se muestra en la figura 4.7.(a). Estos son ponderados por una ventana Gaussiana, indicada por el círculo superpuesto. Estas muestras se acumulan en los histogramas de orientación que resumen el contenido sobre subregiones de 4x4, como se muestra en la figura 4.7.b), con la longitud de cada flecha correspondiente a la suma de las magnitudes del gradiente cerca de esa dirección dentro la región. La figura 4.7.(b) muestra un descriptor obtenido de un arreglo de 2x2 cuyos valores son calculados a partir de una región de 8x8 mientras que el método SIFT utiliza un arreglo de 4x4 calculado sobre una muestra de 16x16. De todas formas, el gráfico es representativo.

El descriptor está formado por un vector que contiene los valores de todas las entradas del

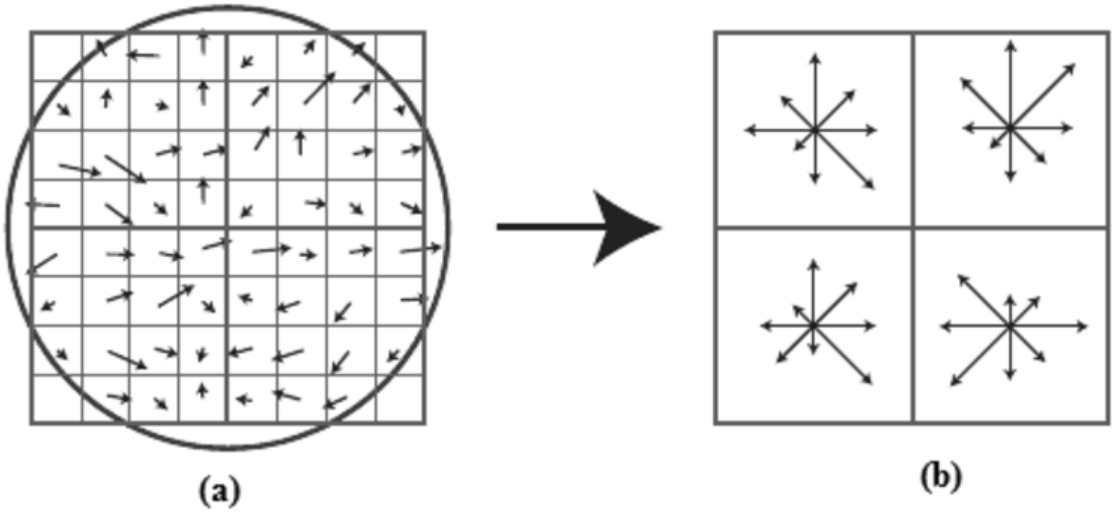

Figura 4.7: (a) gradientes de la imagen, (b) descriptor del punto clave

histograma de orientación, que corresponden a las longitudes de las flechas de la figura 4.7.(b). utilizando un arreglo de 4x4.

Por lo tanto, se utilizan $4 * 4 * 8 = 128$ elementos del vector de características para cada punto clave. Finalmente, el vector de características es modificado para reducir los efectos del cambio de iluminación. Primero, el vector se normaliza a la longitud unidad. Luego se realiza un cambio de contraste de la imagen multiplicando el valor de cada pixel por una constante. Nótese que los gradientes serán multiplicados por la misma constante, por lo que este cambio de contraste será cancelado por la normalización del vector. También se produce un cambio de brillo añadiendo una constante a cada píxel de la imagen. Esto tampoco afecta los valores del gradiente, ya que son calculados a partir de diferencias de píxeles. Por lo tanto el descriptor es invariante a los cambios en la iluminación.

Sin embargo, los cambios de iluminación no lineales también pueden ocurrir debido a la saturación de la cámara o a los cambios de iluminación que afectan a las superficies 3D con diferentes orientaciones en diferentes cantidades. Estos efectos pueden causar un gran cambio en las magnitudes relativas de algunos gradientes, pero tienen menos probabilidades de afectar a las orientaciones del gradiente. Por lo tanto, se reduce la influencia de las grandes magnitudes de gradientes haciendo una umbralización de los valores del vector de características para que no supere el valor de 0.2, y luego se realiza una renormalización [48].

4.8. Trabajos Relacionados

Esta sección describe brevemente otros métodos que, al igual que SIFT, permiten reconocer objetos en una imagen por medio de descriptores.

Como podrá observarse, o bien son variantes de SIFT o bien son específicos en la solución de algún tipo de problema. En esta tesis se utilizó SIFT ya que no sólo obtiene buenos resultados sino que es lo suficientemente general como para brindar una solución robusta ante cambios de distintos aspectos de la imagen.

