**CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA**


DIVISIÓN DE FÍSICA APLICADA,
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

---

# Un Nuevo Detector Sub-pixel Paramétrico de Esquinas Múltiples, Empleando Técnicas de Optimización Tradicionales y Cómputo Evolutivo.

---

## Tesis

que para cubrir parcialmente los requisitos necesarios para obtener el grado de Maestro en Ciencias

Presenta:

Benjamín Hernández Valencia

Diciembre del 2002.

Resumen de la tesis de Benjamín Hernández Valencia, presentada como requisito parcial, para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación. Ensenada, Baja California, México. Diciembre del 2002.

Un Nuevo Detector Sub-pixel Paramétrico de Esquinas Múltiples, Empleando Técnicas de Optimización Tradicionales y Cómputo Evolutivo.

Resumen aprobado por:

Dr. Gustavo Olague Caballero
Director de Tesis.

La detección de esquinas y bordes es una fase importante en el análisis de bajo nivel de una imagen. La detección de esquinas es útil en el empatamiento de imágenes, reconocimiento de objetos, calibración de cámaras y reconstrucción tridimensional. La detección de la posición exacta de una esquina dentro de una imagen es un proceso complejo debido a diversos factores tales como: 1) la posición y orientación relativa de la cámara con respecto al objeto, 2) la orientación interior propia a la cámara, 3) fluctuaciones en la iluminación de la escena y 4) diversos factores ópticos. Este artículo propone un detector preciso de esquinas nuevo. Dicho detector se basa en una función paramétrica que caracteriza completamente las propiedades geométricas (ángulo y posición) y físicas (amplitud y difuminado) de cada uno de los bordes que conforman una esquina. Estas funciones de borde se combinan a través de operaciones sencillas (suma, resta, división y multiplicación) a fin de producir esquinas complejas o vértices (L, Y, K, T, flecha). El modelo paramétrico resultante que representa a la esquina compleja, es ajustado a los valores de intensidad en tonos de grises de la imagen, a través de las estrategias de optimización global *Nelder y Mead*, *Recocido Simulado* y *Evolutiva* cuya función de aptitud está dado por el método de mínimos cuadrados. La eficiencia y precisión del detector ha sido comprobada sobre imágenes reales y sintéticas.

**Palabras clave**: Detector de esquinas, localización de esquina a nivel de sub-pixel, modelo paramétrico, métodos de optimización, mínimos cuadrados, cómputo evolutivo.

ABSTRACT of the Thesis of Benjamín Hernández Valencia, presented as a partial requirement for obtaining the degree of MASTER OF SCIENCE. Ensenada, Baja California, México. December 2002

A New Parametric Sub-Pixel Multi-Corner Detector, Using Classical Optimization Techniques and Evolutionary Computation.

Abstract approved by :

Dr. Gustavo Olague Caballero
Thesis Director.

Corner and edge detection is a significant stage in low-level image analysis. Corner detection is useful in image matching process, object recognition, camera calibration and 3D object reconstruction. Accurate measurement and recognition of the corner position in a digital image is a complex process, due to several factors such as: 1) relative position and orientation of the camera with respect to the object, 2) intrinsic parameters of the camera, 3) illumination fluctuation of the scene and 4) several optics factors. This work presents a new multi-corner detector based on a unit step edge function (USEF) that defines a straight line edge. USEFs are combined in order to produce complex corner structures, using simple arithmetic operators, such as additions and multiplications. With the correct combination of the USEFs, the new parametric model represents the gray level intensity variation of the sub-structure of the real digital image. The parameters of the model are fitted using classical optimization techniques such as Nelder & Mead, Simulated Annealing, and the new Evolutionary Computations approach. In order to discover the best fit parameters, in all cases, the coast function is the mean of a least square fit estimator. Examples and experimental results illustrate the quality and the efficacy of the detector.

**Key words:** Corner detector, sub-pixel accurate detector, parametric model, optimization techniques, least square fit estimator, evolutionary computation.

# Dedicatoria

A mi esposa, Alma

y a mi hijo Abraham,

por haberme regalado el tesoro más apreciado
que un ser humano posee,
su amor y su tiempo.

A mi madre, Silvia.

En memoria de mi padre, Baltasar.

# Agradecimientos

A Dios.

Al Dr. Gustavo Olague Caballero por el gran equipo formado y la ayuda brindada a lo largo de la realización de este trabajo.

A los miembros del Comité de Tesis por sus valiosos comentarios.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Instituto de Astronomía Campus Ensenada, Universidad Nacional Autónoma de México.

# Contenido

# Lista de Figuras

# Lista de Tablas

# Capítulo I

# Introducción

El ser humano percibe al mundo que lo rodea esencialmente a través de su sentido de la vista. Más del 99% de la información que procesa el cerebro humano proviene de su órgano sensor llamado ojo [Russ, 1995]. Es por ello, que la idea de aumentar y mejorar la percepción del mundo a partir de la visión por computadora, juega un papel importante en el desarrollo científico y tecnólogico del quehacer humano.

La visión por computadora es una rama de la inteligencia artificial, que tiene como propósito la descripción explícita y significativa de las características de los objetos físicos, captados en una imagen digital. Para lograr este objetivo, la visión por computadora requiere de una combinación de dos elementos: 1) un tratamiento de bajo nivel de una imagen, como es el reconocimiento de bordes y esquinas, remover ruidos, mejorar el contraste, etc., 2) un procesamiento de alto nivel, como es el caso de la segmentación de una imagen por diversos métodos, reconocimiento de texturas, reconocimiento de objetos, empatamiento de imágenes y detección de movimiento, entre otras.

El componente básico para un sistema de visión, es un *sistema de adquisición de imágenes*. Es importante puntualizar, que un sistema de visión no sólo se enfoca al problema de la extracción de información desde sensores visuales, sino también tiene como propósito el como debe ser extraída, representada y empleada dicha información [Faugeras, 1999]. Un sistema de adquisición de imágenes consta de los siguientes elementos [Gonzalez, 1987], ver Figura 1:

1. Un sistema digitalizador. Convierte una imagen del mundo real en una repre-

sentación numérica entendible por un computador. Existe una gran variedad de dispositivos empleados para este fin; video-cámaras, microdensitómetros, escáneres de barrido y matrices fotosensibles de estado sólido. En la actualidad, las cámaras digitales son los elementos empleados comúnmente en los sistemas de adquisición de imágenes. Las cámaras construidas a partir de matrices fotosensibles de estado sólido, llamados en inglés *"Charge Coupled Device"* (CCD) sensan las variaciones de intensidad de la luz ocurrida en la escena. Dependiendo de la tecnología de los CCD's la imagen adquirida puede ser en tonos de gris o en colores. En el desarrollo del presente trabajo, se empleará una cámara digital PULNIX 9710 que registra la escena en tonos de gris.

2. Un procesador de imágenes. Consiste de un conjunto de módulos de hardware que realizan cuatro funciones básicas: adquisición de la imagen, almacenamiento, procesamiento rápido de bajo nivel y desplegado. En la actualidad, dichos elementos están condensados en una tarjeta compatible con los ductos que maneja una computadora personal. El procesador de imagen es conocido como *tarjeta de adquisición* o *"frame grabber"* en inglés. La tarjeta de adquisición Mutech M-Vision 1500 se emplea en este desarrollo.

3. Computadora Digital. En un sistema de adquisición de imágenes, la computadora digital de propósito general se utiliza para realizar cálculos intensivos sobre arreglos de datos grandes a un costo relativamente bajo. El tipo de computadora depende del proceso o algoritmo que actúa sobre la imagen y la rapidez de respuesta que se espera.

4. Sistema de almacenamiento. Una imagen digital que consiste de $2048 \times 2048$ pixeles cuantificado en 16 bits requiere de 8 megabytes (MB) para su almacenamiento. Si pretendemos analizar, experimentar o generar un conjunto de 100 imágenes de este tipo necesitaremos un espacio de almacenamiento de 800 MB. Por lo tanto, un dispositivo adecuado de almacenamiento es requerido por el sistema de adquisición de imágenes. Generalmente se emplea el acceso aleatorio que ofrecen los disco duros.

5. Sistema de Despliegue. Se emplean monitores monocromáticos o a color como el sistema principal de despliegue. Algunas cámaras digitales incluyen una salida

directa a un monitor monocromático de televisión. Otros sistemas emplean una tarjeta de adquisición para desplegar directamente la imagen en el monitor de una computadora.

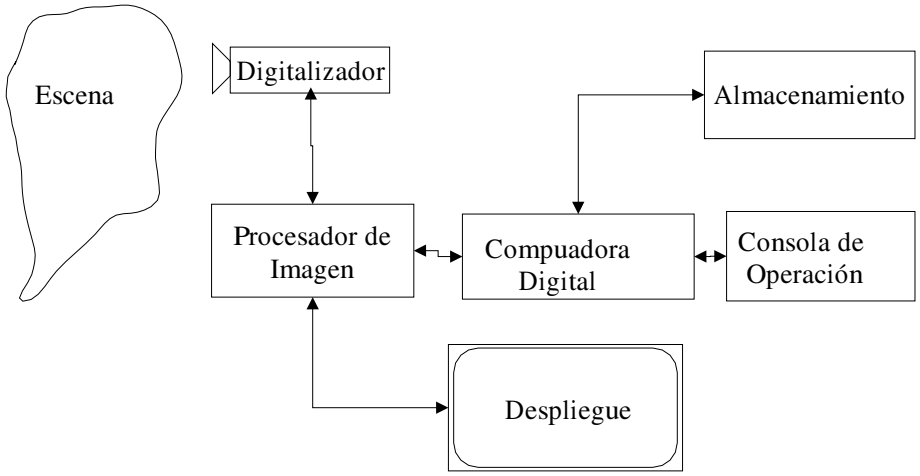



Figura 1: Elementos esenciales de un sistema de adquisición de imágenes.

Uno de los procesos fundamentales en visión por computadora, es la detección y extracción de las características de los objetos discretizados por un sistema de adquisición de imágenes. Los *bordes* y las *esquinas* observados en una imagen digital son considerados como una de las características básicas [Deriche, 1993b] [Dreschler, 1982] [Hernández, 2001] [Rohr, 1992] [Rosin, 1996]. Los bordes y las esquinas sirven como elementos de referencia, llamados *puntos de control* o *puntos prominentes*, para procesos de alto nivel. Dichos puntos de control son útiles en los siguientes procesos: 1) el empatamiento de imágenes; algoritmos de correspondencia entre dos o más imágenes, 2) reconocimiento de objetos; por ejemplo, la detección de las esquinas de un poliedro, 3) calibración de cámaras; encontrar los parámetros de las deformaciones proyectivas y factores de escala provocados por la arquitectura de la cámara, y 4) reconstrucción tridimensional; a través de un conjunto de imágenes, información de contexto y parámetros de calibración de la cámara, se puede lograr una reconstrucción tridimensional de la escena observada.

Cuando tratamos de reconocer la ubicación de un elemento de control a partir de las variaciones de la intensidad en una imagen, se tienen una serie de problemas. Nuestro principal interés es la determinación del punto de esquina. Debido a que pequeños errores

en las mediciones en el plano de la imagen provocan, en general, errores significantes en su interpretación tridimensional, la ubicación precisa de la esquina impacta directamente en la calidad de dicha interpretación. La Figura 2 muestra la transformación de un poliedro dentro del plano de la imagen. Si alejamos el plano de la imagen con respecto al objeto hasta un punto donde sea reconocida la estructura, cada uno de los vértices del poliedro tenderán a unirse en el plano de la imagen. Por tanto, la ubicación precisa de estas estructuras afecta directamente a los procesos de más alto nivel; por ejemplo a los algoritmos de calibración y reconstrucción tridimensional. En este sentido, la exactitud en la ubicación de las esquinas es una factor fundamental en el desarrollo de un sistema de visión por computadora.

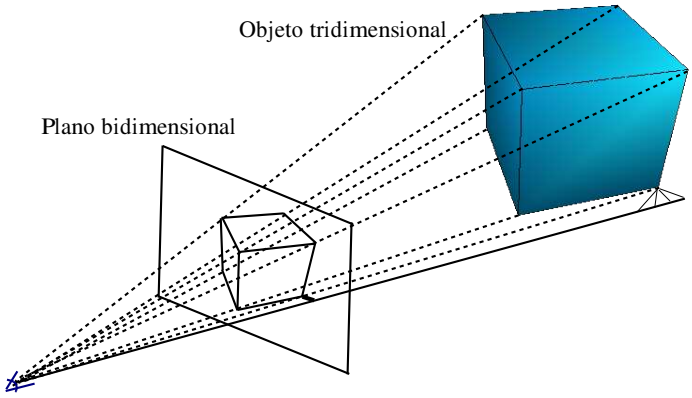



Figura 2: Errores pequeños en el plano de la imagen provocan errores significativos en la interpretación tridimensional del objeto.

Los factores fundamentales que producen incertidumbre en la la ubicación de los puntos prominentes son:

- El ruido inherente de la señal. Para poder detectar un borde se requiere de un cambio significativo de intensidad (gradiente) entre dos regiones uniformes y planas. En este sentido, un borde puede ser representado como una rampa entre ambas regiones. En una imagen podemos encontrar los casos siguientes: 1) una transición gradual entre estas dos regiones, Figura 3.a., 2) regiones de intensidad no uniforme, Figura 3.b., 3) umbral insuficiente para distinguir un borde, Figura 3.c., 4) ruido en la señal, Figura 3.d., 5) transición gradual con ruido, Figura 3.e., y 6) bordes indistinguibles, Figura 3.f.

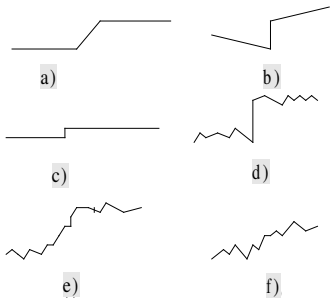



Figura 3: Tipos de bordes que pueden presentarse en una imagen digital.

- La posición y orientación de la cámara con respecto al objeto. Un sistema de adquisición de imágenes tiene como elemento de entrada un sistema digitalizador, una cámara digital generalmente. La posición y orientación de la cámara provoca que las estructuras observadas estén más o menos borrosas. Si el punto de interés está mal enfocado se produce un efecto de difuminado continuo en la imagen, imposibilitando la detección clara de los bordes y las esquinas.

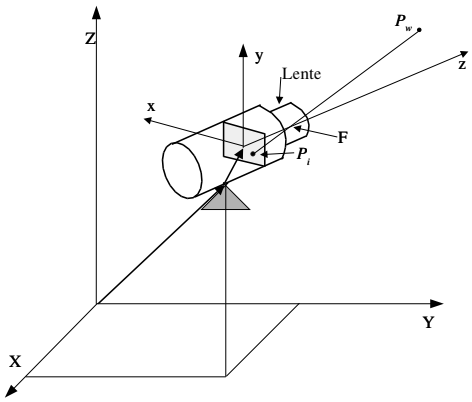



Figura 4: Modelo geométrico de una cámara empleada como un sistema de medición.

- La orientación interior propia de la cámara. Una cámara es un sistema que puede ser modelado a partir de un conjunto de transformaciones (proyectivas, Euclidianas y afines) de un punto en el espacio tridimensional del mundo real al plano de la imagen. La Figura 4, representa un modelo geométrico de una cámara empleada como un sistema de medición. El punto $P_w$ en el sistema coordenado del mundo real $(X, Y, Z)$ cruza por el eje óptico $z$ de la cámara en un punto llamado foco $F$. El plano

de la rejilla fotosensible representado por el sistema de coordenadas $(x, y)$ capta al punto $P_w$ en el punto $P_i$. El punto $P_i$ es el resultado de múltiples transformaciones geométricas entre las que se incluyen los parámetros de fabricación de la cámara. La determinación de cada uno de los parámetros involucrados en este proceso es llamado calibración de la cámara. Por tanto, mientras más certera y precisa es la ubicación de los puntos prominentes, se puede obtener una mejor calibración.

- La geometría de los elementos fotosensibles. Hemos observado que los CCD's de tipo rectangular producen un grado de difuminado distinto en la dirección de los dos ejes principales del plano sensor. La Figura 5, es una ampliación de una esquina rectangular adquirida con una cámara con sensores rectangulares. Obsérvese que la región de transición entre los niveles de gris que conforman los bordes de la esquina son diferentes. Se requieren más pixeles en el eje horizontal $\sigma_1$ que sobre el eje vertical $\sigma_2$.

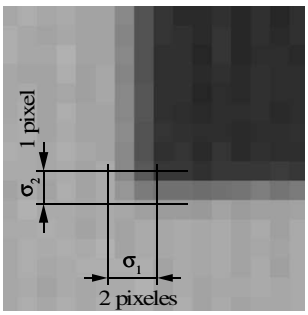



Figura 5: Los CCD's rectangulares producen difuminados distintos sobre los ejes del plano del sensor.

- Las variaciones de iluminación en una escena provocan también incertidumbre en la detección de puntos de control. Una mala iluminación causa que no puedan distinguirse con claridad las estructuras contenidas en ella. La Figura 6.a muestra una imagen con variaciones en la iluminación. Obsérvese la dificultad en la detección del perfil de los bordes que componen a la esquina, Figura 6.b.

Aunado a los problemas anteriores, los sistemas de adquisición de imágenes estan limitados en el espectro de frecuencias de las emisiones o radiaciones de la luz que pueden captar sus elementos fotosensibles. Algunos autores simplifican este concepto, reportando

que dichos sistemas están limitados en banda. En resumen, todos estos factores provocan que una imagen digital esté borrosa siempre. Más aún, estos factores de difuminado están acompañados de un desplazamiento de las estructuras de los puntos de control en el plano de la imagen, ver por ejemplo a Baker y Nayar (1996), Deriche y Giraudon (1993) y Rohr (1992).

El presente trabajo plantea un ***nuevo detector preciso de esquinas múltiples*** que intenta minimizar la problemática descrita en los párrafos anteriores. El detector preciso de esquinas múltiples reporta la ubicación del punto donde se encuentra la esquina a un nivel sub-pixel, es decir, en coordenadas con números reales. El detector opera sobre una región cuadrada de la imagen definida por el usuario. Dicho detector tiene las propiedades siguientes:

1. Se basa en una ***función paramétrica*** que caracteriza completamente las propiedades geométricas (posición y orientación) y físicas (amplitud y difuminado) de cada uno de los bordes que conforman una esquina. A esta función se le llamará ***Función Unitaria de Borde*** (FUB) [Olague, 2002].

2. La FUB se combina a través de operaciones aritméticas sencillas, como son la suma, la resta, la división y la multiplicación, a fin de producir estructuras más complejas. Por tanto, esquinas con más de dos bordes pueden ser modeladas fácilmente a partir de la combinación correcta de estos operadores aritméticos.

3. El modelo resultante de esquina tiene las siguientes características:

   - La esquina se mueve libremente en la región explorada.
   - Los ángulos de cada borde son independientes. Por tanto se pueden modelar esquinas con un ángulo de apertura en el intervalo abierto $(0^o, 180^o)$.
   - Cada uno de los bordes tiene diferentes niveles de difuminado. Esto permite modelar el fenómeno producido por detectores CCD rectangulares. Dicho fenómeno no ha sido modelado en trabajos previos.
   - Los niveles de gris dentro y fuera de la esquina se ajustan automáticamente a la región en estudio.

Como resultado del análisis de la geometría que presenta nuestro detector, hemos planteado un nuevo criterio de ubicación exacta de una esquina formada por dos bordes.

La esquina $P_e$ se localiza exactamente en el punto que representa la distancia mínima entre la curva de contorno central de nuestro modelo y el punto de intersección de las rectas de borde $P_0$, ver Figura 6.e.

El modelo de esquinas propuesto es paramétrico. Cada uno de estos parámetros se ajustan a la intensidad en tonos de gris, a partir de métodos de optimización mutidimensional para modelos no lineales. Para encontrar al mejor conjunto de parámetros que se ajusten a la imagen observada, se emplean los métodos siguientes:

1. *Levenberg-Marquardt.*
2. *Descendiendo la Colina*, conocido en inglés como *"Down Hill Simplex"*.
3. *Recocido Simulado*, conocido en inglés como *"Simulated Anneling"*.
4. *Cómputo Evolutivo*, conocido en ingés como *"Evolutionary Computation"*.

El método de Levenberg-Marquardt se utiliza para determinar el valor de la suma de la diferencia cuadrática entre nuestro modelo y la región en estudio, conocida como el estimador por mínimos cuadrados $\chi^2$. Levenberg-Marquardt representa un algoritmo que acelera la búsqueda de los mejores valores de los parámetros, si estos se encuentran lo suficientemente cercanos de un mínimo local. Por tanto, Levenberg-Marquardt es un algoritmo de optimización local. El valor de $\chi^2$ es el criterio de minimización empleado por nosotros en los algoritmos de optimización global (Descendiendo la Colina, Recocido Simulado y Cómputo Evolutivo).

La Figura 6 sintetiza el desarrollo del presente trabajo. Dada una imagen digital, Figura 6.a, en la cual queremos detectar la ubicación exacta de una esquina a partir de una región dada por el usuario, Figura 6.b, nuestro detector preciso de esquinas múltiples trabaja con la lógica siguiente:

1. Obtenemos los mejores parámetros del detector preciso de esquinas múltiples, aplicando los métodos de optimización global previamente citados, ver Figura 6.d. Como resultado de este proceso, obtenemos las características físicas y geométricas de las rectas que definen a cada borde de la esquina. La Figura 6.f., es un gráfico tridimensional de los valores de intensidad en tonos de gris de la región original, superpuesta con el modelo resultante después del proceso de optimización. Este

gráfico muestra que también han sido ajustados los parámetros del modelo a la región en estudio.

2. Calculamos el punto exacto de esquina $P_e$ a través de nuestro criterio de ubicación exacta de esquina, ver Figura 6.e.

Finalmente, el presente trabajo es relevante para procesos que requieran de gran exactitud en la detección de puntos de control. Algunos ejemplos de ello son: 1) calibración de cámaras, 2) empatamiento de imágenes, 3) reconstrucción precisa de objetos, 4) trazado de mapas urbanos a partir de fotografías aéreas.

El trabajo está organizado en los siguientes capítulos:

- *Introducción*
- *Estado del Arte.* Se introduce la definición de esquina y todos los elementos que la componen. Así mismo, se efectúa una revisión bibliográfica de los desarrollos propuestos, haciendo enfásis en los trabajos afines a nuestro modelo.
- *Modelo de Esquina.* Se plantean las ecuaciones y características del detector preciso de esquinas múltiples. Se discute nuestro criterio de ubicación exacta de esquina y se compara con otros criterios previamente desarrollados.
- *Modelado de Datos y Optimización Multidimensional.* El objetivo de este capítulo es mostrar el criterio de optimización que se utilizó en el ajuste de datos, así como explicar las técnicas de optimización multidimensional que se codificaron.
- *Experimentación y Conclusiones.* Se plantea la arquitectura con que fue creado el sistema y las tablas comparativas entre los diferentes métodos de optimización programados, aplicados a distintas imágenes.
- *Apéndice A. Características del Sistema de Adquisición de Imágenes.* Se muestran las características técnicas del sistema empleado en el desarrollo.
- *Apéndice B. Código Fuente.* Se hace referencia a la estructura de archivos de los programas incluidos en el disco compacto perteneciente a este desarrollo.

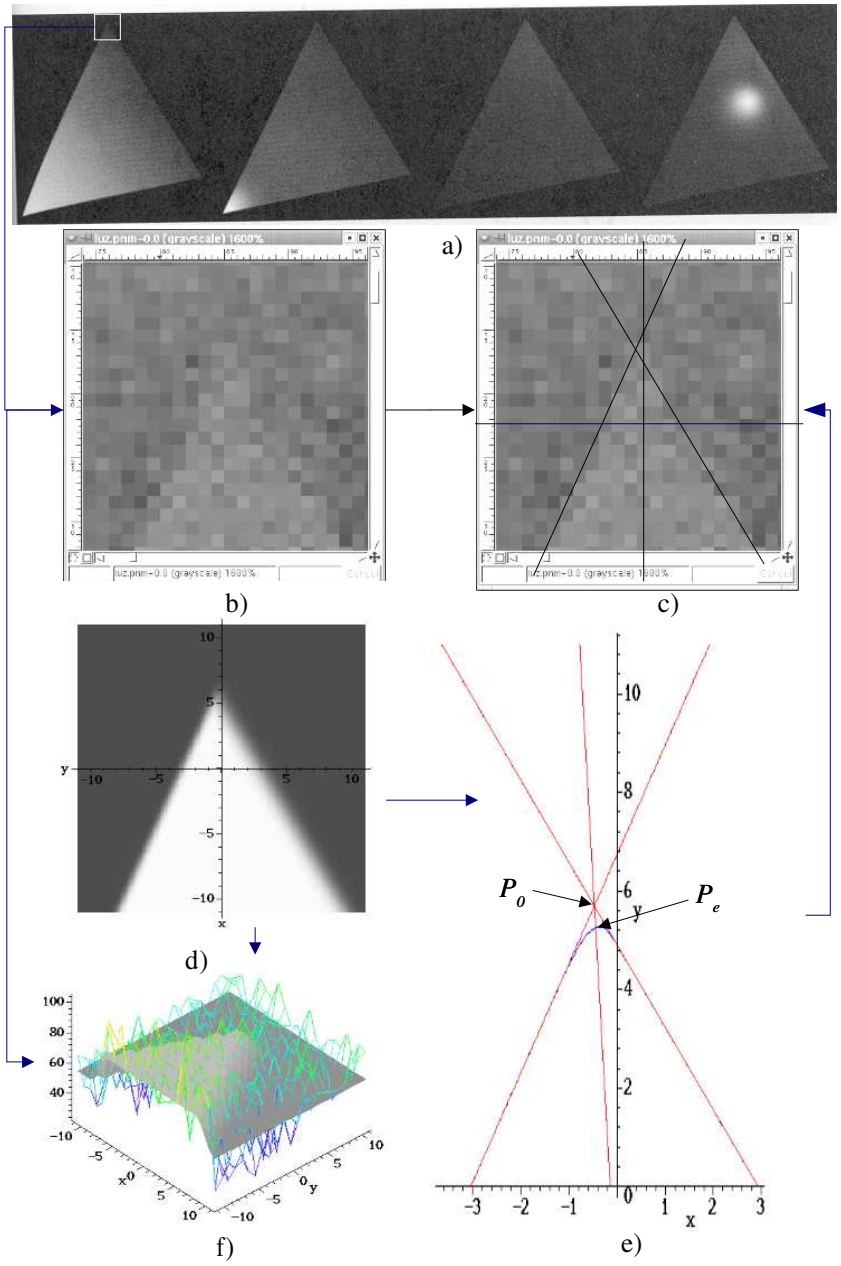



Figura 6: Ubicación del punto exacto de esquina $P_e$ en una imagen digital. $P_e$ es encontrado a través de la aplicación de nuestro modelo analítico de esquina (d,f) y nuestro criterio de ubicación exacta (e) en la región de interés (b).

# Capítulo II

# Estado del Arte

Un paradigma común en el campo de visión por computadora es la extracción de características de bajo nivel de una imagen digital, que sirvan como base para un análisis posterior de dicha imagen. Las esquinas y los bordes son consideradas como una de las características de bajo nivel principales. La localización de dichas esquinas juega un papel preponderante en problemas de visión tales como: análisis de formas y escenas, reconocimiento de objetos, detección de movimiento, empatamiento estereoscópico, reconstrucción tridimensional y calibración de cámaras [Hernández, 2001].

La detección de la ubicación precisa de una esquina es un proceso complejo, ya que intervienen diferentes factores como son: 1) la posición y orientación de la cámara con respecto al objeto, 2) la orientación interior propia de la cámara, 3) fluctuaciones en la iluminación de la escena y 4) diversos factores ópticos. Para solventar este problema, diversos autores han propuesto una gama muy amplia de métodos y modelos. En este capítulo daremos una reseña breve de los trabajos más comunes clasificados, por nosotros, en tres grandes grupos: los modelos basados en puntos de frontera, los modelos basados en propiedades geométricas y los basados en modelos paramétricos.

El capítulo está estructurado de la forma siguiente:

> *Concepto de Esquina.* Para iniciar la discusión, introduciremos el concepto de *esquina* y sus propiedades morfológicas, geométricas y físicas. En esta sección se acentuarán las definiciones de los conceptos que serán usados ampliamente en el desarrollo de nuestro trabajo.

*Métodos de Detección de Esquinas.* Se discutirán los desarrollos más importantes a nuestro criterio, los cuales son utilizados en la detección de esquinas. Adicionalmente, se tratarán más en detalle los trabajos afines a nuestro modelo, los modelos geométricos y paramétricos, con el propósito de fundamentar las comparaciones que se discutirán en los subsecuentes capítulos.

*Conclusiones.* Se plantea un resumen de los conceptos principales en lo referente a las esquinas y sus métodos de localización.

## II.1 Concepto de Esquina

En 1992 Karl Rohr discutió el concepto de esquina para imágenes en tonos de gris como sigue: si en una escena tridimensional (escena-3D) al menos dos superficies se encuentran, existirá un borde tridimensional (borde-3D). Una Esquina en una escena-3D es formada cuando al menos dos bordes-3D se unen mutuamente, en cuyo caso se estará hablando de una esquina tridimensional (esquina-3D). Análogamente, un borde bidimensional (borde-2D) se forma cuando al menos dos superficies se encuentran. Una esquina bidimensional (esquina-2D) existe si al menos dos bordes-2D se unen en una imagen digital en dos dimensiones. Por tanto, el término *esquina* es caracterizado uniformemente en dos y tres dimensiones (2D y 3D respectivamente). Las esquinas son siempre un subconjunto de los bordes.

Las esquinas denotan una característica de una imagen. Dichas esquinas contienen distintas propiedades, que agruparemos en los siguientes rubros:

1. *Morfológicas.* Se refieren a las formas producidas por los bordes que generan la esquina.

2. *Geométricas.* Se refieren a las propiedades que describen la ubicación espacial de la esquina en un sistema coordenado dado.

3. *Físicas.* Se refieren a las características producidas por los sensores en el momento de digitalizar un escenario real.

### II.1.1 Morfología de una Esquina

Las propiedades *morfológicas* se clasifican atendiendo a la forma y al número de bordes que conforman una esquina. La tipificación más común es la siguiente [Rosin, 1996]:

- *Esquina-L.* Se genera cuando dos bordes se unen en un extremo, lo que implica que existan dos zonas de gris con intensidad similar, ver Figura 7.a. La esquina-L es la estructura más simple de una esquina.

- *Esquina-T.* Se produce cuando un extremo de un borde se une en una región que no pertenece a uno de los extremos del segundo borde, ver Figura 7.b. Regularmente la intersección de los bordes forma un ángulo recto. En una esquina-T existen tres regiones con diferente intensidad en tonos de gris.

- *Esquina-Y.* También llamada "punta de flecha", se produce cuando tres bordes se unen en un extremo común, ver Figura 7.c. En este tipo de unión existen 3 zonas de gris distintas.

- *Esquina-K.* Es la unión de una esquina-L en una región que no pertenece a los extremos de un tercer borde, ver Figura 7.d. En este sentido, una esquina-T es un caso particular de una esquina-K; el ángulo de la esquina-L es de 90 grados unida al tercer borde en alguna región que no es un extremo. De forma similar se forman 3 zonas de niveles de gris distintas.

- *Esquina-X.* Se genera cuando dos bordes se unen en una región que no pertence a algún extremo de dichos bordes, ver Figura 7.e. Este tipo de estructuras puede verse también, como 4 bordes que confluyen en un mismo punto y los ángulos de los bordes no adyacentes son iguales.

- *Vértice.* Se genera cuando más de dos bordes se unen en un extremo común. En un vértice el numéro de zonas de gris diferentes es igual al número de bordes que la conforman, vea Figura 7.f. La esquina-T, la esquina-Y, la esquina-K y la esquina-X son diferentes tipos de vértices. El término vértice se emplea exactamente en este sentido en el presente trabajo. Por tanto existen sólo dos tipos de esquinas, la esquina-L y los vértices.

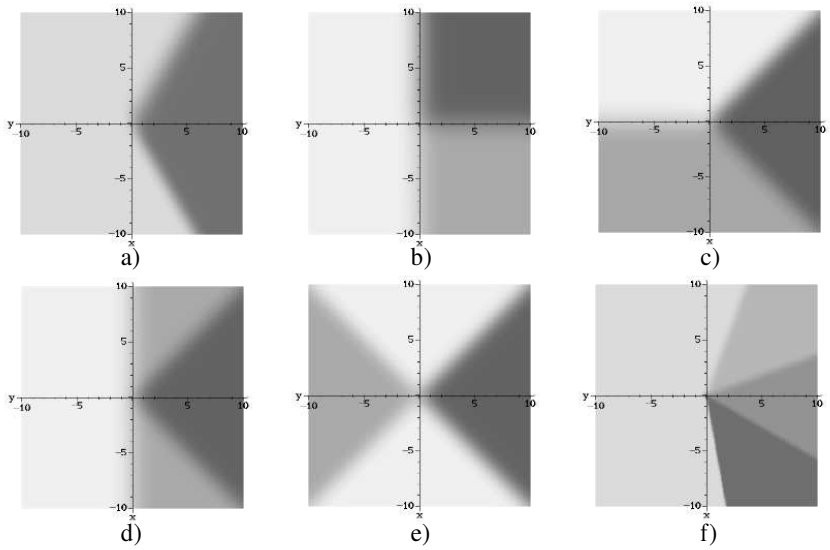



Figura 7: Clasificación de las esquinas dependiendo del tipo de unión de los bordes que las conforman. a) Esquina-L. b) Esquina-T. c) Esquina-Y. d) Esquina-K. e) Esquina-X. f) Vértice. Todas estas figuras fueron desarrolladas con base al modelo que introduciremos en este trabajo.

En nuestro trabajo nos concentraremos en la ubicación exacta del punto de esquina en una estructura del tipo *esquina-L*. Sin embargo, dicho detector tiene la habilidad de modelar estructuras más complejas como los *vértices*.

### II.1.2 Geometría de una Esquina

Las propiedades *geométricas* de una esquina, se refieren a la descripción de las rectas de los bordes que la generan y a la curvatura que puede producirse en el extremo de unión de los bordes. Es importante enfatizar que un sistema de adquisición de imágenes, es un sistema limitado en banda y produce una imagen borrosa siempre [Rohr, 1992] [Baker, 1998]. Por tanto, existe un grado de incertidumbre referente a la localización de ese punto exacto donde se ubica la esquina. Generalmente, esta incertidumbre puede verse como una curvatura en el punto de unión de los bordes. En la sección siguiente mencionaremos algunos trabajos que han sido desarrollados para tal efecto. Los atributos geométricos son:

1. *Ángulo de los bordes.* Una esquina puede verse como una estructura centrada en el origen de un sistema coordenado. Por tanto, el ángulo que forman los bordes con respecto a un eje coordenado es suficiente para caracterizar al borde en nuestro análisis, ver Figura 8.

2. *Ángulo de apertura de la esquina.* Es el ángulo $\alpha$ formado por los dos bordes, ver Figura 8.

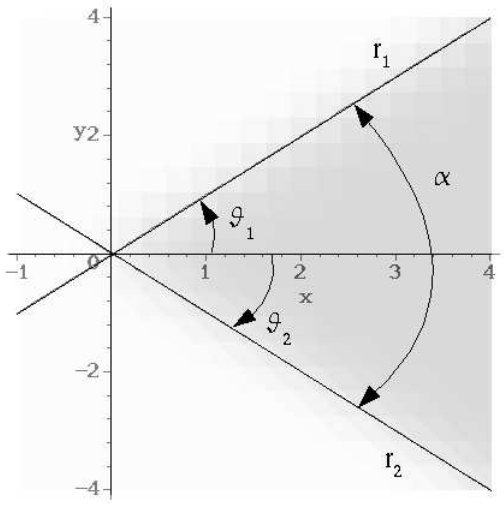



Figura 8: Geometría de los bordes de una esquina-L. $r_1$ y $r_2$ son los bordes de la esquina. $\vartheta_1$ y $\vartheta_2$ son los ángulos de $r_1$ y $r_2$ respectivamente. $\alpha$ es el ángulo de apertura de la esquina.