- PCA-SIFT, el algoritmo se basa en el código original del descriptor SIFT, modificando solo la cuarta etapa del mismo, que es la generación de descriptores de puntos claves [52]. Acepta la misma entrada que el descriptor SIFT: la ubicación del sub-píxel, la escala, y las orientaciones dominantes del punto clave. El resultado del algoritmo es, por cada punto clave, un vector de dimensión 3042 que luego se reduce a dimensión 32 utilizando PCA (Principal Component Analysis).
- SURF (Speeded Up Robust Features) es un descriptor/detector de puntos de interés invariante a escala y rotación que propone aproximar o incluso superar los esquemas previamente propuestos con respecto a repetibilidad, distintibidad y robustez [53]. SURF esta inspirado en SIFT y se basa en imágenes integrales para convulsiones de imágenes con la intención de reducir el tiempo de procesamiento, se basa en las fortalezas de los detectores y descriptores líderes existentes. Describe una distribución de las respuestas Haar wavelet en el vecindario del punto de interés. Las imágenes integrales son usadas para velocidad y sólo son usadas 64 dimensiones reduciendo el tiempo del cómputo y comparación de característica. La etapa de indexado está basada en Laplacian, lo que aumenta la velocidad de comparación y la robustez del descriptor. El propósito del descriptor SURF se basa en propiedades similares a las del descriptor SIFT, pero con más complejidad. El primer paso consiste en fijar orientaciones basadas en la información de una región circular alrededor del punto de interés. Luego, se construye una región cuadrada alineada con la orientación seleccionada, y se extrae el descriptor SURF de la misma. SURF es, hasta cierto punto, similar en su concepción a SIFT, ya que ambos se centran en la distribución espacial de la información del gradiente.
- RIFT (Rotation Invariant Feature Transform) el descriptor es construido usando regiones circulares normalizadas divididas en anillos concéntricos de igual ancho y con cada anillo se computa un histograma de orientación del gradiente [54]. Para mantener la invariancia de rotación, la orientación es medida en cada punto crítico con respecto a la dirección hacia afuera desde el centro. A diferencia de SIFT, la orientación está representada por 8 intervalos distintos, por lo que el Descriptor final tiene un total de 24 entradas.

- G-RIF (Generalized Robust Invariant Feature) es un descriptor de contexto general que codifica la orientación de borde, la densidad de borde y la información de tonalidad en una forma unificada combinando la información sensorial con la codificación espacial [55]. Este enfoque se compone de la detección, descripción y clasificación de la parte visual. El primer paso es la detección de la parte visual (orientación espacial y bordes). Para que el reconocimiento de objetos sea exitoso, las partes visuales deben cumplir algunos requisitos. En primer lugar, la información del fondo debería evitarse mientras sea posible. En segundo lugar, deben ser perceptualmente significativas. Por último, deben ser resistentes a los efectos de las distorsiones geométricas y fotométricas. El esquema de reconocimiento de objetos utiliza el contexto de vecindad basado en votación para estimar el modelo de objetos.

4.9. Conclusiones

Los descriptores SIFT detallados en este capítulo son especialmente útiles debido a su carácter distintivo, lo que permite la correspondencia de objetos en diferentes imágenes.

La correspondencia de características locales se ha convertido en el método más usado para comparar imágenes. Varios métodos han sido propuestos. Sin embargo, no ha sido establecido cual es el descriptor más apropiado y cual tiene el mejor rendimiento. La elección del método depende principalmente de la aplicación o problema a resolver. No hay un método que sea el óptimo en mediciones de tiempo, escala, rotación, iluminación, etc. El método SIFT fue presentado por Lowe para la extracción de características distintivas de imágenes logrando ser invariantes a la escala y rotación. Este fue ampliamente utilizado en mosaico de imágenes, reconocimiento, recuperación, etc. Luego, Ke and Sukthankar utilizaron PCA para normalizar el camino del gradiente en lugar del histograma, mostrando así que PCA basado en descriptores locales, es también distintivo y robusto a las deformaciones de las imágenes. Sin embargo ambos métodos eran todavía muy lentos por lo que Bay y Tuytelaars presentaron SURF, demostrando en sus experimentos que dicho método era más rápido. SIFT, con su alta precisión y robustez a los errores de localización y relativamente bajo tiempo de cómputo, se ha convertido en el estándar. En comparación, PCA-SIFT es más distintivo y más compacto, con lo cual lleva a lograr mejoras de velocidad. Sin embargo presenta bajo rendimiento para imágenes borrosas y/o de poca resolución. SURF muestra gran desempeño de estabilidad y velocidad. Por otro lado presenta cierta inestabilidad a la rotación y cambios de iluminación.

El método RIFT es conveniente para modelar una amplia gama de transformaciones en 3D, incluyendo el cambio de punto de vista y las deformaciones no rígidas.

El método G-RIF muestra un mejor rendimiento en el reconocimiento de objetos en

entornos severos (poca iluminación y/o bajo contraste) que el resto de los métodos.

Finalmente, SIFT demuestra ser la mejor opción dado su fuerte invarianza a los cambios de escala y rotación y a su invarianza parcial frente a cambios de puntos de vista e iluminación. Es uno de los métodos más utilizados, sin embargo, su costo computacional es alto si se piensa en su aplicación en tareas tales como tracking de video.