3. *Posición de la esquina.* Es el punto donde se localiza la esquina. Existen muchos desarrollos al respecto, tales como: las técnicas basadas en puntos de frontera, las técnicas basadas en propiedades geométricas y las técnicas basadas en modelos paramétricos. Estas técnicas se discutiran en detalle en la sección II.2. En la Figura 9 se muestran tres criterios distintos para la ubicación de la esquina. $P_1$ denota el punto de intersección de rectas de borde $r_1$ y $r_2$. Si en una esquina las dos regiones de gris de la esquina están bien definidas o describen un perfil claro, entonces la esquina pudiera ubicarse en $P_1$. Para el caso de $P_3$, la localización de la esquina es obtenida calculando la máxima curvatura planar de la estructura. Otras

aproximaciones extraen los puntos en el contorno de los bordes, y los tratan como una cadena de puntos. A partir de interpolaciones u otros métodos calculan la intersección de las rectas generadas para determinar $P_2$. En general $P_2$ se localiza en la región comprendida entre $P_1$ y $P_3$, ver [Rohr, 1992].

4. *Orientación de la esquina.* La orientación de la esquina está implícita en las coordenadas del punto de esquina. Sin embargo, algunos autores argumentan que la orientación, puede estar dada por el ángulo de la recta que bisecta el ángulo de apertura y pasa por el punto de esquina [Rosin, 1996]. En la Figura 9, $r_{bis}$ es la recta bisectriz del punto de esquina. Para este caso, la esquina mostrada es simétrica con respecto al eje $x$, por tanto $r_{bis}$ está sobre el eje $x$.

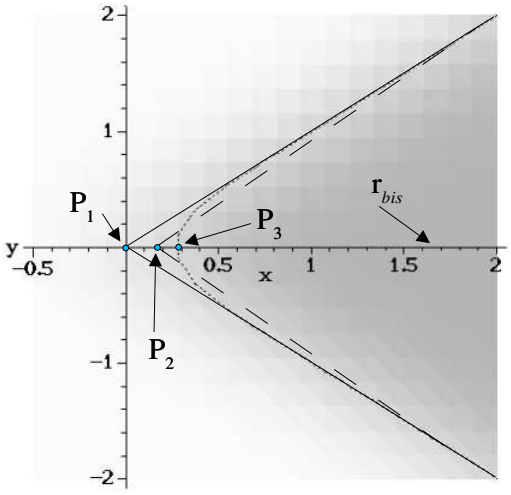



Figura 9: Ubicación del punto de esquina-L para diferentes métodos. $P_1$ es el punto de intersección de las dos rectas de borde. $P_2$ aproximación atendiendo a los puntos de frontera de los bordes. $P_3$ aproximación atendiendo a la máxima curvatura.

### II.1.3 Propiedades Físicas de una Esquina

Las características *físicas* de una esquina denotan la calidad de la esquina. Estas características cuantifican básicamente, la iluminación de una escena, la calidad del enfoque

y la distorsión provocada por la forma y el tamaño de los sensores digitales, CCD. Las características físicas son:

1. *Difuminado*. El término difuminado se refiere a que tanto, una imagen es borrosa o está empañada. Indica si el borde está bién definido, en otras palabras, si el perfil del borde puede verse con claridad. El difuminado es el factor principal que produce incertidumbre con respecto a la posición donde se debe ubicar a la esquina. Siempre y cuando pueda discernirse entre las dos zonas de intensidad que forman una esquina. Los factores que provocan este fenómeno son:

   - Enfoque. Si una sección de la escena se encuentra fuera del plano focal del sistema de adquisición de imágenes, entonces se dice que la imagen está desenfocada. Por tanto, la ubicación de la cámara con respecto al objeto juega un papel preponderante en el proceso de localización de la esquina. Este efecto provoca un difuminado continuo u homogéneo en la imagen final.
   - El tamaño finito de apertura del lente causa que la función de transferencia óptica de un sistema de adquisición de imágenes esté limitada en banda en su resolución espacial [Baker, 1998]. Este efecto es cuantificado también, como un difuminado homogéneo.
   - Las variaciones en la iluminación de la escena causan que el difuminado varíe en diferentes secciones de la imagen. Sin embargo, si la región de interés donde se ubica la esquina es relativamente pequeña, este factor se convierte en un difuminado homogéneo.
   - Una escena se discretiza típicamente empleando detectores CCD. Los CCD contienen un arreglo o matriz de elementos fotosensibles, donde cada una de estas celdas representa un pixel en una imagen digital. Dichos elementos son generalmente rectangulares, como es el caso de la cámara digital PULNIX 9701 donde cada celda fotosensible es de $11.6\mu m \times 13.6\mu m$. **Hemos encontrado que el fenómeno producido por elementos fotosensibles rectangulares, puede cuantificarse a través de dos factores diferentes de difuminado en la dirección de los dos ejes principales del plano del sensor. El modelo analítico de esquina que se presenta en este trabajo, caracteriza completamente este fenómeno**. En la literatura especializada, no hemos encontrado el modelado de dicho fenómeno.

La Figura 10.a, es la imagen de una mirilla de calibración adquirida con la cámara PULNIX 9701. El factor de difuminado $\sigma$ es la región de transición entre las dos zonas de intensidad en tonos de gris que componen a la esquina. Si aumentamos la imagen en la vecindad del pixel $(477, 135)$, Figura 10.b, se observa con claridad que los factores de difuminado $\sigma_1$ y $\sigma_2$ son distintos. Es importante enfatizar, que los problemas de enfoque, la banda limitada de los sistemas de adquisición y las variaciones de iluminación provocan un difuminado homogéneo. Por tanto, se esperaría que $\sigma_1$ y $\sigma_2$ fueran iguales, siempre y cuando sólo estos factores determinaran el grado de difuminado en una imagen. Como se observa en la Figura 10.b ésto no ocurre. Más aún, $\sigma_1$ y $\sigma_2$ están en la dirección del eje horizontal y vertical del plano del sensor respectivamente. Debido a lo anterior y que en nuestro desarrollo es importante la exactitud en la localización del punto de esquina, $\sigma_1$ y $\sigma_2$ son considerados en nuestro modelo analítico de esquina.

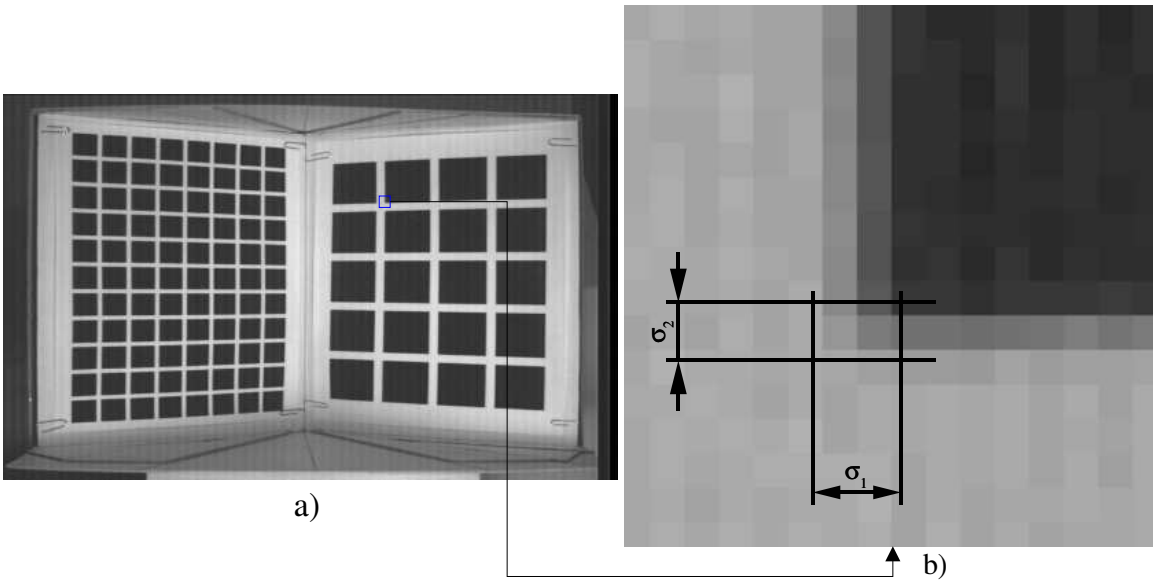



Figura 10: Difuminados $\sigma_1 \neq \sigma_2$ producidos por celdas fotosensibles rectangulares. a) Malla de calibración adquirida con una cámara PULNIX 9701. b) Zona amplificada en la vecindad del pixel $(477, 135)$.

2. *Intensidad.* En imágenes digitales en tonos de gris, los diferentes patrones de intensidad identifican a las estructuras contenidas en ellas. Para una esquina se requiere

de al menos dos zonas de gris distintas que representen a dos bordes que confluyen en una región en común. Para una esquina-L es común utilizar los términos de intensidad dentro de la esquina e intensidad fuera de la esquina. En la Figura 11, se muestra este concepto. Así mismo, la deferencia entre estas dos intensidades se le conoce como *amplitud de la esquina* [Rohr, 1992] y representa la calidad del contraste de la esquina. $I_0$ representa el fondo de la imagen e $I_1$ la cresta de la esquina.

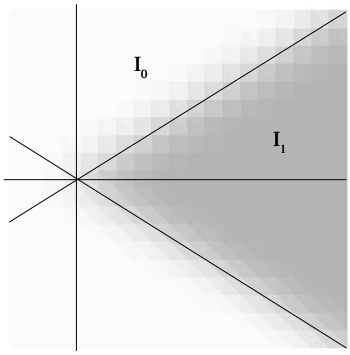



Figura 11: Identificación de las intensidades en una esquina-L. $I_0$ es la intensidad fuera de la esquina. $I_1$ es la intensidad dentro de la esquina

En conclusión, una esquina se define en función de las características geométricas y físicas de cada uno de los bordes que la forman. La morfología de la esquina nos ayuda a tipificar a las estructuras que observamos en una imagen. Como se mencionó en la parte introductoria, el problema de localización de una esquina es complejo. En la siguiente sección describiremos algunos trabajos que han sido publicados para éste fin.

## II.2 Métodos de Detección de Esquinas

Nuestra revisión bibliográfica no intenta ser exhaustiva. El propósito es ubicar a nuestro modelo en el marco contextual de los diversos trabajos que han sido propuestos en la literatura. Las técnicas de detección de esquinas pueden clasificarse en tres grandes grupos: los basados en puntos de frontera, los basados en propiedades geométricas y los basados en modelos paraméticos.

El modelo que se presenta en este trabajo, está basado en una función paramétrica que describe las propiedades físicas y geométricas de cada uno de los bordes involucrados en la creación de una esquina. Adicionalmente, dicho modelo presenta interesantes propiedades geométricas en una esquina-L. Es por ello, que se desarrollaran más en detalle los detectores basados en propiedades geométricas y los basados en modelos paramétricos.

### II.2.1 Métodos Basados en Puntos de Frontera

Los métodos basados en puntos de frontera (PF), operan sobre todos aquellos puntos que describen el contorno de un objeto. La esquina es identificada como aquel PF que tienen máxima curvatura o donde se localiza el punto de inflexión, de la estructura representada por la unión de todos los PF.

Tsai *et al.* (1999) proponen una medida de la curvatura basada en los valores propios de la matriz de covarianza de los PF. La máxima curvatura ocurre en los valores significativamente grandes de los valores propios [Tsai, 1999].

Sohn *et al.* (1998) proponen un método de suavizado de los PF para estimar la curvatura, empleando aproximaciones determinísticas basadas en el recocido simulado [1]. Los máximos de esta función de curvatura una vez aplicado el método, identifican a cada una de las esquinas.

Medioni y Yasumoto (1987) emplean B-splines para aproximarse al contorno descrito por los PF. La esquina se ubica en el máximo de la curvatura una vez calculados los coeficientes de los B-splines [Medioni, 1987].

### II.2.2 Métodos Basados en Propiedades Geométricas

Los métodos basados en propiedades geométricas operan sobre los valores de intensidad en tonos de gris en una región de la imagen. Una esquina representa entonces una superficie. La esquina es detectada a partir del cálculo de diversas propiedades geométricas de dicha superficie basados en los principios de geometría diferencial. En la literatura se les conoce como métodos directos [Rohr, 1992], debido a que trabajan directamente con los valores de intensidad de la región donde está ubicada la esquina. Estos métodos detectan el punto de esquina a un nivel de pixel (números enteros). Esta limitante, provoca que

procesos de mayor nivel, por ejemplo, la calibración de una cámara, no tenga la precisión de una detección de esquinas a nivel subpixel. Los métodos que serán discutidos son:

- Detector de Beaudet.
- Detector de Dreschler y Nagel.
- Detector de Kitchen y Rosenfeld
- Detector de Wang y Brady.

#### II.2.2.1 Detector de Beaudet

En 1978, Beaudet propone el Operador Rotacional Invariante (DET), basado en el cálculo del determinate del Hessiano de la superficie que representa una esquina [Beaudet, 1978]. La expresión propuesta por Beaudet es la siguiente:

$$K_{Beaudet} = \Delta(H) = \begin{vmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{vmatrix} \tag{II.1}$$

donde $I(x, y)$ es la función de intensidad en tonos de gris de una imagen. $\Delta(H)$ es el determinante del Hessiano de $I(x, y)$. $I_{xx}$, $I_{yy}$ e $I_{xy}$ son las segundas derivadas parciales de $I(x, y)$ con respecto a $xx$, $yy$ y $xy$ respectivamente.

La localización de una esquina consta de dos pasos:

1. Se calculan los extremos del operador rotacional invariante $K_{Beaudet}$ y se toman sus correspondientes valores absolutos.
2. Si el valor calculado en la Ecuación (II.1) sobrepasa un umbral definido por el usuario, entonces la esquina se encuentra en las coordenadas (x,y) del máximo encontrado.

La Figura 12, muestra el comportamiento tridimensional del operador rotacional invariante para el caso de una esquina-L a $90^o$ y una factor de difuminado $\sigma = 1$. Las características de este detector son:

1. El detector $K_{Beaudet}$ otorga una máximo positivo y un mínimo negativo.

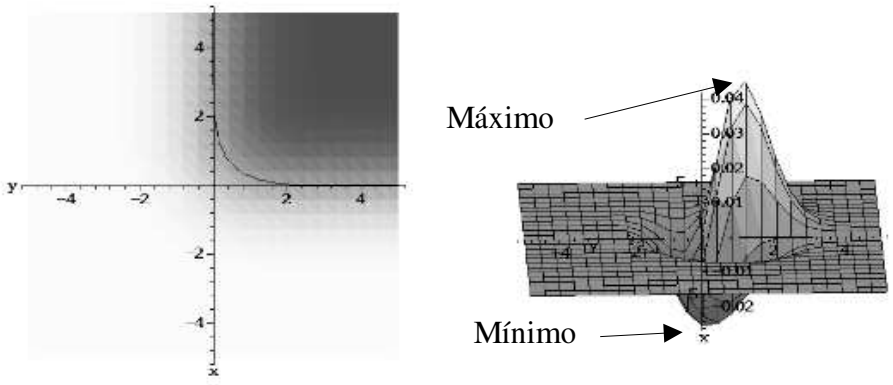



Figura 12: Detector de Beaudet. a) Esquina-L. b) Comportamiento tridimensional de $K_{Beaudet}$.

2. El mínimo negativo detecta una posición falsa del punto de esquina, vea Figura 12.b. Por tanto se debe tener cuidado en el momento de calcular los extremos.

3. El máximo positivo de $K_{Beaudet}$ determina la ubicación de la esquina. Si aplicamos el operador rotacional invariante a nuestro modelo de esquina, observamos que el máximo se encuentra desplazado hacia la cresta de la esquina, vea Figura 13.

4. El detector no es estable en el espacio de escala lineal Gaussiano. Esto es, si aplicamos distintos factores de difuminado la ubicación de la esquina se altera. Un análisis detallado de este comportamiento se encuentra en la sección III.5.

5. El detector es sensible al ruido. El operador rotacional invariante no contempla algún elemento que suavize las regiones de intensidad dentro y fuera de la esquina.

6. Se requiere definir intuitivamente un umbral para considerar la existencia de una esquina, $máximo(K_{Beaudet}) \geq Umbral$.

#### II.2.2.2 Detector de Dreschler y Nagel

En 1982, Dreschler y Nagel proponen un operador basado en la *curvatura Gaussiana principal* de una superficie. El aporte radica en aplicar el concepto de curvatura Gaussiana

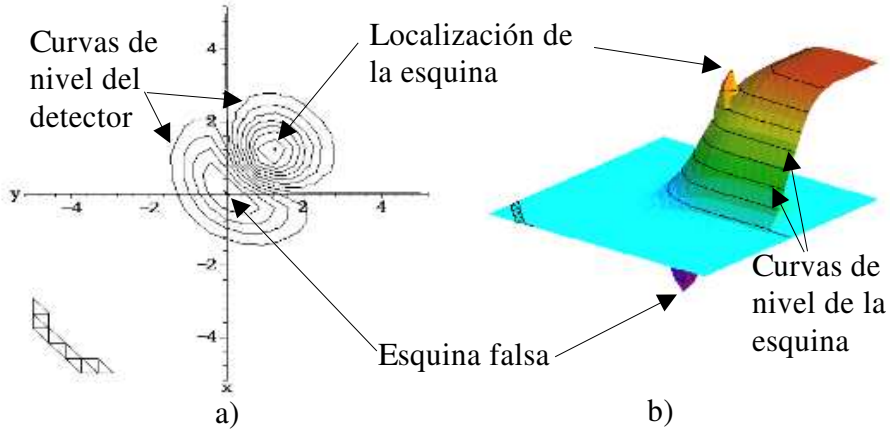



Figura 13: Ubicación del punto de esquina de $K_{Beaudet}$. a) Curvas de contorno de $K_{Beaudet}$. b) Superposición tridimensional de nuestro modelo de esquina con el operador rotacional invariante.

de una superficie (Geometría Diferencial) con el comportamiento de una esquina en una imagen. La curvatura Gaussiana principal $k_{min}k_{max}$ esta dada por:

$$K_{D\&N} = k_{min}k_{max} = \frac{\Delta(H)}{(1 + I_x^2 + I_y^2)} \tag{II.2}$$

donde $I_x$ e $I_y$ son la primeras derivadas parciales de la función de intensidad $I(x, y)$ con respecto a $x$ y $y$ respectivamente. $\Delta(H)$ es el determinate del Hessiano, equivalente al detector $K_{Beaudet}$.