Capítulo 5

Reconocimiento de Rostros

5.1. Introducción

El reconocimiento de rostros es una técnica biométrica muy utilizada en diversas áreas como seguridad y control de accesos, medicina forense y controles policiales. Se trata de identificar si la imagen del rostro de una persona se corresponde o no con alguna de las imágenes existentes en una base de datos. Este problema es difícil de resolver automáticamente debido a los cambios que distintos factores, como la expresión facial, el envejecimiento e incluso la iluminación, producen en la imagen.

Este capítulo propone utilizar sólo los descriptores SIFT más representativos de la imagen logrando un buen reconocimiento y resolviendo a la vez los dos grandes problemas que posee esta caracterización: la detección de falsos positivos y el tiempo requerido para realizar el reconocimiento.

La selección de los vectores de características SIFT se realiza mediante una variante de PSO (Particle Swarm Optimization) binario y se aplica únicamente a los vectores de las imágenes de la base por lo que su procesamiento se realiza en una etapa previa al momento en que debe realizarse el reconocimiento.

5.2. Trabajos Relacionados

Actualmente existen distintas soluciones a este problema utilizando descriptores SIFT. En [56] se ha demostrado que usar características SIFT para el proceso de reconocimiento de rostro es superior a los conocidos algoritmos Eigenfaces y Fisherfaces. Se utilizaron diferentes tamaños en el conjunto de datos de entrenamiento, dónde se comprobó que el rendimiento decae cuando dicho conjunto es de menor tamaño. Con respecto al número significativo de características SIFT necesarias para una comparación confiable,

se comprobó que usando una menor cantidad de las características el rendimiento es mejor que Eigenfaces y Fisherfaces.

Para abordar el problema de comparar un gran número de características dimensionalmente grande entre todas las imágenes en una base de datos, en [57] se propone una clasificación discriminatoria de características SIFT que es utilizada con el fin de recortar el número de características SIFT para el reconocimiento facial. De esta manera se reduce el número de comparaciones y el proceso de reconocimiento se hace más rápido. Este proceso también filtra características que son irrelevantes para el reconocimiento de rostro obteniendo un aumento en la precisión del reconocimiento.

Por otro lado, en [58] se presenta un algoritmo de reconocimiento de rostros que utiliza el algoritmo PSO binario con motivo de explorar el espacio de solución para un óptimo subconjunto de características con el fin de incrementar la taza de reconocimiento y la separación de clases. El algoritmo propuesto se aplica a los vectores de características extraídos utilizando la Transformada de Coseno Discreta (DCT) y Transformada Wavelet Discreta (DWT).

5.3. Método Propuesto

Para realizar el reconocimiento de rostros, el método propuesto utiliza una base de datos de tamaño mínimo, formada por los vectores SIFT de las imágenes de entrada.
La selección de los vectores busca reducir a la vez el tiempo de cálculo requerido para efectuar las comparaciones necesarias así como la detección de falsos positivos. El armado de la base se realiza en forma previa al reconocimiento por lo que su construcción no afecta el tiempo de respuesta para el usuario final.

El reconocimiento de un nuevo rostro implica los siguientes pasos:

- Calcular los vectores SIFT correspondientes a la imagen de entrada.
- Comparar cada vector de la base de datos con el conjunto de vectores del nuevo rostro acumulando las coincidencias no por imagen sino por el número de la persona a la cual corresponde el vector de la base de datos.
- El nuevo rostro corresponderá a la persona que mayor cantidad de coincidencias haya acumulado.

Es importante destacar que la comparación de cada vector de la base con el conjunto de vectores de la imagen que se desea reconocer es una tarea netamente paralela. Si se dispusiera de una arquitectura de cómputo paralela podría particionarse la base de datos de vectores SIFT de manera que cada procesador tuviera la información de los vectores de

una misma persona o, mejor aún, los de una misma imagen. De esta forma, se aceleraría el cálculo de la cantidad de correspondencias encontradas.

En cuanto al reconocimiento del nuevo rostro, puede utilizarse un umbral mínimo de coincidencias para identificar los rostros que no se corresponden con ninguno de los de la base.

A continuación se describe la manera de seleccionar los vectores SIFT que formarán la base de datos.

5.3.1. Construcción de la base de datos

El método comienza obteniendo todos los vectores SIFT correspondientes a cada imagen de entrada. La selección de los vectores SIFT más representativos se realiza aplicando una variante del método descripto en la sección 4 donde cada imagen de la base tiene su propia población.

La longitud del vector posición de cada partícula de una población estará determinada por la cantidad de vectores SIFT de la imagen correspondiente. Es decir que la longitud de las partículas de poblaciones distintas puede ser diferente.

El vector correspondiente a la j-ésima partícula de la subpoblación i tiene la siguiente forma

$$X_j^i = (x_{j1}^i, x_{j2}^i, \ldots, x_{jm_i}^i) \tag{5.1}$$

donde m_i es la cantidad de descriptores SIFT de la imagen i y x_{jk}^i vale 1 si el k-ésimo descriptor SIFT debe ser incluido en la base y 0 si no.