El comportamiento del detector $K_{D\&N}$ es similar al propuesto por Beaudet, en su forma y propiedades. Sin embargo, el determinante del Hessiano no es la expresión exacta que denota a la curvatura Gaussiana $k_{min}k_{max}$ [Deriche, 1993b]. El punto de esquina se localiza como sigue:

1. Se calcula la curvatura Gaussiana dada por la Ecuación (II.2).

2. Se identifica el máximo positivo y el mínimo negativo de la curvatura, puntos $P_a$ y $P_b$ de la Figura 14. El punto $P_a$ representa el extremo cuando $k_{min}k_{max} > 0$. En la literatura, $P_a$ es un punto elíptico. Por otro lado, el punto $P_b$ representa el extremo cuando $k_{min}k_{max} < 0$ y se le conoce como punto hiperbólico [Deriche, 1993a].

3. El punto de esquina $P_e$ se localiza en la intersección de la línea que une a $P_a$ y $P_b$ con la curva $k_{min}k_{max} = 0$. La curva $k_{min}k_{max} = 0$ es la curvatura Gaussiana principal. La Figura 14 muestra gráficamente estos conceptos.

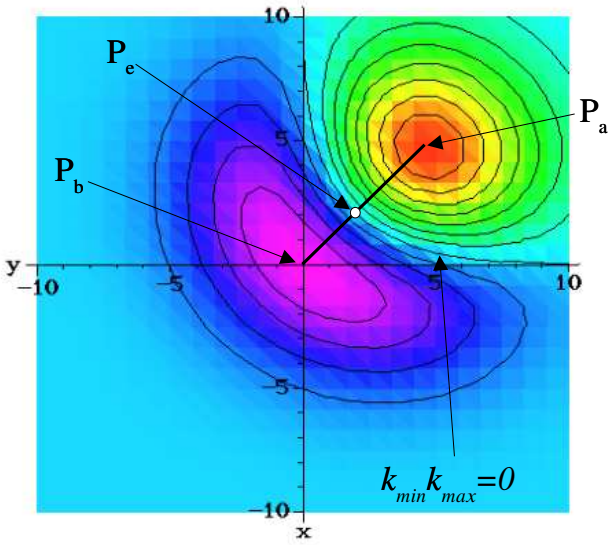



Figura 14: Ubicación del punto de esquina del detector de Dreschler y Nagel, para una esquina con un ángulo de apertura de $90^o$ y $\sigma = 1$.

Características del detector, ver [Zheng, 1999]:

- Sensible al ruido.
- Se requiere definir un umbral para discernir la existencia de una esquina.
- No es estable en el espacio de escala lineal Gaussiano.

### II.2.2.3 Detector de Kitchen y Rosenfeld

En 1982, Kitchen y Rosenfeld propusieron uno de los detectores más populares que trabajan con imágenes digitales. El detector está basado en en el cambio de la dirección del gradiente a lo largo del borde que define una esquina, multiplicado por la magnitud del gradiente [Kitchen, 1982]. La expresión que representa este detector esta dada por:

$$K_{K\&R} = \frac{I_{xx}I_y^2 + I_{yy}I_x^2 - 2I_{xy}I_yI_y}{I_x^2 + I_y^2} \tag{II.3}$$

La ubicación de la esquina se encuentra en las coordenadas (x,y) donde se localiza el extremo dado por la Ecuación (II.3), ver Figura 15. Las características del detector son:

1. Existe sólo un mínimo negativo. El punto de esquina $P_e$ corresponde al extremo de $K_{K\&R}$, ver Figura15.a.

2. El máximo del detector $K_{K\&R}$ está ubicado fuera de curva que representa la curvatura principal. La Figura 15.c muestra gráficamente este efecto.

3. No es estable en el espacio de escala lineal Gaussiano. En la sección III.5 se discute en detalle el problema de desplazamiento de $P_e$ y su sensibilidad en diferentes factores de difuminado.

4. Es sensible al ruido.

5. Se requiere definir intuitivamente un valor de umbral para discernir la existencia de una esquina.

#### II.2.2.4 Detector de Wang y Brady

En 1995, Wang y Brady proponen otra medida de curvatura $K_{W\&B}$. El valor de la curvatura $K_{W\&B}$ es proporcional a la segunda derivada tangencial en la dirección del borde e inversamente proporcional a la magnitud de su velocidad de cambio [Wang, 1995]. El detector de Wang y Brady esta dado por:

$$K_{W\&B} = \frac{\frac{\partial^2 I}{\partial t^2}}{|\nabla(I)|} \qquad para \qquad \nabla^2(I) >> 1 \tag{II.4}$$

donde $\frac{\partial^2 I}{\partial t^2} = \frac{I_{xx}I_y^2 + I_{yy}I_x^2 - 2I_{xy}I_yI_y}{|\nabla(I)|^2} = K_{K\&R}$, es la segunda derivada tangencial de $I$. $\nabla(I)$ es el gradiente de $I$. $\nabla^2(I)$ es el Laplaciano de $I$. Aplicando la supresión de respuesta falsa de esquina, la Ecuación (II.4) se reduce a la expresión siguiente:

$$K_{W\&B} = (\nabla^2(I))^2 - S|\nabla(I)|^2 \tag{II.5}$$

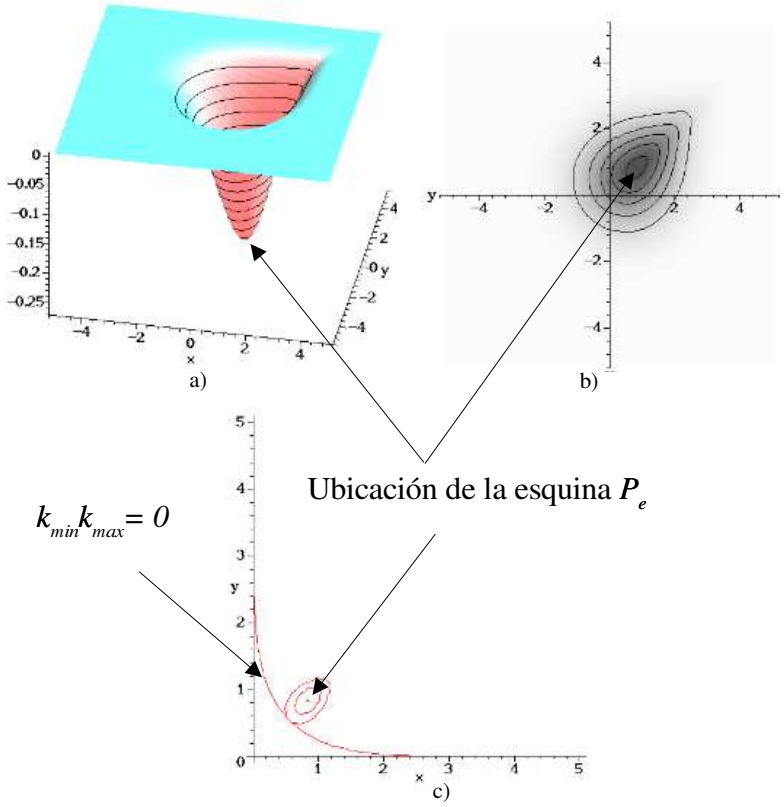



Figura 15: Ubicación del punto de esquina del detector de Kitchen y Rosenfeld, para una esquina con un ángulo de apertura de $90^o$ y $\sigma = 1$. a) Modelo tridimensional de $K_{K\&R}$. b) Curvas de contorno de $K_{K\&R}$. c) Ubicación del punto de esquina $P_e$.

donde $S$ es una constante en la supresión de respuesta falsa de una esquina. $S$ es controlada empíricamente por el usuario. La Figura 16 muestra el comportamiento tridimensional de $K_{W\&B}$.

La ubicación de la esquina se localiza calculando el máximo de la Ecuación (II.5). Una de la propiedades más importante del detector de Wang y Brady radica en su rápida ejecución, satisface los requerimientos de tiempo real, para la estimación de escenas en movimiento [Zheng, 1999]. Sin embargo, la ubicación de la esquina se encuentra desplazada en la dirección de la cresta de la esquina, como se observa en la Figura 16.b.

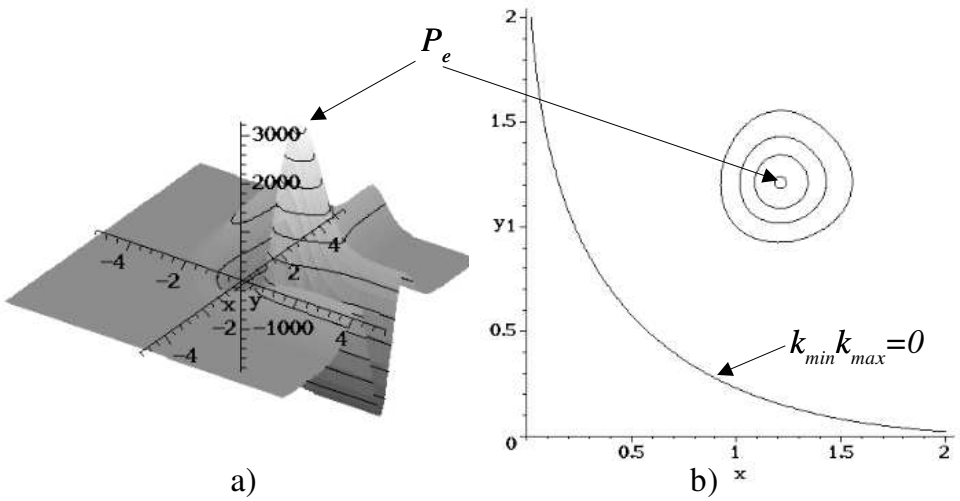



Figura 16: Detector de Wang y Brady, para una esquina con un ángulo de apertura de $90^o$ y $\sigma = 1$. a) Modelo tridimensional de $K_{W\&B}$. b) Ubicación del punto de esquina $P_e$.

### II.2.3 Métodos Basados en Modelos Paramétricos

Los detectores paramétricos se basan en modelos analíticos que representan las variaciones de intensidad captadas en una imagen digital. Los parámetros de dichos modelos describen los efectos ópticos o físicos y geométricos del objeto. Estos parámetros generalmente son:

- El ángulo de apertura de la esquina.
- Los valores de intensidad fuera y dentro de la esquina.
- La posición espacial de la esquina.
- El grado de difuminado de la esquina.

Los detectores propuestos por Rohr (1992), Deriche y Giraudon (1993), Deriche y Blaszka (1993), y Baker *et al.* (1998) son conocidos como *detectores paramétricos de características de una imagen.* Cada uno de estos detectores contienen los siguientes elementos.

1. Una función escalón unitaria que representa un borde ideal.

2. Una filtro Gaussiano o exponencial que simula el efecto de difuminado de la estructura.

3. Un proceso de convolución entre el filtro Gaussiano y el borde ideal. El proceso de convolución es necesario para incluir el efecto de difuminado en el modelo de esquina ideal.

4. Una descripción de los parámetros modelados.

5. Una región de la imagen en la vecindad donde se ubica el punto de esquina. Generalmente una región cuadrada.

6. Un método para la obtención de los mejores valores de los parámetros que se ajusten a la imagen en tonos de gris en la región en estudio. Generalmente, los parámetros se ajustan a la imagen a través de métodos de optimización multidimensionales, (descenso de gradiente, mínimos cuadrados).

La posición exacta de la esquina se incluye como un parámetro en estos desarrollos. Por tanto, la ubicación de la esquina es conocida una vez realizado el ajuste de los parámetros de los modelos a la imagen original. Dicho ajuste proporciona una primera aproximación de la ubicación exacta de la esquina. Es importante enfatizar, *que el proceso de convolución de la esquina ideal y el filtro Gaussiano provoca un desplazamiento del punto de esquina* [Rohr, 1992]. Un análisis detallado del fenómeno de desplazamiento del punto de esquina para modelos paramétricos ha sido incluido en la sección III.5.4 y III.5.5.

El primer modelo paramétrico fue propuesto por Rohr. Rohr propone un modelo basado en una función escalón convolucionada con un filtro Gaussiano. Deriche y Blaszka (1993) desarrollaron un modelo simplificado del detector de Rohr. Ellos proponen un filtro exponencial en lugar de un filtro Gaussiano y caracterizan a cada borde de la esquina en lugar del eje de simetría que emplea Rohr. Por tal motivo mostraremos en detalle el modelo paramétrico de Rohr ($M_{Rohr}$) y haremos las anotaciones pertinentes para el modelo propuesto por Deriche.

Finalmente, un trabajo más reciente fue propuesto por Baker *et al.* Estos autores desarrollan un algoritmo que construye un detector para un número arbitrario de parámetros

[Baker, 1998]. El trabajo hace un interesante énfasis en los efectos ópticos y de percepción en los sistemas de adquisición de imágenes. Cada característica es representada como un valor de densidad en un subespacio múltiple de muestreo simplificado. Las muestras del subespacio son tomadas de los valores de cada uno de los parámetros del modelo. Una característica es detectada si la proyección de los valores de intensidad alrededor del punto de interés, la esquina, está lo suficientemente cerca del subespacio de muestreo.

### II.2.3.1 Modelo Paramétrico Propuesto por Rohr

El detector de Rohr se basa en una función escalón ideal. Para el caso de una esquina-L, el eje de simetría de dicho escalón está centrado en el eje $x$. Las coordenadas del punto de esquina corresponden al origen del sistema de coordenadas (x,y) [Rohr, 1992]. El ángulo de apertura de la esquina es $\beta$ y el valor de la amplitud de esquina es $a$, ver la Figura 17. El escalón ideal es de la forma:

$$U_{Rohr}(x, y, \beta, a) = \begin{cases} a \quad si \quad x \geq 0 \quad \wedge \\ \quad |y| \leq \tan(\frac{\beta}{2})x \\ 0 \quad de \quad otro \quad modo \end{cases} \tag{II.6}$$

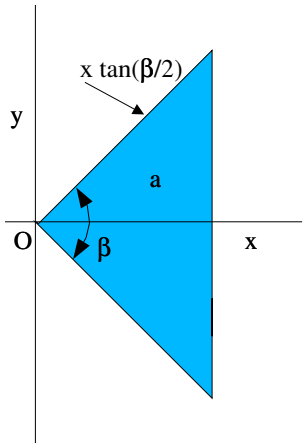



Figura 17: Esquina ideal $U_{Rohr}$.

El filtro Gaussiano simula el efecto de difuminado, dado por:

$$G(x, y) = G(x)G(y), \qquad G(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{II.7}$$

Si representamos a $\vec{x} = (x, y)$ como un sistema cartesiano general y a $\vec{\xi} = (\xi, \eta)$ como el sistema coordenado local de la esquina ideal. El modelo de esquina-L en coordenadas locales es:

$$\begin{aligned} M_{local}(\vec{x}, \beta, a) &= U_{Rohr}(\vec{x}, \beta, a) * G(\vec{x}) \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} U_{Rohr}(\vec{\xi}, \beta, a) G(\vec{x} - \vec{\xi}) d\vec{\xi} \end{aligned} \tag{II.8}$$

donde $*$ significa convolución. Rohr reporta que la integración de la Ecuación (II.8) se simplifica utilizando una función de error Gaussiana $\phi(x)$. Utilizando las siguientes expresiones:

$$\begin{aligned} t &= \tan(\beta/2) \\ \zeta_2 &= tx - y \\ \phi(x) &= \int_{-\infty}^{x} G(\xi) d\xi \\ \phi(\vec{x}) &= \phi(x)\phi(y) \\ D(\vec{x}) &= G(x)\phi(y) \end{aligned}$$

la mitad superior de la esquina, sin considerar la amplitud $a$, queda definida por

$$\begin{aligned} M(\vec{x}, \beta) &= \int_{\xi=0}^{\infty} \int_{\eta=0}^{t\xi} G(\vec{x} - \vec{\xi}) d\vec{\xi} \\ &= \phi(\vec{x}) - m(\vec{x}, \beta) \end{aligned} \tag{II.9}$$

donde

$$m(\vec{x}, \beta) = \int_{-\infty}^{x} D(\xi, t\xi - \zeta_2) d\xi$$

El modelo de esquina-L $M_{local}$ queda completo adicionando la reflexión sobre el eje de simetría. Si empleamos a $\vec{x}^{\#} = (x, -y)$ como la reflexión sobre el eje $y$, y a $\phi(-y) = 1 - \phi(y)$ como una propiedad de la función de error, el modelo de esquina-L en coordenadas locales es descrito por:

$$\begin{aligned} M_{local}(\vec{x}, \beta, a) &= a(M(\vec{x}, \beta) + M(\vec{x}^{\#}, \beta)) \\ &= a(\phi(x) - m(\vec{x}, \beta) - m(\vec{x}^{\#}, \beta) \end{aligned} \tag{II.10}$$

Finalmente, se translada el sistema de coordenadas locales mediante $\vec{x}_0 = (x_0, y_0)$, el eje de simetría $\xi$ se rota un ángulo $\alpha$ y se incluye el factor de escala $\sigma$ introducido por el filtro Gaussiano. La expresión que representa dicha transformación esta dada por:

$$\vec{x}^{*}(\alpha, \sigma) = \left( \frac{(x - x_0)\cos(\alpha) + (y - y_0)\mathrm{sen}(\alpha)}{\sigma}, \frac{-(x - x_0)\mathrm{sen}(\alpha) + (y - y_0)\cos(\alpha)}{\sigma} \right)$$

El modelo de esquina-L propuesto por Rohr es expresado por:

$$M_{Rohr}(\vec{x}, \vec{x}_0, \alpha, \beta, a, \sigma) = M_{local}(\vec{x}^*(\alpha, \sigma), \beta, a) \tag{II.11}$$

La Figura 18 muestra la ubicación de los siguientes elementos:

$(x_0, y_0)$ Ubicación del punto de esquina.

$\beta$ Ángulo de apertura de la esquina.

$\xi$ Eje de simetría de la esquina.

$\eta$ Eje normal a $\xi$.

$\alpha$ Ángulo de rotación del eje de simetría $\xi$.

$a$ Amplitud de la esquina.

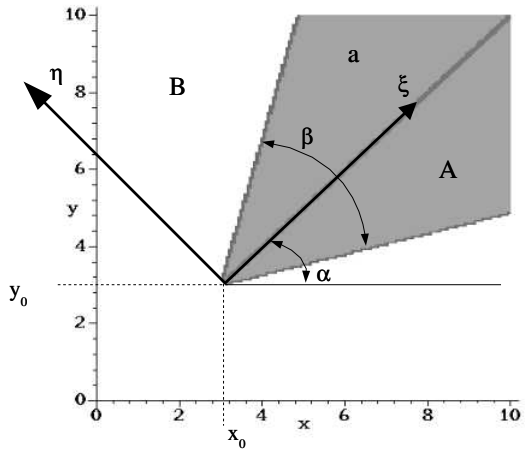



Figura 18: Modelo general de esquina-L $M_{Rohr}$.

El método de descenso de gradiente propuesto por Powell (1964) y el método de Levenberg-Marquardt (1963) fueron utilizados por Rohr. Dichos métodos se emplean para ajustar los 6 parámetros de su modelo $x_0, y_0, \alpha, \beta, a, \sigma$ a los valores de intensidad en tonos de gris de la estructura en estudio. En particular, nosotros empleamos también el método de Levenberg-Marquardt.

Rohr analiza el desplazamiento de la esquina-L causado por el factor de difuminado del filtro Gaussiano y propone una ecuación implicita, ver Ecuación III.22 en la página 64, para determinar dicho desplazamiento. Finalmente, concluye que la posición de la esquina-L es independiente de la altura de esquina $a$ y que tiene una relación lineal con respecto al factor de difuminado $\sigma$ y además tiene una relación no líneal con respecto al ángulo de apertura de la esquina $\beta$. Un análisis detallado del dezplazamiento de esquina-L del modelo propuesto por Rohr y nuestro modelo de esquina, se incluye en la sección III.5.5.

#### II.2.3.2 Modelo Paramétrico Propuesto por Deriche y Giraudon

Deriche y Giraudon proponen un modelo similar al de Rohr. Ambos trabajos parten de un modelo de esquina ideal convolucionado con un filtro Gaussiano. Su desarrollo inicia con una función escalón unitaria del borde ideal, ver referencias [Deriche, 1993a] y [Deriche, 1993b], representada por:

$$U(x) = \begin{cases} 1 & si \quad x \geq 0 \\ 0 & de \quad otro \quad modo \end{cases}$$

El modelo de esquina-L ideal, función escalón unitaria, con respecto a los ejes $\xi$ y $\eta$ está dado por:

$$U_{D\&G}(\xi, \eta, \beta) = U(\xi)U(t\xi - \eta) \tag{II.12}$$

Convolucionando $U_{D\&G}$ y $G(x, y)$, Deriche y Giraudon reportan la siguiente expresión que representa a su modelo general de esquina-L, ver Figura 18:

$$M_{D\&G}(x, y, x_0, y_0, \alpha, \beta, \sigma, A, B) = (A - B)M_1\left(\frac{\xi(x, y, x_0, y_0, \alpha)}{\sigma}, \frac{\eta(x, y, x_0, y_0, \alpha)}{\sigma}, \beta\right) + B \tag{II.13}$$

donde:

$$\xi(x, y, x_0, y_0, \alpha) = (x - x_0)\cos(\alpha) + (y - y_0)\text{sen}(\alpha) \tag{II.14}$$

$$\eta(x, y, x_0, y_0, \alpha) = -(x - x_0)\text{sen}(\alpha) + (y - y_0)\cos(\alpha) \tag{II.15}$$

$$M_1(\xi, \eta, \beta) = M_0(\xi, \eta, \beta) + M_0(\xi, -\eta, \beta) \tag{II.16}$$

$$M_0(\xi, \eta, \beta) = \phi(\xi)\phi(\eta) - \int_{z=-\infty}^{\xi} G(z)\phi(\eta + t(z - \xi))dz \tag{II.17}$$

$A$ y $B$ son loas valores de intensidad en la cresta y en el piso de la esquina respectivamente.

Las expresiones $M_{Rorh}$ y $M_{D\&G}$ son equivalentes. Notese como en el desarrollo que emplean estos autores existen similitudes. Tal es el caso de la definición de las ecuaciones de translación y rotación de los sistemas coordenados locales de la esquina $(\xi, \eta)$, ecuaciones II.14 y II.15 de la aproximación de Deriche, con respecto a la definición de $\vec{x}^*(\alpha, \sigma)$ propuesta por Rohr. El modelo de esquina local $M_{local}$, Ecuación (II.10), y $M_1$, Ecuación (II.16), de los trabajos de Rohr y Deriche respectivamente. En términos de modelado, Deriche incluye el nivel de intensidad dentro de la esquina $A$ y la intensidad fuera de la misma $B$, ver Figura 18. En oposición, Rohr modela la amplitud de la esquina $a$. Sin embargo, esta diferencia es mínima ya que $a = A - B$.

La diferencia fundamental entre estos dos métodos radica en la forma de localizar el desplazamiento del punto de esquina. Deriche basa su análisis en el comportamiento de su modelo y los detectores de Beaudet y Dreschler y Nagel, en el espacio de escalas lineal Gaussiano. Deriche y Giraudon reportan que la posición exacta de la esquina puede ser detectada en el cruzamiento por cero del modelo en el espacio de escalas Gaussiano, es decir, cuando el Laplaciano del modelo es igual a cero. Adicionalmente, afirman que los detectores $K_{Beaudet}$ y $K_{D\&N}$ se desplazan a lo largo de una recta bisectriz que pasa por la posición correcta del punto de esquina, en el espacio de escalas Gaussiano. A partir de estas dos observaciones, proponen un método para determinar el punto de esquina, vea sección III.5.4 para mayor información. Por otro lado, el desplazamiento de la esquina-L de Rohr, obedece a un comportamiento no lineal con respecto al ángulo de apertura $\beta$.

## II.3 Conclusiones

En este capítulo se discutieron las definiciones y propiedades que tiene una estructura llamada *esquina*. En resumen, una esquina está formada por la unión de dos o más bordes. Un borde es la unión de dos o más superficies en una escena-3D o en una imagen bidimensional. Por tanto, el término *esquina*, es caracterizado uniformemente en 2D y en 3D. Las esquinas, a su vez, contienen distintas propiedades. Estas propiedades se clasifican en tres grupos: 1) morfológicas; forma de unión de los bordes, 2) geométricas; geometría que describen las estructuras y 3) físicas; elementos de la calidad de la imagen adquirida.

Adicionalmente, se planteó que el factor de difuminado en una imagen es la propiedad principal que induce cierto grado de incertidumbre, en relación a la ubicación exacta del punto de esquina. El efecto de difuminado es causado por:

1. El enfoque.

2. Las variaciones en la iluminación de la escena.

3. La limitación en el ancho de banda de los sistemas de adquisición de imágenes.

4. Las dimensiones de los elementos fotosensibles de los detectores, empleados en los sistemas de adquisición, generalmente detectores CCD. **En particular, hemos encontrado que el fenómeno producido por elementos fotosensibles rectangulares, puede cuantificarse a través de dos diferentes factores de difuminado en la dirección de los dos ejes principales del plano del sensor**. Dicho fenómeno no ha sido tomado en cuenta en trabajos previos. El modelo análitico de esquina que se presenta en este trabajo, caracteriza completamente a este fenómeno.

En la literatura especializada existe una gran variedad de desarrollos propuestos para este fin. Dichos desarrollos serán clasificados en tres grupos: 1) detectores basados en punto de frontera, 2) detectores basados en propiedades geométricas y 3) detectores paramétricos. El modelo de esquina propuesto en este trabajo pertenece a los *detectores paramétricos de características de imagen*. Los detectores de Beaudet, Dreschler y Nagel, Kitchen y Rosenfeld y Wang y Brady originalmente fuerón desarrollados para detectar el punto de esquina en coordenadas enteras, a nivel pixel. Sin embargo, los gráficos presentados de dichos detectores se reportan en coordenadas reales, a un nivel sub-pixel, ver por ejemplo las figuras 13 a 16. Esto es posible y válido, debido a que aplicamos las ecuaciones correspondientes de estos operadores a nuestro modelo analítico de esquina, apoyandonos en el paquete de matemática simbólica MAPLE.

Finalmente, es necesario puntualizar que una esquina-L sufre un desplazamiento provocado por el proceso de convolución de un filtro Gaussiano y un modelo de esquina ideal. Como resultado del análisis de dicho desplazamiento, se pueden observar tres comportamientos:

1. La posición donde se ubica una esquina-L es independiente del los niveles de intensidad dentro y fuera de la esquina. Esta propiedad refuerza la consistencia de los gráficos de los modelos presentados correspondientes a diversos autores.

2. La posición donde se ubica una esquina-L tiene una relación lineal con respecto al factor de difuminado $\sigma$.

3. La posición donde se ubica una esquina-L tiene una relación no lineal con respecto a su ángulo de apertura $\beta$.

# Capítulo III

# Modelo de Esquina

## III.1 Introducción

En los dos capítulos anteriores se ha planteado la problemática de detectar y localizar los puntos prominentes en una imagen digital. Así mismo, se han descrito los desarrollos propuestos para este fin por parte de algunos autores. En particular, hemos ubicado a nuestro detector preciso de esquinas múltiples, como parte de los detectores paramétricos de características de una imagen. En el presente capítulo se describen las ecuaciones que sustentan a dicho detector y sus características y se le confronta con algunas aproximaciones descritas en la literatura.

Una esquina se define como el punto de unión de dos o más bordes. El detector preciso de esquinas múltiples descrito en este capítulo, se basa en una función paramétrica que caracteriza completamente las propiedades físicas y geométricas de cada uno de los bordes que conforman una esquina. Cada una de estas funciones de borde se combinan a través de operaciones aritméticas simples (suma, resta multiplicación y división), con el propósito de producir esquinas. A partir de la combinación correcta de los operadores aritméticos se pueden modelar fácilmente esquinas-L y vértices.

La expresión analítica que empleamos para modelar a un borde se basa en una *función de distribución de probabilidad Gaussiana*, que llamaremos ***F**unción **U**nitaria de **B**orde* (FUB). EL objetivo de este capítulo es presentar las características de la FUB y su combinación a fin de representar esquinas simples y múltiples, tomando como referencia

las variaciones de intensidad de una región en una imagen digital.

El presente capítulo está organizado de la siguiente forma:

En la sección *Modelo Unitario de Borde* se describen las expresiones matemáticas que modelan a cada uno de los bordes, en la dirección de los dos ejes principales del plano de la imagen. Así mismo, se explica el significado físico de cada uno de los parámetros que componen nuestro modelo.

En la sección *Modelo de Esquina* se presenta la mecánica básica para generar una esquina-L y estructuras más complejas, a través de la combinación correcta de operadores aritméticos entre cada modelo FUB. Así mismo, se especifican las características del modelo resultante.

En la sección *Ubicación Exacta del Punto de Esquina para Esquinas tipo L* se propone un nuevo criterio sobre la ubicación precisa del punto exacto donde se ubica una esquina-L. Tomando como referencia la geometría observada en el modelo, los elementos de la geometría que se analizan son: el punto de intersección de las rectas de bordes que conforman una esquina, el ángulo de apertura de la esquina y las diferentes curvas de nivel de nuestro modelo.

Finalmente, en la sección *Discusión de Otros Criterios de Ubicación de Esquinas*, dicho modelo se confronta con algunos detectores de esquina clásicos que operan con imágenes en tonos de gris.

## III.2 Modelo Unitario de Borde

Dado un sistema coordenado de imagen $(x, y)$ y el conjunto desconocido de parámetros denotados por $\vec{P} = (p_1, p_2, ..., p_n)$, la *función unitaria de borde* (FUB) se representa como sigue:

$$U_1(x, y, \vec{P}) = \pm \frac{1}{\sigma_1 \sqrt{2\pi}} \int_0^x e^{-\frac{(t - y\tan(\vartheta_1) - \mu_1)^2}{2\sigma_1^2}} dt + \frac{1}{2} \qquad \text{(III.1)}$$

donde las coordenadas $(x, y)$ están definidas en el rango de $[-w, w]$. La expresión denotada en el numerador de la función exponencial, representa una recta centrada a lo largo del borde $r_1$ de la forma siguiente:

$$r_1 = x - y\tan(\vartheta_1) - \mu_1 \tag{III.2}$$

en la cual, el borde $r_1$ está orientado por el ángulo $\vartheta_1$ medido en el sentido de las manecillas del reloj, a partir del lado positivo del eje $y$ en el intervalo de $(-\frac{\pi}{2}, +\frac{\pi}{2})$. El punto central $\mu_1$ designa la posición $x$ de la línea $r_1$ que cruza a lo largo del eje $y$ en el intervalo $(-w, w)$. Finalmente, necesitamos considerar el factor de escala $\sigma_1$ que caracteriza la cantidad de difuminado introducido durante el proceso de discretización. $\sigma_1$ está definido en el intervalo $(0, w]$.

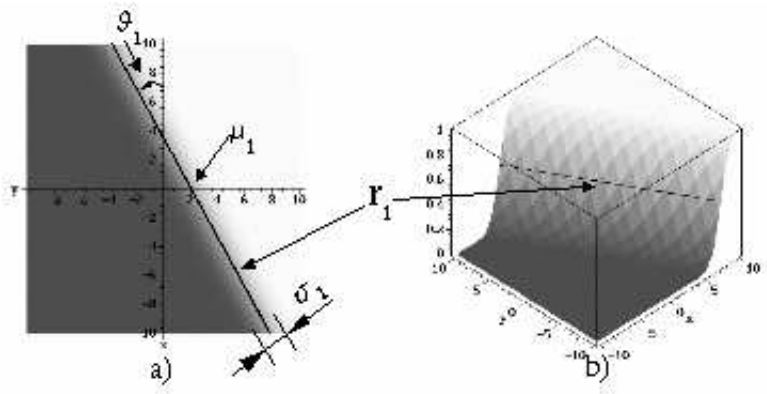



Figura 19: Función Unitaria de Borde $U_1$. a) Ubicación de los parámetros $\vec{P}$ en $U_1$. b) Vista Tridimensional de $U_1$.

La *función unitaria de borde*, Ecuación (III.1), describe una función de distribución de probabilidad Gaussiana que crece en forma uniforme desde 0 hasta 1 con respecto al eje $x$. Su gráfica tridimensional es el escalón unitario mostrado en la Figura 19.

La FUB describe completamente el comportamiento del borde en sólo una ecuación. En trabajos previos (Rohr 1992, Deriche y Giraudon 1993, Baker *et al.* 1998) el modelado de un borde requiere de un proceso de convolución entre una función escalón unitaria y un filtro Gaussiano. Estos autores emplean la función escalón unitaria para modelar a un borde ideal. El proceso de convolución intenta simular el efecto de difuminado producido durante la adquisición de la imagen. **A diferencia de éstos desarrollos, la Ecuación (III.1) modela completamente a un borde y por tanto no se requiere el proceso de convolución.**

Para obtener el modelo completo de un borde, la FUB simplemente requiere de operaciones de multiplicación y adición para escalar las variaciones de intensidad en 2D como sigue:

$$U_1'(x, y, \vec{P}) = U_1(x, y, \vec{P})A + B \tag{III.3}$$

donde $A$ representa la distancia, llamada *amplitud*, entre las dos zonas de gris inferior y superior y $B$ representa la zona de gris inferior llamada *piso* del borde, ver Figura 20.

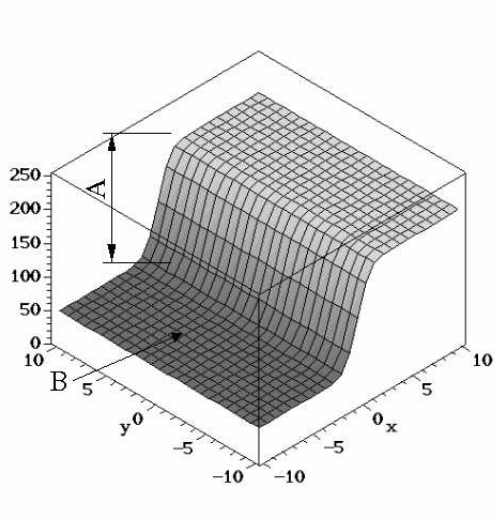



Figura 20: Función de Borde $U_1'(x, y, \vec{P})$, para $A = 150$ y $B = 50$.

Por otra parte, la función unitaria de borde $U_2(x, y, \vec{P})$ modela un borde que crece en forma uniforme con respecto al eje $y$. Los intervalos de los parámetros de $U_2(x, y, \vec{P})$ están definidos de forma similar que la Ecuación (III.1). $U_2(x, y, \vec{P})$ es representada por la expresión siguiente:

$$U_2(x, y, \vec{P}) = \pm \frac{1}{\sigma_2\sqrt{2\pi}} \int_0^y e^{-\frac{(t - x\tan(\vartheta_2) - \mu_2)^2}{2\sigma_2^2}} dt + \frac{1}{2} \tag{III.4}$$

donde la expresión en el numerador de la función exponencial denota la recta de borde $r_2$ de la forma:

$$r_2 = y - x\tan(\vartheta_2) - \mu_2 \tag{III.5}$$

en la cual, $\mu_2$ designa la posición $y$ de la línea que cruza a largo del eje $x$. El ángulo de rotación $\vartheta_2$ representa la dirección del borde dado por $r_2$ medido en sentido contrario de

las manecillas del reloj a partir del lado positivo del eje $x$. La ubicación de los parámetros y el comportamiento de $U_2$ se muestran en la Figura 21.

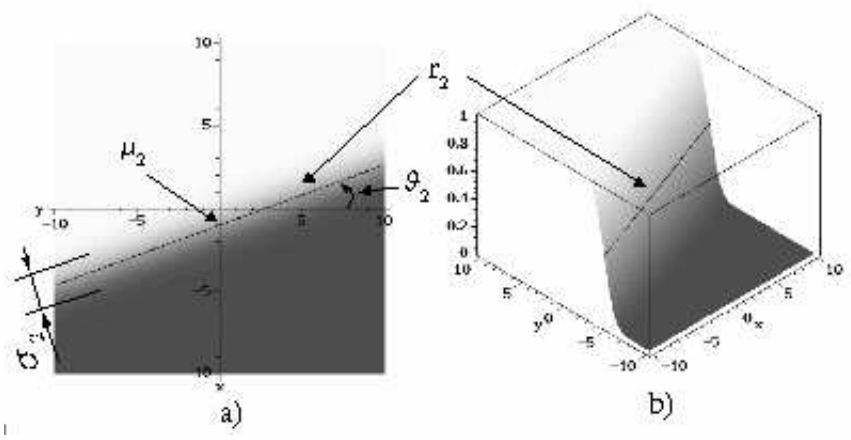



Figura 21: Función Unitaria de Borde $U_2$. a) Ubicación de los parámetros $\vec{P}$ en $U_2$. b) Vista Tridimensional de $U_2$.