Este criterio de especificación permite que el movimiento de cada partícula se calcule sólo a partir de los vectores SIFT de una misma imagen. De esta forma, cada población realiza la búsqueda en una parte diferente del espacio de soluciones. La solución final buscada surge de la concatenación de los mejores individuos de cada población. Esto puede ser expresado de la siguiente forma:

$$X = (X_{best}^1, X_{best}^2, \ldots, X_{best}^M) \tag{5.2}$$

donde M es la cantidad de imágenes distintas utilizadas para formar la base y X_{ibest} es el mejor individuo de la i-ésima subpoblación.

En cada iteración se reduce el valor de w, como se indica en [36]. Se ha utilizado elitismo de manera que, si al mover los individuos no se ha logrado al menos mantener el valor de fitness más alto encontrado hasta el momento, el mejor individuo de la iteración anterior sea devuelvo a su posición previa, recuperando el valor de fitness perdido. El algoritmo termina cuando se alcanza la cantidad máxima de iteraciones o cuando luego de un cierto número de iteraciones consecutivas no se producen cambios en el fitness del mejor individuo.

El algoritmo 3 resume lo antes expuesto.

Algorithm 3: Pseudocódigo del método propuesto para seleccionar los vectores SIFT que formarán la base de datos

```
ImList ← { lista de imágenes de entrada };
w_ini ← aceleración inicial ;
w_fin ← aceleración final ;
MAX_ITERA ← máxima cantidad de iteraciones ;
Popsize ← Cant.de elementos de cada subpoblación ;
Pop = CrearPobIniciales(Popsize, imList);
ite = 0;
while no se alcance la condición de terminación do
    ite = ite + 1;
    w = w_ini − (w_ini − w_fin) ∗ ite/MAX_ITERA;
    for i = 1 to Cant.de imágenes de entrada do
        Guardar la mejor partícula de la subpoblación Pop(i);
        Mover las partículas de Pop(i) según PSO Binario definido en (XX);
        Evaluar el fitness de todas las partículas de Pop(i);
        Aplicar elitismo restaurando la mejor di es necesario;
    salida ← Concatenación de las mejores partículas de cada subpoblación;
    FitnessSalida ← Suma de los fitness de las mejores partículas de cada
    subpoblación;
```

5.3.2. Evaluar el valor de fitness de cada partícula

Resta describir la manera en que se mide el fitness de cada partícula. Debe utilizarse una expresión que reduzca los falsos positivos. Por lo tanto, su valor se incrementa cuando el vector seleccionado posee una correspondencia en una imagen del sujeto al cual pertenece y se decrementa cuando pasa lo contrario.

Sea X_j^i el vector posición de la j-ésima partícula de la subpoblación i, como se definió en (5.1).

Sea $C1_{jk}^i$ el total de coincidencias entre el k-ésimo vector SIFT de la imagen i y el resto de las imágenes que corresponden al mismo sujeto que el representado por la imagen i.

Sea $C2_{jk}^i$ el total de coincidencias entre el k-ésimo vector SIFT de la imagen i y las imágenes que corresponden a sujetos distintos al representado por la imagen i.

El fitness de la j-ésima partícula de la subpoblación i se calcula como

$$Fit^i_j = \sum_{k=1}^{m} x^i_{jk} * (\alpha_1 * C1^i_{jk} - \alpha_2 * C2^i_{jk}) \tag{5.3}$$

donde α_1 y α_2 son constantes con valores entre (0,1) y cuantifican la importancia que cada término tiene dentro de la expresión. Recuerde que x^i_{jk} vale 1 si el k-ésimo descriptor SIFT de la imagen i debe incluirse en la base de datos y 0 si no.

5.4. Resultados obtenidos

Las mediciones fueron realizadas utilizando dos bases de datos obtenidas de [59]. La primera es la base de rostros YALE que contiene 165 imágenes de 15 sujetos distintos con 11 imágenes por persona. Cada imagen tiene una resolución de 320x243 pixels.

La segunda base de rostros es la base $AT\&T$ que contiene 400 imágenes de 40 personas con 10 imágenes por individuo. El tamaño de cada imagen es de 112x92 pixels.

Las imágenes disponibles han sido divididas en dos partes:

- Subconjunto de imágenes de entrada, cuyos vectores serán seleccionados mediante el método propuesto en la sección 3.3
- Subconjunto de imágenes de testeo que serán comparadas contra los vectores SIFT seleccionados a fin de efectuar el reconocimiento.

Los vectores SIFT iniciales para cada imagen han sido determinados utilizando un umbral de 0.5 como se recomienda en [47].

Los parámetros utilizados por PSO, en ambos casos, son los siguientes:

- Valores de inercia inicial y final: 1.2 y 0.2 respectivamente.
- Máxima cantidad de iteraciones = 500,
- α_1= 1/(cantidad de imágenes de entrada),
- α_2= 16/(cantidad de imágenes de entrada),

Se realizaron 30 corridas independientes del proceso descripto en la sección 3.3 variando el porcentaje de imágenes de la base utilizado.

La figura 5.1 muestra el porcentaje de aciertos calculado sobre las imágenes de testeo. Como puede observarse, la selección de los descriptores SIFT utilizando PSO ayuda al proceso de reconocimiento y brinda una tasa de acierto superior.

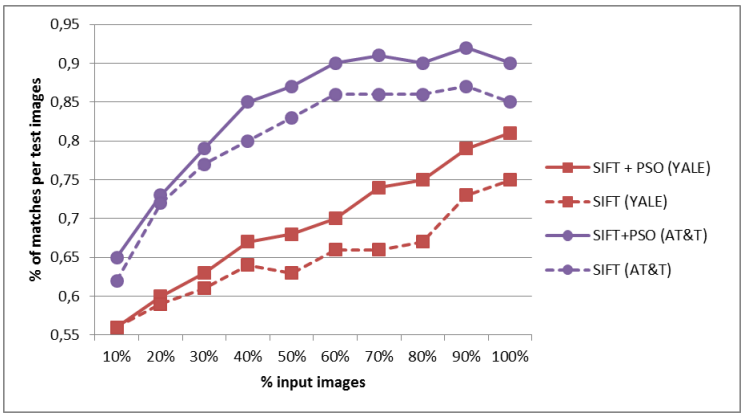

Figura 5.1: Porcentaje de aciertos para las imágenes de testeo usando el método propuesto (SIFT+PSO) y el método SIFT original para distintos porcentajes de imágenes de las bases YALE y AT&T

Otro aspecto a tener en cuenta es la precisión de la respuesta obtenida. Esto se relaciona con la similitud que cada descriptor SIFT de la imagen a clasificar tiene con los descirptores almacenados en la base. Es importante que exista una marcada diferencia entre los dos mejores candidatos encontrados para poder afirmar con certeza que corresponde a una imagen determinada. La figura 5.2 muestra que las diferencias promedio entre las dos mejores soluciones encontradas son más marcadas si se seleccionan los descriptores utilizando PSO. Esto permite afirmar que la respuesta de la clasificación es más contundente que utilizando directamentee todos los descriptores SIFT identificados por el método de Lowe.

Finalmente, la figura 5.3 muestra la cantidad de descriptores SIFT utilizados en promedio para cada imagen de la base. Puede verse que, si bien la reducción en la cantidad de descriptores es mayor para YALE que para AT&T, en ambos casos es significativa.

La figura 5.4 muestra los descriptores SIFT originales en la fila superior y los descriptores seleccionados por el método propuesto en la fila inferior.

5.5. Conclusiones

Se ha descripto una estrategia que permite realizar el reconocimiento de rostros a partir de descriptores SIFT que posee la capacidad de reducir el tamaño de la base de datos utilizando una variación del método PSO Binario estandard; este método fué descripto en la sección 3.3.

Las pruebas realizadas sobre las bases YALE y AT&T han permitido demostrar que el método propuesto alcanza tasas de reducción considerables (50 % en YALE y 25 % en

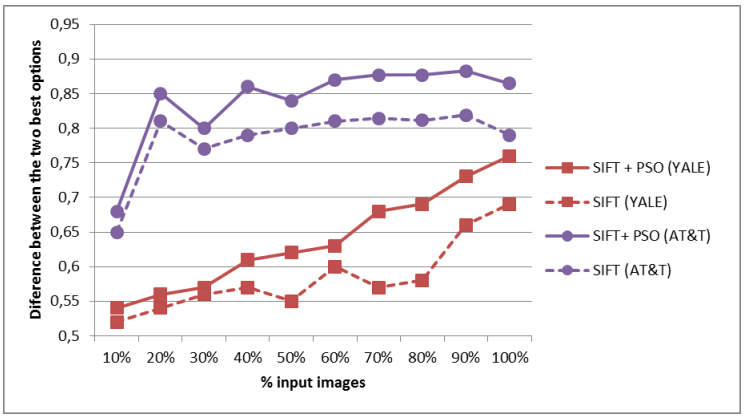

Figura 5.2: Valor promedio por imagen de la diferencia entre los dos valores más altos de correspondencias correctas, dividido por el número total de resultados encontrados para las bases de datos de YALE y AT&T

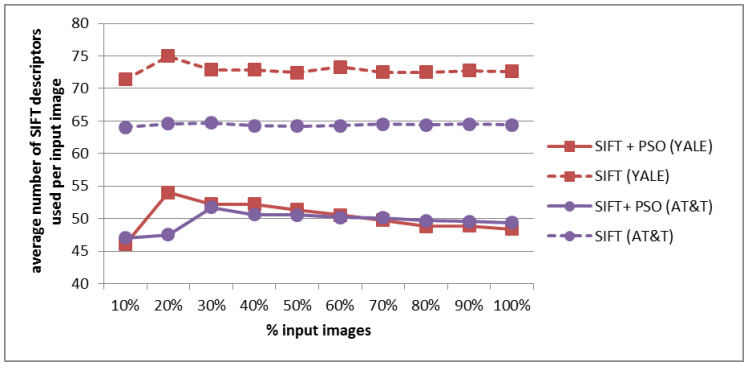

Figura 5.3: Cantidad promedio de descriptores SIFT utilizados para cada imagen de las bases de datos YALE y AT&T

Figura 5.4: Descriptores SIFT de una persona de la base de datos YALE. La fila superior muestra todos los descriptores encontrados, mientras que la fila inferior muestra sólo los descriptores seleccionados por el método propuesto.