Las FUB dadas por las ecuaciones (III.1) y (III.4) caracterizan a cada uno de los bordes que se generan a lo largo de los dos ejes principales del plano del sensor. Finalmente, la representación completa de un borde consta de 5 parámetros; el factor de difuminado $\sigma_1$, el ángulo $\vartheta_1$, la posición $\mu_1$, la amplitud $A$ y el piso $B$. La ecuaciones (III.1) y (III.4) pueden ser evaluadas numéricamente empleando la función de error

$$\phi(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \tag{III.6}$$

de la forma:

$$U_i(x, y, \vec{P}) = \pm \frac{1}{2} \phi \left( \frac{r_i}{\sqrt{2}\sigma_i} \right) + \frac{1}{2} \qquad para \quad i = 1, 2 \tag{III.7}$$

## III.3 Modelo de Esquina

El presente trabajo propone un nuevo detector de esquina basado en dos FUB. Para obtener una ***Función Unitaria de Esquina*** (FUE) simplemente multiplicamos ambas FUB de la forma siguiente, ver Figura 22:

$$M_L(x, y, \vec{P}) = U_1(x, y, \vec{P}')U_2(x, y, \vec{P}'') \tag{III.8}$$

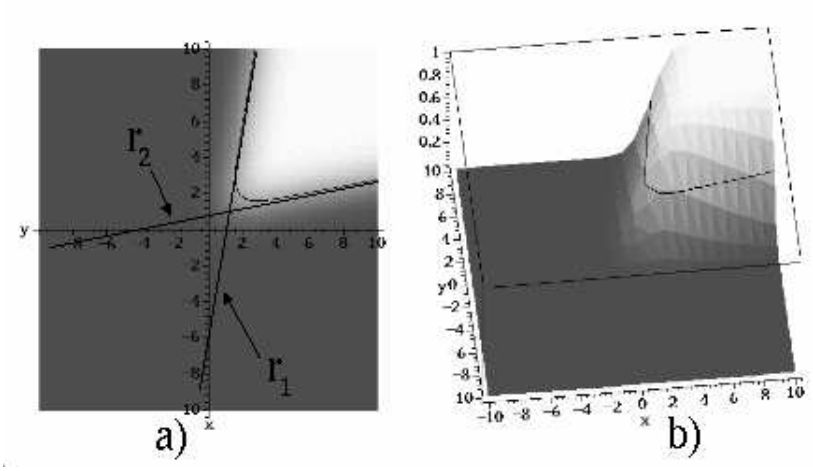



Figura 22: Función Unitaria de Esquina $M_L(x, y, \vec{P})$ construida a partir de 2 FUB. a) Muestra las dos rectas, $r_1$ y $r_2$, que modelan a cada uno de los bordes en la dirección de los ejes principales. b) Vista tridimensional de la esquina.

La estructura generada por la Ecuación (III.8) es conocida en la literatura como *"esquina-L"*. Nuestro modelo se basa en una expresión analítica con las siguientes características, las cuales no se presentan en los trabajos anteriores:

1. Cada esquina posee dos bordes los cuales tienen diferentes niveles de difuminado. Este fenómeno es producido por detectores CCD rectangulares, como es el caso de la cámara Pulnix 9701. La FUE modela el grado de difuminado, $\sigma_1$ y $\sigma_2$, en la dirección principal de cada uno de estos bordes.

2. Cada ángulo es independiente. La FUE no tiene restricción con respecto al ángulo de apertura de una esquina, por tanto, puede modelar esquinas obtusas y agudas. El ángulo de apertura está definido en el intervalo abierto de $(0, \frac{\pi}{2})$.

3. La esquina se mueve libremente en la región explorada. Nosotros obtenemos la posición y orientación de la esquina alrededor de cualquier punto dentro de la región en estudio. El tamaño de la ventana a explorar es de $(2w + 1) \times (2w + 1)$ pixeles.

4. Los símbolos $\pm$ descritos en las ecuaciones (III.1) y (III.4) nos permiten representar esquinas en las direcciones de los cuatro cuadrantes que forman el sistema coordenado de la imagen. Para modelar una esquina en la dirección del cuadrante ($x^+$,$y^+$) los signos de $U_1$ y $U_2$ deben ser + y + respectivamente. Para la dirección del cuadrante ($x^-$,$y^+$) los signos de $U_1$ y $U_2$ son $-$ y + respectivamente y así sucesivamente. Un mosaico de este efecto se muestra en la Figura 23. En una imagen real o en la aplicación de $M_L(x, y, \vec{P})$, la determinación de los signos de $U_1$ y $U_2$ se obtienen fácilmente, promediando el valor de intensidad de los pixeles que se encuentra en la parte positiva y negativa de cada uno de los ejes $x$ y $y$. El promedio mayor será el que determine el signo de $U_1$ y $U_2$. En forma específica:

   Para obtener el signo de $U_1$, si $I(x, y)$ es una función que representa los valores de intensidad de una imagen en el sistema coordenado del modelo, calculamos el promedio de la parte positiva del eje $x$,

$$px^+ = \frac{1}{w(2w+1)} \sum_{x=1}^{w} \sum_{y=-w}^{w} I(x, y)$$

   calculamos el promedio de la parte negativa del eje $x$

$$px^- = \frac{1}{w(2w+1)} \sum_{x=-w}^{-1} \sum_{y=-w}^{w} I(x, y)$$

   Si $px^+ > px^-$ el signo de $U_1$ es **positivo** (+) de otra forma es negativo ($-$).

   De forma similar, para obtener el signo de $U_2$ calculamos el promedio de la parte positiva del eje $y$,

$$py^+ = \frac{1}{w(2w+1)} \sum_{x=-w}^{w} \sum_{y=1}^{w} I(x, y)$$

   calculamos el promedio de la parte negativa del eje $y$

$$py^- = \frac{1}{w(2w+1)} \sum_{x=-w}^{w} \sum_{y=-w}^{-1} I(x, y)$$

   Si $py^+ > py^-$ el signo de $U_2$ es **positivo** (+) de otra forma es negativo ($-$).

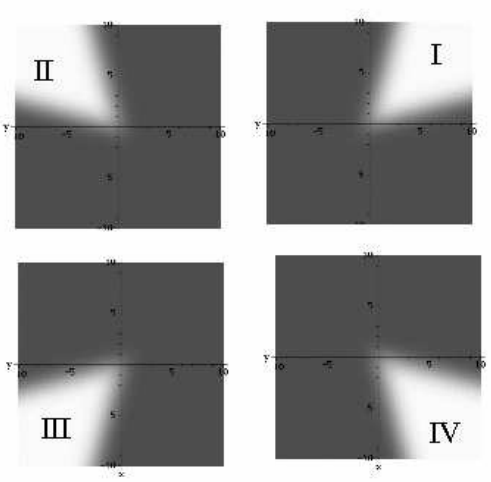



Figura 23: Esquinas con un ángulo de apertura de $60^o$, centradas en el origen $\mu_1 = \mu_2 = 0$ y un factor de difuminado $\sigma_1 = \sigma_2 = 1$ en diferentes cuadrantes. Para el cuadrante I, $U_1$ y $U_2$ tienen signo $+$. Para el cuadrante II, $U_1$ tiene signo $-$ y $U_2$ tiene signo $+$. Para el cuadrante III, $U_1$ y $U_2$ tiene signo $-$. Para el cuadrante IV, $U_1$ tiene signo $+$ y $U_2$ tiene signo $-$.

5. Los niveles de gris dentro y fuera de la esquina se ajustan automáticamente a la región de interés, gracias al método de ajuste de mínimos cuadrados utilizado, ver sección IV.3.

Considerando los valores de intensidad en tonos de gris dentro y fuera de la esquina, la FUE queda representada por:

$$M'_L(x, y, \vec{P}) = M_L(x, y, \vec{P})A + B \tag{III.9}$$

donde $A$ y $B$ son la *amplitud* y el *piso* de la esquina.

Finalmente, la Ecuación (III.8) puede ser evaluada numéricamente con facilidad empleando un producto de funciones de error como sigue:

$$M_L(x, y, \vec{P}) = \left(\pm\frac{1}{2}\phi\left(\frac{\sqrt{2}r_1}{2\sigma_1}\right) + \frac{1}{2}\right)\left(\pm\frac{1}{2}\phi\left(\frac{\sqrt{2}r_2}{2\sigma_2}\right) + \frac{1}{2}\right) \tag{III.10}$$

### III.3.1 Modelado de Esquinas Múltiples

Las estructuras complejas en tonos de gris, pueden ser modeladas a través del mismo procedimiento descrito en la sección III.2. La representación de esta estructura compleja es generada aplicando simples operaciones aritméticas (suma, resta multiplicación o división) entre las FUB. Si queremos representar 3 regiones adyacentes (esquina-T o esquina-K) el vértice es definido por:

$$V(x, y, \vec{P}) = U_1(x, y, \vec{P}')U_2(x, y, \vec{P}'')A + U_3(x, y, \vec{P}''')B + C \qquad \text{(III.11)}$$

donde $U_1$, $U_2$ y $U_3$ son tres FUB en las direcciones de cada borde. $C$ es el piso del vértice. $A$ y $B$ son las distintas amplitudes de los escalones que componen al vértice. $V(x, y, \vec{P})$ está compuesta de una esquina-L (FUE) arbitraria sumada con un borde arbitrario. Si el ángulo de cada una de las rectas ($r_1$,$r_2$ y $r_3$) es ortogonal entonces obtenemos una esquina-T. Si variamos los ángulos de la FUE entonces podemos obtener una esquina-K. Este proceso ejemplifica la facilidad con la que podemos generar nuevas estructuras partiendo de las FUB. La Figura 24, muestra un vértice generado por la Ecuación (III.11).

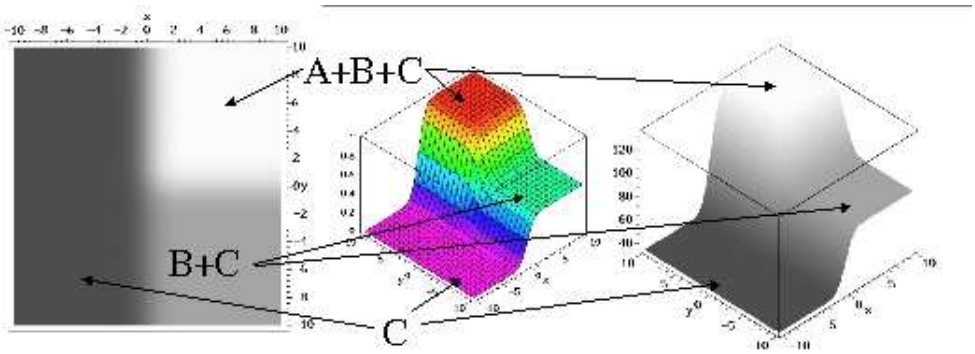



Figura 24: Vértice $T$ generado a partir de 3 FUB.

En general, el número total de parámetros necesarios para representar esquinas múltiples esta dado por la siguiente expresión:

$$n = D + 2N + E \qquad \text{(III.12)}$$

donde $D$ es número total de diferentes difuminados que queremos representar, en cada una de las direcciones principales de los bordes. $N$ es el número de funciones unitarias de borde. $E$ es el número de escalones incluyendo el piso de la esquina. $E$ se define también,

como el número de zonas con intensidad similar que se ajusten a la complejidad de la estructura.

Para el caso de una esquina-L, Ecuación (III.9), el número de difuminados es $D = 2$, el número de FUB es $N = 2$ y el número de escalones $E = 2$, entonces el número total de parámetros del modelo es $n = 2 + 2(2) + 2 = 8$. Para el caso del vértice dado por (III.11) el número total de parámetros $n = 3 + 2(3) + 3 = 12$.

## III.4 Ubicación Exacta del Punto de Esquina-L

En secciones anteriores hemos presentado el modelo que describe las características de interés. En este apartado, nos enfocaremos a determinar el punto exacto donde se ubica la esquina-L. Para lograr la localización exacta de este tipo de esquinas, consideramos el efecto de difuminado a lo largo de las dos direcciones principales de los bordes que definen a la esquina. En la literatura este efecto es llamado *"Desplazamiento de esquina-L"*, ver [Rohr, 1992] y [Deriche, 1993b].

Nuestro criterio de ubicación exacta del punto de esquina-L, se basa en la geometría que describe $M_L$. La geometría del modelo es definida por: 1) el punto de intersección de la rectas de borde, 2) el ángulo de apertura de esquina y 3) la relación que guardan las diferentes curvas de contornos de la estructura generada por el $M_L(x, y, \vec{P})$.

### III.4.1 Punto de Intersección de los Bordes

Las rectas $r_1$ y $r_2$ definidas explícitamente en la FUE, caracterizan a cada uno de los bordes que conforman una esquina. El punto de intersección ($x_0$,$y_0$) se localiza fácilmente resolviendo el siguiente sistema de ecuaciones:

$$\begin{aligned} x - y\tan(\vartheta_1) - \mu_1 &= 0 \\ y - x\tan(\vartheta_2) - \mu_2 &= 0 \end{aligned}$$

donde cada una de las ecuaciones corresponden a las rectas $r_1$ y $r_2$ definidas en $M_L(x, y, \vec{P})$. La Figura 25 muestra gráficamente la ubicación del punto de intersección de los bordes.

Resolviendo el sistema anterior, el punto ($x_0$,$y_0$) en función de los parámetros del

modelo se calcula mediante la siguiente expresión:

$$\left(x_0 = -\frac{\tan(\vartheta_1)\mu_2 + \mu_1}{\tan(\vartheta_2)\tan(\vartheta_1) - 1}, y_0 = -\frac{\tan(\vartheta_2)\mu_1 + \mu_2}{\tan(\vartheta_2)\tan(\vartheta_1) - 1}\right) \tag{III.13}$$

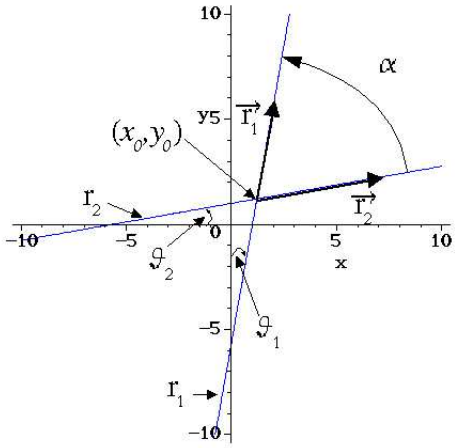



Figura 25: Punto de intersección de los bordes $(x_0,y_0)$ y el ángulo de apertura de la esquina $\alpha$.

### III.4.2 Ángulo de Apertura de la Esquina

El ángulo de apertura de la esquina $\alpha$ esta dado por el ángulo formado entre $r_1$ y $r_2$. Cada una de estas rectas pueden ser convertidas en los vectores $\vec{r_1}$ y $\vec{r_2}$ en la dirección de la esquina, ver Figura 25, tal que $\alpha$ es calculada mediante:

$$\alpha = \arccos\left(\frac{\vec{r_1} \cdot \vec{r_2}}{|\vec{r_1}||\vec{r_2}|}\right) \tag{III.14}$$

Para convertir la recta $r_1$ en el vector $\vec{r_1}$ se requieren dos puntos conocidos. El primero es el punto de intersección $(x_0,y_0)$. El segundo punto $(x_{r_1}, y_{r_1})$ se encuentra evaluando la recta $r_1$ en la coordenada $y_0 \pm 1$. Este criterio se toma debido a que $r_1$ se encuentra escrita de manera natural en (III.2) como una ecuación en función de $f(y)$. El signo $\pm 1$ nos proporciona la dirección de la esquina sobre el eje $y$. La FUB que ofrece esta

funcionalidad es $U_2$, vea sección III.3, por tanto $s_{U_2} = \pm 1$, la llamaremos *signo en la dirección y*. El punto $(x_{r_1}, y_{r_1})$ es representado por:

$$(x_{r_1} = (y_0 + s_{U_2}) \tan(\vartheta_1) + \mu_1 \quad , \quad y_{r_1} = y_0 + s_{U_2}) \tag{III.15}$$

de tal forma que las componentes del vector $\vec{r_1}$ son $(x_{r_1} - x_0, y_{r_1} - y_0)$. Aplicando la Ecuación (III.15), $\vec{r_1}$ se define como:

$$\vec{r_1} = ((y_0 + s_{U_2}) \tan(\vartheta_1) + \mu_1 - x_0 \quad , \quad s_{U_2}) \tag{III.16}$$

De forma análoga construimos el vector $\vec{r_2}$. El signo $s_{U_1} = \pm 1$ en (III.17) nos indica la dirección de la esquina con respecto al eje $x$. Las componentes de $\vec{r_2}$ son:

$$\vec{r_2} = (s_{U_1} \quad , \quad (x_0 + s_{U_1}) \tan(\vartheta_2) + \mu_2 - y_0) \tag{III.17}$$

substituyendo (III.16) y (III.17) en (III.14):

$$\alpha = \arccos\left(\frac{s_{U_1}(\tan(\vartheta_1)(y_0 + s_{U_2}) + \mu_1 - x_0) + s_{U_2}(\tan(\vartheta_2)(x_0 + s_{U_1}) + \mu_2 - y_0)}{\sqrt{(\tan(\vartheta_1)(y_0 + s_{U_2}) + \mu_1 - x_0)^2 + 1}\sqrt{(\tan(\vartheta_2)(x_0 + s_{U_1}) + \mu_2 - y_0)^2 + 1}}\right) \tag{III.18}$$

Finalmente substituyendo (III.13) en (III.18) y simplificando, el ángulo de apertura de la esquina $\alpha$ en función de los parámetros del modelo se define como:

$$\alpha = \arccos\left(s_{U_1} s_{U_2}(\sin(\vartheta_1)\cos(\vartheta_2) + \sin(\vartheta_2)\cos(\vartheta_1))\right) \tag{III.19}$$

### III.4.3 Comportamiento Geométrico del Modelo

Cada uno de las rectas que definen a los bordes ($r_1$ ó $r_2$) son asíntotas a la curva de contorno $M_L(x, y, \vec{P}) = 0.5$. Dicho comportamiento pueder observarse en la Figura 26. Recuerde, que en la sección III.2 $r_1$ y $r_2$ están centrada en nuesto modelo. En términos físicos, significa que existe igual cantidad de señal o energía hacia el piso como hacia la cresta del borde, apartir de dichas rectas. En términos geométricos, $r_1$ y $r_2$ son asíntotas a $M_L(x, y, \vec{P}) = 0.5$.