AT&T).

Si bien la tasa de aciertos para cada imagen de testeo utilizando la base de descriptores seleccionados con PSO es ligeramente superior al proceso de utiliza todos los descriptores SIFT, la proporción de falsos positivos es menor. Además, la reducción del tamaño de la base permite garantizar una clara reducción del tiempo necesario para el reconocimiento.

Es necesario continuar analizando los parámetros involucrados a fin de determinar si efectuando un ajuste más preciso es posible reducir la número máximo de iteraciones necesarias para alcanzar una selección de descriptores optima. En este sentido, la paralelización del algoritmo propuesto es también un tema de interés.

Capítulo 6

Conclusiones generales

El aporte central de esta tesis es la presentación de una nueva técnica de optimización. Por tal motivo, comienza con una introducción general al tema donde se distinguen las dos categorías más grandes: las técnicas que proponen hallar una solución exactas y las alternativas para encontrar una solución aproximada. Hallar una solución óptima de manera exacta es un proceso que requiere un costo computacional muy alto y por tal motivo, las soluciones aproximadas cobran interés. Es decir que se está dispuesto a perder algo de precisión (dentro de ciertos límites) con el objetivo de reducir el tiempo de cálculo necesario para obtener la respuesta.

Dentro de las técnicas de optimización aproximadas, el aporte de esta tesis utiliza una estrategia poblacional basada en un cúmulo de partículas. La inteligencia de cúmulo es empleada por varios métodos que proponen la mejora de un conjunto de soluciones iniciales aleatorias representadas como individuos de una población. Estos individuos colaboran entre sí compartiendo información a fin de mejorar su propio desempeño llegando, algunos de ellos, a convertirse en buenas soluciones del problema planteado.

Dentro de estos métodos, se ha seleccionado PSO (Particle Swarm Optimizacion) y por este motivo, el capítulo 2 está totalmente dedicado a describir su implementación y particularidades. También se presentan en dicho capítulo variaciones de PSO desarrolladas en el III-LIDI que han servido de introducción para el método propuesto en esta tesis tales como:

1. El mecanismo de control de velocidad donde se logra evitar que las partículas se muevan demasiado rápido por el espacio de búsqueda.

2. El PSO de población variable donde no es necesario definir a priori la cantidad de soluciones a utilizar.

3. El PSO con control de oscilaciones para evitar que las partículas queden atrapadas en mínimos locales.

Originalmente, PSO fue desarrollado para espacios de valores continuos, pero hay una gran cantidad de problemas de optimización que involucran variables que deben tomar valores discretos. Por este motivo, fue necesario proveer una versión del algoritmo PSO que trabaje con una representación binaria de sus soluciones, denominado PSO binario.

El algoritmo PSO binario fue detallado en el capítulo 3 y se presentó una variante de dicho algoritmo que proponía un cambio en la forma de actualizar la velocidad de las partículas para mejorar la capacidad exploratoria del método.

Dada la importancia que tiene una adecuada modificación del vector velocidad, se realizaron modificaciones en el PSO binario y se propuso un nuevo método buscando resolver el problema que habitualmente presenta cuando el vector velocidad toma valores excesivamente grandes haciendo que la probabilidad de cambio de los bit de la partícula sea casi nula.

El método propuesto fue comparado con el PSO binario original y la variante mencionada anteriormente, logrando obtener mejores resultados al minimizar un conjunto de funciones con diferentes números de variables.

Los resultados obtenidos también permiten concluir que si bien el método propuesto realiza una mayor presión en la dirección del mejor que el método convencional, no pierde diversidad permitiendo llegar a alcanzar soluciones mejores con una desviación del fitness menor.

Como caso de aplicación se utilizó el método propuesto para resolver el problema de reconocimiento de rostros. Este problema fue atacado utilizando imágenes en 2D con el sólo fin de mostrar la factibilidad y utilidad de emplear una técnica de optimización en este tema.

El problema del reconocimiento de una persona a partir de la imagen de su rotro requiere disponer de un conjunto de imágenes de cada persona a reconocer almacenados en una base de datos. Luego, cada vez que se desea identificar una persona, se registra su imagen y se realiza el proceso de comparación con cada una de las imágenes almacenadas previamente en la base. Si se encuentra una imagen que supere un cierto umbral de similitud, la persona es reconocida.

Este proceso, si bien es simple de describir resulta bastante complejo de llevar a la practica.

Por ejemplo, si se espera tomar la imagen del rostro de una persona con una cámara de vigilancia y utilizar esta propuesta para determinar si debe permitírsele el acceso o no deberán tenerse en cuenta varios factores, la mayoría de ellos referidos a las caracteristicas de la nueva imagen registrada y que tienen que ver con la escala, iluminación, punto de vista, etc.

Por tal motivo, una de las principales preocupaciones reside en la representación de la imagen a reconocer. El enfoque de utilizar un conjunto de vectores que las describa es uno de los más utilizados. Nuevamente en esta dirección existen distintas soluciones

implementadas entre las cuales se ha seleccionado el método SIFT (Scale invariant feature transform). El capítulo 4 está dedicado a describir este método en especial. Al final del capítulo se ha realizado una breve mención a otras alternativas. Como allí se explica, muchas de ellas utilizan a SIFT como base y por tal motivo es el método seleccionado.

SIFT es un método general que requiere de algunas consideraciones a la hora de ser aplicado al problema de rostros. Por ejemplo, en este caso puntual resulta de interés reducir el tamaño de la base de datos y por lo tanto, es de utilidad tener alguna manera de seleccionar los vectores más representativos de cada imagen. Nótese que a la hora de seleccionar, el aspecto más importante a tener en cuenta es considerar precisamente aquellos vectores que permitan distinguir las imágenes con mayor claridad. Ese es el objetivo del método propuesto.

El método propuesto en esta tesina ha sido aplicado en la selección de los vectores SIFT más representativos de cada imagen con dos objetivos:

- Mejorar la tasa de reconocimiento a través de la reducción de falsos positivos.
- Disminuir el tamaño requerido para almacenar las imágenes en la BBDD.

El capítulo 6 está dedicado a este tema y muestra los resultados obtenidos evidenciando el buen desempeño del algoritmo propuesto.

Como trabajo futuro, resultaría interesante considerar imágenes color o en 3D. Esto implica un nuevo desafío a la hora de representar la información. Si bien existen adaptaciones de los métodos propuestos que tienen en cuenta ambos aspectos, sería deseable analizar el rendimiento de una técnica de optimización para mejorar la respuesta.

Otro aspecto necesario a la hora de realizar un reconocimiento es considerar una clase de rechazo. La solución indicada en el capítulo 5 carece de ella y si se espera transferir esta propuesta a una situación real es importante considerar su análisis e inserción.

El proceso de análisis automático que lleva a tomar una decisión final siempre será perfectible y por tal motivo es un tema de interés permanente.

Índice de figuras

Índice de tablas

Bibliografía

[1] R. Marti. Procedimientos metaheurísticos en optimización combinatoria.

[2] F. Glover. Future paths for integer programming and links to artifcial intelligence. *Computers and Operations Research*, (13):533–549, 1986.

[3] C.R. Reeves. Modern heuristic techniques for combinatorial problems. *Blackwell Scientific Publishing*, 1993.

[4] E. Alba. Parallel metaheuristics: A new class of algorithms. *John Wiley and Sons*, October 2005.

[5] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[6] F. Glover and G. Kochenberger. Handbook of metaheuristics. *Kluwer Academic Publishers*, 2002.

[7] T. Crainic and M. Toulouse. Handbook of metaheuristics. *chapter Parallel Strategies for Metaheuristics*, pages 475–513, 2003.

[8] C. Gelatt S. Kirkpatrick and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[9] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optmization*, (6):109–133, 1999.

[10] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers Oper. Res*, (24):1097–1100, 1997.

[11] T. Stützle. Local search algorithms for combinatorial problems analysis, algorithms and new applications. *Technical report, DISKI Dissertationen zur Künstliken Intelligenz.*, 1999.

[12] D. Fogel T. Bäck and Z. Michalewicz. Handbook of evolutionary computation. *IOP Publishing and Oxford University Press*, Feb 1997.

[13] J. Owens L. Fogel and M. Walsh. Artificial intelligence through simulated evolution. 1966.

[14] I. Rechenberg. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. *Fromman-Holzboog Verlag*, 1973.

[15] J. Holland. Adaptation in natural and artificial systems. *The MIT Press*, 1975.

[16] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, (8):156–166, 1977.

[17] M. Dorigo. Optimization, learning and natural algorithms. *PhD thesis, Dipartamento di Elettronica*, 1992.

[18] J. Kennedy and R. Eberhart. Particle swarm optimization. *IEEE International Conference on Neural Networks*, IV:1942–1948, 1995.

[19] J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2001.

[20] R.Poli. An analysis of publications on particle swarm optimisation applications. Technical Report CSM-469, Department of Computer Science, University of Essex, UK, 2007.

[21] R.Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, pages 1–10, 2008.

[22] Shi Y. and Eberhart R. Parameter selection in particle swarm optimization. *7th International Conference on Evolutionary Programming.*, pages 591–600, 1998.

[23] Kennedy J. Clerc M. The particle swarm - explosion, stability and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation.*, 6(1):58–73, 2002.

[24] Van den Bergh F. An analysis of particle swarm optimizers. *Ph.D. dissertation. Department Computer Science. University Pretoria. South Africa.*, 2002.

[25] Esquivel S. Cagnina L. and Coello Coello C. A bi-population pso with a shake-mechanism for solving constrained numerical optimization. *IEEE Congress on Evolutionary Computation, CEC 2007*, pages 670–676.

[26] Ward C. Mohais A, Mendes R. and Postho C. Neighborhood re-structuring in particle swarm optimization. advances in artificial intelligence. *Lecture Notes in Computer Science*, 3809/2005:776–785, 2005.