La Figura 26, muestra el comportamiento geométrico de una esquina-L con un ángulo de apertura de $90^o$, un factor de difuminado $\sigma_1 = \sigma_2 = 1$ y $r_1$ y $r_2$ siendo paralelas a los ejes $y$ y $x$ respectivamente. La Figura 26.a representa el perfil del borde $U_2$ que pertenece

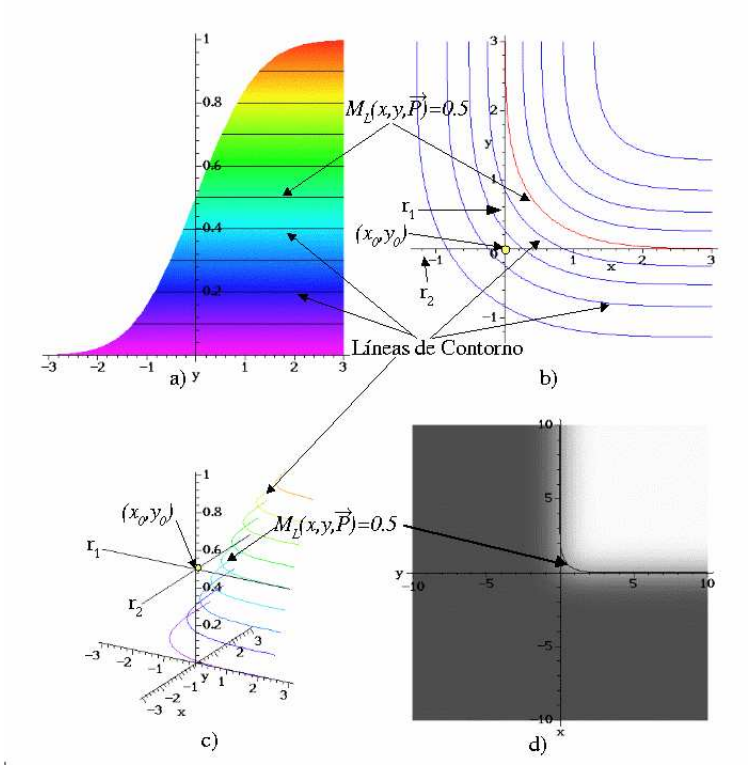



Figura 26: Geometría de $M_L(x, y, \vec{P})$. a) Corte sobre el plano $YZ$. b)Proyeccción de las curvas de contorno sobre el plano $XY$. c) Modelo tridimensional de los contornos y las rectas de cada borde. d) Esquina L en tonos de gris.

a la FUE; donde se observa que para $M_L(x, y, \vec{P}) = 0.5$ existe la misma cantidad de señal hacia la cresta como hacia el piso de la esquina. Por otro lado, en los gráficos 26.b y 26.c se observa con claridad que $r_1$ y $r_2$ son asíntotas a la curva $M_L(x, y, \vec{P}) = 0.5$.

El punto $(x_0,y_0)$ proporciona una primera aproximación de la ubicación de la esquina. En la geometría de la FUE se observa que este punto no pertenece a la esquina (Figura 26.c). Nuestra meta consiste ahora, en encontrar el punto de esquina $(x_e,y_e)$. Este punto pertenece a la línea de contorno definida por la ecuación:

$$M_L(x, y, \vec{P}) = 0.5 \tag{III.20}$$

La expresión III.20 es una ecuación trascendental implícita, donde el vector de parámetros $\vec{P}$ es conocido. Encontrar una expresión analítica que nos proporcione el valor de la

coordenada $y$ en función de $x$ que satisfaga a (III.20) no es una tarea sencilla. La simple evaluación de $M_L$ para valores aleatorios de las coordenadas $x$ y $y$ que satisfagan a (III.20) es una tarea impráctica y computacionalmente costosa. Sin embargo, nosotros podemos calcular numéricamente la ecuación implícita anterior utilizando el algoritmo de *"cotas y bisección*[1]*"* (Press *et al.*, 1998). Los puntos que alimentarán al algoritmo de cotas y bisección, son extraidos de los valores del modelo cuyas coordenadas resulten al evaluar una recta $r_s$ que pase por ($x_0$,$y_0$) y se oriente entre las dos líneas de borde $r_1$ y $r_2$. $r_s$ describe un perfil de búsqueda para el algoritmo de cotas y bisección. A este procedimiento lo llamaremos algoritmo de *"curva de contorno"*. Primeramente daremos una breve reseña del algoritmo de *"cotas y bisección"*.

El algoritmo de *"cotas y bisección"* para búsquedas de raices de una función consiste en:

- *Cotas.* Dada una función $f$ se dice que una raíz está en el intervalo ($a$,$b$) si $f(a)$ y $f(b)$ tienen signos opuestos. Si $f$ es continua en $(a, b)$, entonces existe al menos una raíz en dicho intervalo (*teorema del valor intermedio*).

- *Bisección.* Dadas las cotas ($a$,$b$) con la característica anterior, evaluamos $f$ en el punto medio del intervalo y examinamos el signo. Reemplazamos el límite que tiene el mismo signo por el valor del punto medio. Después de cada iteración las cotas que contienen a la raíz decrecen en un factor de dos.

El algoritmo de *"curva de contorno"* nos apoya en la localización de las coordenadas ($x$,$y$) que pertenecen a una curva de contorno de $M_L$. En particular nos interesa conocer la curvatura que se genera en el ángulo de apertura de esquina $\alpha$ y que satisfaga a (III.20). El algoritmo de curva de contorno contiene los pasos siguientes:

1. Transladamos la FUE de tal forma que la curva de contorno que queremos analizar este sobre el plano $\overline{XY} = 0$. Como la curva de contorno que nos interesa es $M_L(x, y, \vec{P}) = 0.5$ generamos una nueva función:

$$M_{con}(x, y, \vec{P}) = M_L(x, y, \vec{P}) - 0.5$$

[1]conocido en inglés como *"Bracketing and Bisección"*

Esta ecuación, representada en la Figura 27, nos permite utilizar la propiedad de *"cotas"* del algorimo de *"cotas y bisección"*. En otras palabras, dado un par de coordenadas ($x_a$,$y_a$) y ($x_b$,$y_b$) que representan los límites, si $M_{con}(x_a, y_a, \vec{P})$ y $M_{con}(x_b, y_b, \vec{P})$ tiene signo opuesto entonces las coordenadas que satisfacen a (III.20) están contenidas en este intervalo.

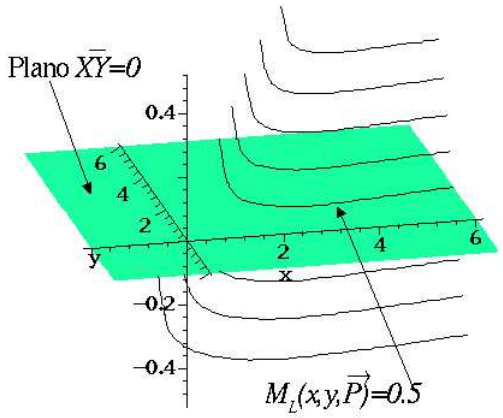



Figura 27: Curvas de contorno de $M_{con}(x, y, \vec{P})$. La curva $M_L(x, y, \vec{P}) = 0.5$ se encuentra sobre el plano $\overline{XY} = 0$.

2. Generamos una recta $r_s$ con un ángulo $\beta$ tal que $\beta$ esté en el intervalo abierto $(\vartheta_2,\ \vartheta_2 + \alpha)$ y pase por el punto ($x_0$,$y_0$). La ecuación de la recta en su forma punto-pendiente es:

$$r_s(x) = \tan(\beta)(x - x_0) + y_0$$

La expresión anterior representa una recta en la dirección del ángulo de apertura de la esquina. $r_s(x)$ permite reducir a *uno* el número de variables independientes en $M_{con}$ y ofrece una relación entre el valor de la coordenada $y$ con la coordenada $x$. La Figura 28, muestra el comportamiento tridimensional de modelo $M_{con}(x_s, r_s(x_s), \vec{P})$, para todos los valores $x_s$ que pertenecen a la recta $r_s(x)$.

3. Finalmente empleamos la propiedad de *bisección* para localizar el punto que satisface a (III.20). El procedimiento es el siguiente:

   (a) El primer límite $(x_a,y_a)$ es igual al punto de intersección de las rectas de borde $(x_0,y_0)$. Para el caso de nuestro modelo $M_{con}(x_a, y_a, \vec{P})$ es siempre negativo, Figura 28.

   (b) El segundo límite $(x_b,y_b)$ es igual a $(x_0+\lambda,r_s(x_0+\lambda)\ )$, donde $\lambda$ es un factor en la dirección de la esquina que nos proporciona el cambio de signo, vea Figura 28. En la práctica $\lambda = 2s_{U_1}(\sigma_1 + \sigma_2)$ es suficiente para producir el cambio de signo. $s_{U_1}$ es el signo de la coordenada $x$ en la dirección del crecimiento de la esquina (sección III.4.2). $\sigma_1$ y $\sigma_2$ son los factores de difuminado que pertenecen a los parámetros del modelo de esquina. El factor de difuminado tiene una relación lineal con respecto al desplazamiento de la esquina, vea sección III.5.

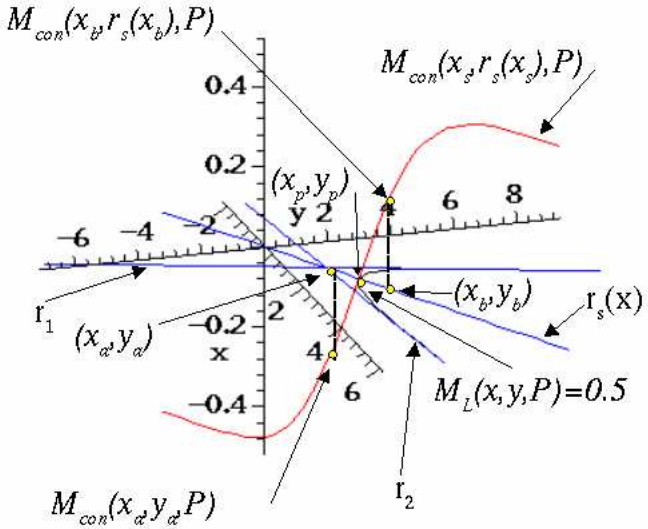



Figura 28: Ubicación del rango de búsqueda del algoritmo; *"curva de contorno"*.

(c) Evaluamos el punto medio $x_m = \frac{x_a - x_b}{2}$ en:

$$M_{con}(x_m, r_s(x_m), \vec{P})$$

(d) Reemplazamos el límite que tiene el mismo signo por el valor del punto medio y volvemos a (c) hasta encontrar el punto que corresponde al contorno deseado. En la Figura 28, el algoritmo termina cuando se encuentra una coordenada ($x_p$,$y_p$) que satisface a $M_L(x_p, y_p, \vec{P}) = 0.5$. Un punto en la línea de contorno de interés, $M_{con}(x_p, y_p, \vec{P}) = 0$ es una de sus raices y pertenece a la recta donde realizamos ls búsqueda $r_s(x_p)$.

En resumen, el algoritmo de *"cotas y bisección"*, vea la rutina **Intercepta** en la sección V.3.1, nos ayuda a localizar rápidamente las coordenadas que satisfacen a $M_L(x, y, \vec{P}) = 0.5$, sobre los valores de la recta de búsqueda $r_s$.

### III.4.4 Criterio de Ubicación Precisa de Esquina

En la sección anterior, se ha descrito el procedimiento para encontrar el conjunto de coordenadas ($x$,$y$) que satisfacen a $M_L(x, y, \vec{P}) = 0.5$. Así mismo, se ha planteado que el punto exacto de esquina ($x_e$,$y_e$) pertenece a la curva de contorno antes citada. En esta sección centraremos la atención en localizar el mejor punto a lo largo de la ecuación implícita (III.20) que defina la ubicación exacta de la esquina.

En el comportamiento geométrico de la curva que contiene al punto de esquina, con respecto al factor de difuminado $\sigma_1$ y $\sigma_2$ se observa que:

- Para valores iguales de $\sigma_1$ y $\sigma_2$ la curvatura de la esquina es simétrica con respecto a una línea que bisecta el ángulo de apertura de la esquina $\alpha$. Esto significa que el punto de intersección entre esta línea bisectriz y (III.20) representa el punto de máxima curvatura. Por tanto el punto ($x_e$,$y_e$) se ubica en dicha coordenada, ver Figura 29.b. En términos de los valores paramétricos de nuestro modelo, la ecuación de la recta bisectriz esta dada por:

$$r_{bis} = \tan(\vartheta_2 + (\frac{\alpha}{2}))(x - x_0) - (y - y_0) \qquad \text{(III.21)}$$

Los modelos paramétricos propuestos por Rohr [Rohr, 1992], Deriche y Giraudon [Deriche, 1993b] y Baker y Nayar [Baker, 1998] únicamente consideran **un** factor de difuminado en su ánalisis, lo que resulta equivalente en nuestro modelo a considerar que $\sigma_1 = \sigma_2$.

- Para valores diferentes de $\sigma_1$ y $\sigma_2$ la curvatura de la esquina *no* es simétrica a $r_{bis}$ y por tanto el punto de intersección entre la bisectriz y $M_L(x, y, \vec{P}) = 0.5$ no representa el punto de máxima curvatura. En particular:

  1. Si $\sigma_2 > \sigma_1$ el punto de máxima curvatura se ubica entre $r_{bis}$ y $r_2$, ver Figura 29.a.
  2. Si $\sigma_1 > \sigma_2$ el punto de máxima curvatura se ubica entre $r_{bis}$ y $r_1$, ver Figura 29.c.

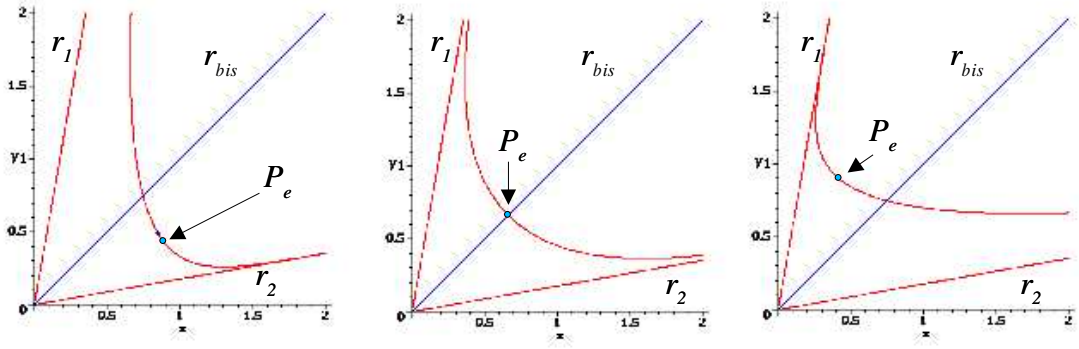



Figura 29: Distorsión de la curvatura $M_L(x, y, P) = 0.5$ causada por diferentes valores de difuminado. a) $\sigma_1 = 0.5, \sigma_2 = 4$. b) $\sigma_1 = \sigma_2 = 1.0$. c) $\sigma_1 = 4, \sigma_2 = 0.5$.

Considerando lo anterior, se propone que ***el punto exacto de esquina*** $(x_e, y_e)$ ***debiera ser el punto que represente la mínima distancia euclidiana*** $D_{min}$ ***entre el punto de intersección*** $(x_0, y_0)$ ***y las coordenadas que satisfacen*** $M_L(x, y, \vec{P}) = 0.5$. Atendiendo a este criterio, el punto de esquina $P_e$ en relación con el factor de difuminado se localiza como sigue:

1. Si $\sigma_1 < \sigma_2$, el punto $P_e$ se encuentra sobre $M_L(x, y, \vec{P}) = 0.5$ en la región comprendida entre $r_{bis}$ y $r_2$, Figura 29.a.

2. Si $\sigma_1 = \sigma_2$, el punto $P_e$ se encuentra en la intersección de $r_{bis}$ con $M_L(x, y, \vec{P}) = 0.5$, Figura 29.b.

3. Si $\sigma_1 > \sigma_2$, el punto $P_e$ se encuentra sobre $M_L(x, y, \vec{P}) = 0.5$ en la región comprendida entre $r_{bis}$ y $r_1$, Figura 29.c.

Por tanto la ubicación precisa de esquina se tranforma en un problema de optimización. La Figura 30, muestra este criterio de ubicación.

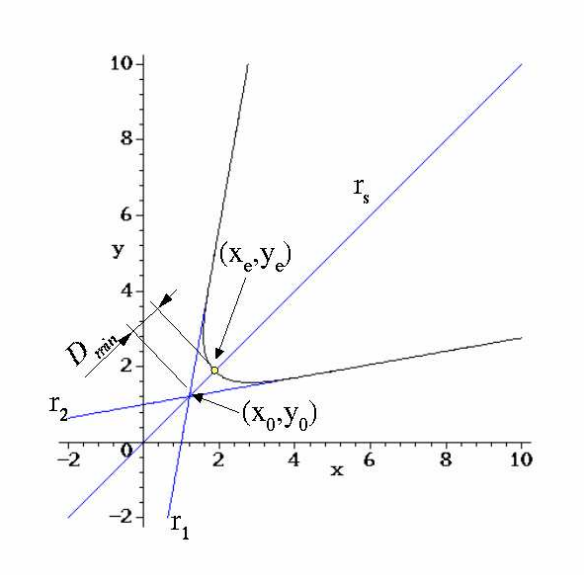



Figura 30: Criterio de ubicación exacta del punto de esquina $P_e(x_e, y_e)$.