[27] Atyabi A. et all. Particle swarm optimizations: A critical review. *5th International Conference on Information and Knowledge Technology*, 2007.

[28] De Giusti A. Lanzarini L., Leza V. Particle swarm optimization with variable population size. *Artificial Intelligence and Soft Computing Â– ICAISC 2008.*, 5097/2008:438–449, 2008.

[29] Vesterstrom J. Riget J. A diversity-guided particle swarm optimizer Â– the arpso. *EVALife Project Group Department of Computer Science*, 2002.

[30] Clerc M. Stagnation. Analysis in particle swarm optimisation or what happens when nothing happens. *Department of Computer Science; University of Essex. Technical Report CSM-460.*, 2006.

[31] Van den Bergh F.and Engelbrecht A. Peer E. Using neighbourhoods with the guaranteed convergence pso. *Swarm Intelligence Symposium. SIS '03.*, pages 235–242, 2003.

[32] J. Kennedy. The particle swarm: Social adaptation of knowledge. *IEEE International Conference on Evolutionary Computation*, pages 303–308, 1997.

[33] J. García Nieto. *Algoritmos basados en Cúmulos de Partículas para la resolución de problemas complejos*. PhD thesis, Universidad de Málaga, 2006.

[34] R. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. *In Proceedings of the International Congress on Evolutionary Computation*, 1:84–88, 2000.

[35] M. Meissner, M. Schmuker, and G. Schneider. Optimized particle swarm optimization (opso) and its application to artificial neural networks training. *BMC Bioinformatics 2006*, pages 7–125, 2006.

[36] Kennedy J. and Eberhart R. A discrete binary version of the particle swarm algorithm. *World Multiconference on Systemics, Cybernetics and Informatics (WMSCI)*, pages 4104–4109, 1997.

[37] Christopher K. Monson and Kevin D. Seppi. The kalman swarm: A new approach to particle motion in swarm optimization. In *Swarm Optimization, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004*, pages 140–150, 2004.

[38] A.De Giusti J. López, L. Lanzarini. Particle swarm optimization with oscillation control. *ACM Special Interest Group on Genetic and Evolutionary Computation. Springer*, pages 1751–1752, 2009.

[39] J. Sadri and Ching Y. Suen. A genetic binary particle swarm optimization model. *IEEE Congress on Evolutionary Computation*, pages 656–663, 2006.

[40] A.P.Engelbrecht G. Pampara, N. Franken. Combining particle swarm optimisation with angle modulation to solve binary problems. *IEEE Congress on Evolutionary Computation*, pages 89–96, 2005.

[41] M.Shahabadi F. Bahrami A.Marandi, F.Afshinmanesh. Boolean particle swarm optimization and its application to the design of a dual-band dual-polarized planar antenna. *IEEE Congress on Evolutionary Computation*, pages 3212–3218, 2006.

[42] Shoorehdeli M. Khanesar M., Teshnehlab M. A novel binary particle swarm optimization. *18th Mediterranean Conference on Control and Automation.*, pages 1–6, 2007.

[43] M. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, (2):179–187, February 1962.

[44] N. Kiryati. Calculating geometric properties of objects represented by fourier coefficients. *Proc. CVPR '88, Ann Arbor, Michigan, USA*, pages 641–646, 1988.

[45] C. Harris and M. Stevens. A combined corner and edge detector. *In Fourth Alvey Vision Conference*, pages 147–151, 1988.

[46] Joseph L. Mundy. Object recognition in the geometric era: a retrospective. *Division of Engineering, Brown University Providence, Rhode Island*, 2006.

[47] D.G. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision, Corfu, Greece*, pages 1150–1157, 1999.

[48] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004.

[49] Scale-space: A framework for handling image structures at multiple scales. `http://www.csc.kth.se/~tony/cern-review/cern-html/cern-html.html`, 1997.

[50] M. Brown and D.G. Lowe. Invariant features from interest point groups. *British Machine. Vision Conference, Cardiff, Wales*, pages 656–665, 2002.

[51] A. M. Romero and M. Cazorla. Local feature view clustering for 3d object recognition. *IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii*, pages 682–688, 2001.

[52] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. *Computer Vision and Pattern Recognition*, 2004.

[53] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. 2006.

[54] C. Schmid S. Lazebnik and J. Ponce. A sparse texture representation using local affine regions. *Technical Report CVR-TR-2004-01, Beckman Institute, University of Illinois*, 2004.

[55] S. Kim and I.-S. Kweon. Biologically motivated perceptual feature: Generalized robust invariant feature. *Conference on Computer Vision (ACCV-06)*, pages 305–314, 2006.

[56] Aly Mohamed. Face recognition using sift features. *CNS186 Term Project*, 2006.

[57] A. Majumdar and R. K. Ward. Discrimitative sift features for face recognition. *Canadian Conference on Electrical and Computer Engineering*, pages 27–30, 2009.

[58] R. Ramadan and R. Abdel Kader. Face recognition using particle swarm optimization-based selected features. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2(2):51–65, 2009.

[59] Face recognition homepage. `http://www.face-rec.org/databases`.