El método de optimización de funciones multidimensionales *"Down Hill Simplex Method"* [Press, 1998], se emplea para la obtención del punto exacto de esquina $(x_e,y_e)$ que representa la mínima distancia $D_{min}$ entre $(x_0,y_0)$ y $M_L(x, y, \vec{P}) = 0.5$. Este método será descrito en detalle en el Cápitulo *Modelado de Datos y Optimización Multidimensional*, en la sección IV.4.1.1.

# III.5 Discusión de Otros Criterios de Ubicación de Esquinas

En la literatura dedicada a los detectores que trabajan en tonos de gris, la ubicación exacta de una esquina-L se define como la posición donde se encuentra el punto de máxima curvatura. En esta sección, se compará nuestro criterio de ubicación precisa de esquinas con algunos detectores clásicos que trabajan sobre imágenes en tonos de gris. Estos métodos ubican el punto de esquina basandose en distintas propiedades del campo de la geometría diferencial. Es importante enfatizar que nuestro criterio de ubicación es puramente geométrico, no se requiere calcular primeras o segundas derivadas parciales de nuestro modelo para ubicar el punto $(x_e,y_e)$.

Los primeros tres operadores que se presentan (Beaudet, Dreschler y Nagel, Kitchen y Rosenfeld) se contruyeron para detectar esquinas a nivel de pixel. Sin embargo, si aplicamos estos detectores a $M_L(x, y, \vec{P})$, podemos observar su comportamiento con respecto a nuestro criterio de ubicación.

A diferencia de los tres anteriores, los dos últimos detectores que se discutirán (Deriche y Rohr), se contruyeron para detectar esquinas a un nivel de sub-pixel y pertenecen a la categoría de *detectores paramétricos*. Estos desarrollos pertenecen a la misma categoria que el modelo que se propone en el presente trabajo.

## III.5.1 Detector Beaudet

El detector de Beaudet $K_{Beaudet}$, Ecuación II.1 página 21, se basa en el cálculo del determinate del Hessiano asociado a la intensidad de la imagen [Beaudet, 1978]. La Figura 31 muestra este detector con respecto a nuestro modelo de esquina $M_L(x, y, \vec{P})$ y la recta bisectriz $r_{bis}$, para los siguientes valores:

- Figura 31.a. Esquina con un ángulo de apertura $\alpha = 90^0$, los factores de difuminado son $\sigma_1 = \sigma_2 = 1$ y está centrada en el origen.
- Figura 31.b. Esquina con un ángulo de apertura $\alpha = 90^0$, los factores de difuminado son $\sigma_1 = 1$ y $\sigma_2 = 2$ y se encuentra centrada al origen.

- Figura 31.c. Esquina con un ángulo de apertura $\alpha = 45^0$, los factores de difuminado son $\sigma_1 = \sigma_2 = 1$ y está centrada en el origen.
- Figura 31.d. Esquina con un ángulo de apertura $\alpha = 45^0$, los factores de difuminado son $\sigma_1 = 1$ y $\sigma_2 = 2$ y está centrada en el origen.

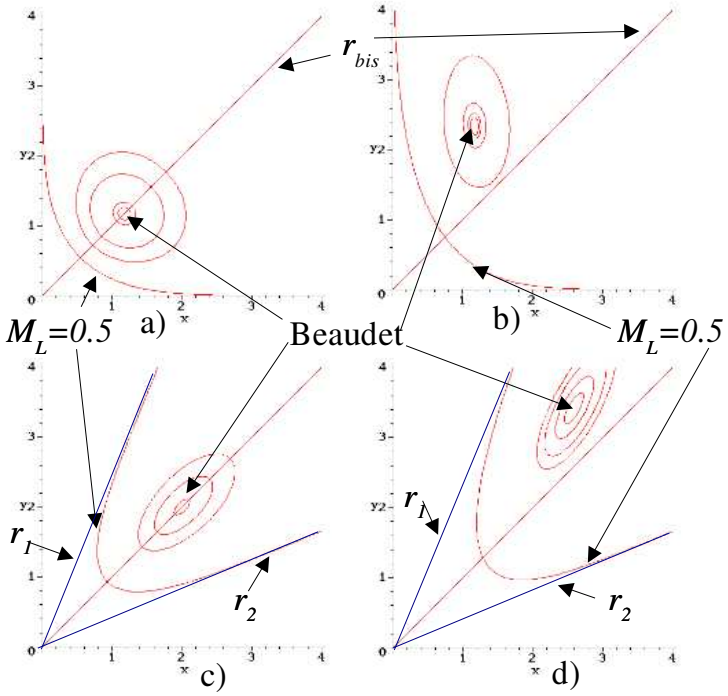



Figura 31: Ubicación exacta del punto de esquina del detector de Beaudet.

Del comportamiento de este operador descrito en la Figura 31 obervamos que:

1. El máximo del detector de Beaudet se encuentra desplazado hacia la cresta de la esquina.
2. Para valores iguales de difuminado $\sigma_1 = \sigma_2$, Figura 31.a y31.c, el máximo se localiza en la dirección de la línea bisectriz $r_{bis}$ generada por el ángulo de apertura de la esquina.
3. Para valores diferentes de $\sigma$, figuras 31.b y 31.d, el punto de esquina se espera encontrar en la región comprendida entre $r_{bis}$ y $r_2$, región donde se ubica la máxima

curvatura, ver sección III.4.4. Sin embargo, el detector de Beaudet localiza la esquina fuera de esta región y fuera de la línea $r_{bis}$. Por tanto, no es estable para distintos valores de difuminado.

### III.5.2 Detector Dreschler y Nagel

El detector propuesto por Dreschler y Nagel $K_{D\&N}$, Ecuación II.2 página 23, se basa en el cálculo de la curvatura Gaussiana [Dreschler, 1982]. Para ubicar una esquina, primeramente se requiere encontrar las coordenadas de los extremos (un máximo positivo y un mínimo negativo) de la curvatura Gaussiana. Con estos valores, el punto de intersección entre la curvatura Gaussiana principal $k_{min}k_{max} = 0$ y la recta que une a estos extremos, representa la ubicación de la esquina. En la Figura 32 donde $\sigma_1 < \sigma_2$, se observa que para valores distintos de difuminado, el detector de Dreschler y Nagel se encuentra desplazado hacia la región comprendida entre $r_{bis}$ y $r_1$. La esquina se espera localizar entre $r_{bis}$ y $r_2$, por tanto, la ubicación de la esquina del detector de Dreschler y Nagel no es el punto exacto donde se encuentra la esquina. Esta aproximación sufre la misma problematica descrita por el detector de Beaudet.

### III.5.3 Detector Kitchen y Rosenfeld

El detector de Kitchen y Rosenfeld $K_{K\&R}$ [Kitchen, 1982] localiza a la esquina en el extremo de la Ecuación II.3. Este operador se basa en el cambio del gradiente a lo largo del contorno que define la esquina escalada mediante la magnitud local del gradiente. La Figura 33, muestra el comportamiento del operador de Kitchen y Rosenfeld con respecto a la curva de contorno $M_L(x, y, \vec{P}) = 0.5$, para los casos de: esquinas con un ángulo recto, figuras 33.a y 33.b; esquinas con ángulo de $45^0$, figuras 33.c y 33.d; y esquinas para valores iguales de difuminado, figuras 33.a y 33.c; esquinas con valores distitos de difuminado ( $\sigma 1 < \sigma_2$), figuras 33.b y 33.d. De las observaciones realizadas con este operador se concluye que:

1. El punto de localización de la esquina del operador $K_{K\&R}$ está desplazado en la dirección de la cresta de esquina.

2. Para valores $\sigma_1 = \sigma_2$, la esquina se desplaza a lo largo de la recta $r_{bis}$ que bisecta el ángulo de apertura de la esquina, figuras 33.a y 33.c.

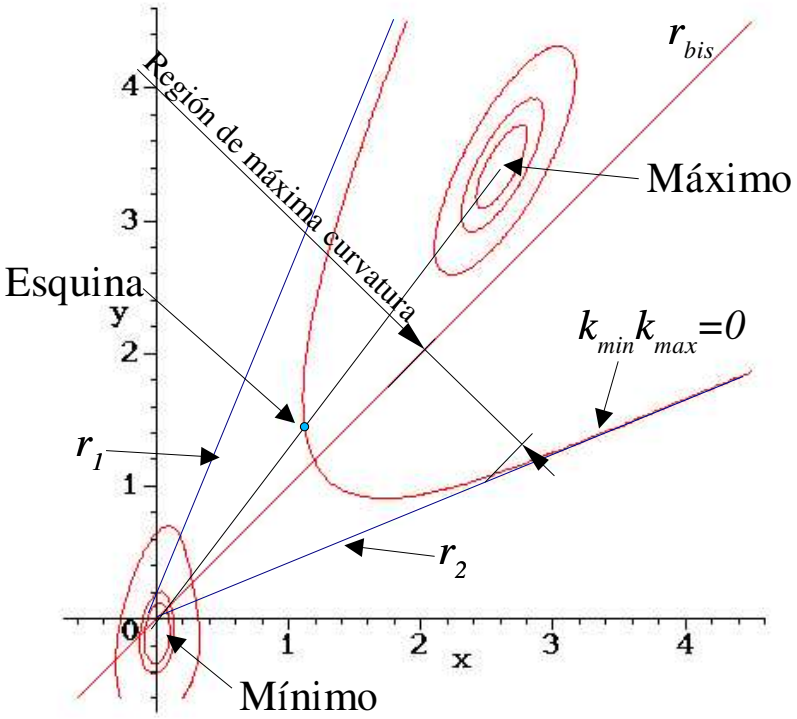



Figura 32: Ubicación exacta del punto de esquina del detector de Dreschler y Nagel.

3. Para diferentes valores de difuminado el desplazamiento de la esquina se localiza en la misma región donde se espera ubicar la esquina, figuras 33.b y 33.d. En la sección anterior se discutió que la esquina está ubicada en la región comprendida entre $r_{bis}$ y $r_2$ cuando $\sigma_1 < \sigma_2$. A diferencia de $K_{Beaudet}$ y $K_{D\&N}$, el detector de Kitchen y Rosenfeld es consistente con nuestro criterio de ubicación de esquina, referente a la región donde se espera ubicar la esquina.

### III.5.4 Detector Deriche

Deriche y Giraudon [Deriche, 1993b] realizan un estudio análitico del desplazamiento de vértices y esquinas-L, reportando las siguientes observaciones:

1. El máximo local de detectores de Beaudet y Nagel se mueve en el espacio de escala

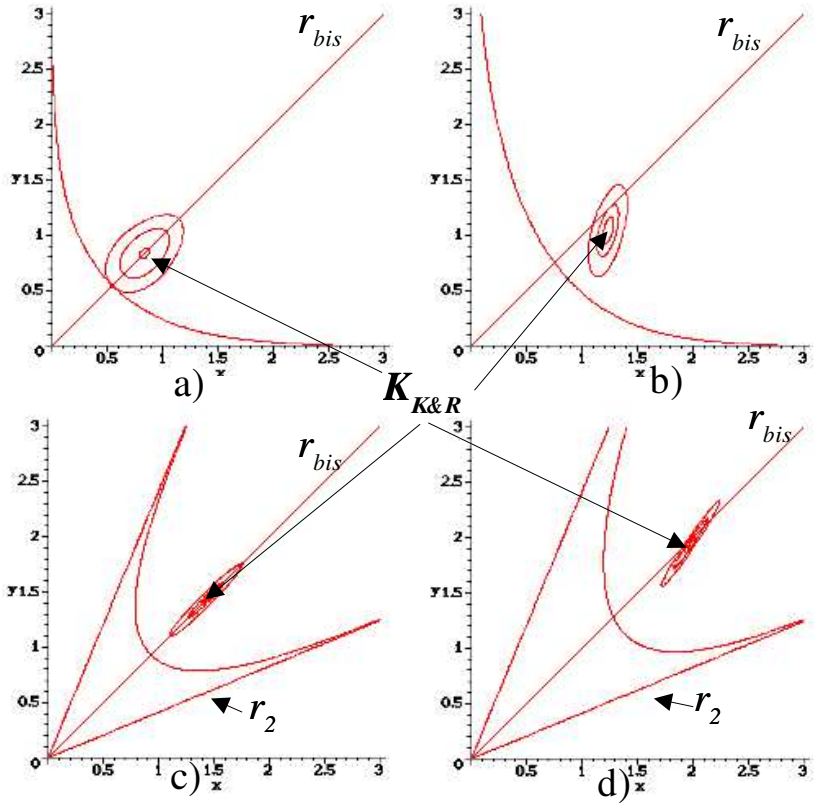



Figura 33: Ubicación del punto de esquina del detector Kitchen y Rosenfeld ($K_{K\&R}$).

Gaussiano, a lo largo de una línea bisectriz que pasa a través de la posición exacta de la esquina (Figura 34).

2. La posición exacta de la esquina puede ser detectada como un cruzamiento por cero estable en el espacio de escala Gaussiano. En específico cuando el Laplaciano de la imagen es cero en el punto de cruzamiento por cero.

Combinando estas dos observaciones su algoritmo de detección es como sigue. Primero, se calcula el Laplaciano de la imagen. Segundo, se calcula el operador de Beaudet para dos escalas diferentes en el espacio de escalas Gaussiano, y se detecta un extremo local para cada escala. Tercero, se marcan estos extremos en la imagen original y se genera una recta que une a estos dos puntos. Finalmente, el punto se encuentra cuando el Laplaciano de la imagen evaluado en la recta que une ambos extremos, cruza por cero.

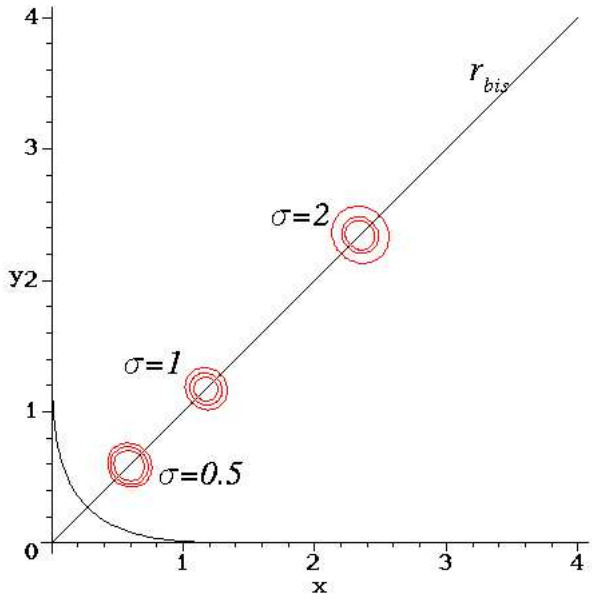



Figura 34: Detector de Beaudet en diferentes escalas del espacio Gaussiano.

El comportamiento de la primera observación de Deriche se muestra con claridad en la Figura 34. Esta figura muestra que el detector de Beaudet se mueve a lo largo de la recta bisectriz $r_{bis}$ de nuestro modelo para diferentes valores en el espacio de escala Gaussiano. Debido a que $\sigma_1 = \sigma_2$ se espera también, que la posición exacta de esquina se ubique a lo largo de dicha recta bisectora. Por otro lado, el cruzamiento por cero del Laplaciano no varia en el espacio de escala Gaussiano, esto significa que las coordenadas del punto de esquina serán las mismas no importando el grado de difuminado de una imagen. Si se aplica el algoritmo de detección de Deriche a nuestro modelo de esquina-L para los valores de $\sigma_1 = \sigma_2 = 1$, $A = 100$, $B = 0$, $\mu_1 = \mu_2 = 0$, que representa una esquina centrada en el origen con un valor de difuminado igual a uno, Figura 35, observamos que:

- Los extremos positivos del operador $K_{Beaudet}$ para distintos valores $\sigma$ están ubicados sobre la recta $r_{bis}$, Ecuación (III.21). Dado que las coordenadas de los extremos del detector de Beaudet se emplean para definir a la recta que bisecta al punto exacto por donde pasa la esquina, $r_{bis}$ es la ecuación de la recta buscada por el algoritmo de Deriche.

- El cruzamiento por cero del Laplaciano de la imagen, implica evaluar el Laplaciano

en los puntos definidos por la recta $r_{bis}$ y encontrar la coordenada cuando la curva así definida cruce por el eje horizontal. Los gráficos en la parte inferior de la Figura 35 representan este proceso.

- Aplicando el algoritmo de detección de Deriche para diferentes ángulos de apertura de la esquina obtenemos la Tabla III.1. La localización de las coordenadas de los puntos de esquina representados en esta tabla, se calcularon utilizando el paquete de matemática simbólica MAPLE. Las esquinas se generaron de forma que la recta bisectora tenga un ángulo de $45^0$ y pase por el origen, tal que la ecuación $r_{bis}$ es siempre $y = x$. Observando la Tabla III.1 tenemos que:

Tabla III.1: Coordenadas de los puntos de esquina del algoritmo de Deriche vs. nuestro criterio de ubicación.

| Ángulo de Apertura $\alpha$ en grados | Cruzamiento por Cero del Laplaciano | Punto de Intersección $r_{bis}$ y $M_L(x, y, \vec{P}) = 0.5$ | Referencia a la Figura 35 |
|---|---|---|---|
| 120 | (0.24190, 0.24290) | (0.42970, 0.42970) | (a) |
| 90 | (0, 0) | (0.54495, 0.54495) | (b) |
| 60 | (-0.83601, -0.83601) | (0.74442, 0.74442) | (c) |
| 45 | (-2.02111, -2.02111) | (0.93029, 0.23029) | (d) |

1. Para el caso de un esquina con un ángulo de $90^0$, el algoritmo de ubicación de esquinas propuesto por Deriche ubica a la esquina en las coordenadas (0,0) mientras que el punto de intersección entre $r_{bis}$ y $M_L(x, y, \vec{P}) = 0.5$ se ubica en las coordendas (.54495, .54495). Más aún, el punto de ubicación de Deriche no variará para diferentes difuminados de la imagen. En general, los puntos propuestos por Deriche para los ángulos representados en la Tabla III.1, son constantes con respecto al factor de difuminado.
2. Para la esquina con un ángulo de apertura inferior a $90^0$ el cruzamiento por cero del Laplaciano se desplaza en sentido contrario a la curva de contorno donde nosotros esperamos encontrar la esquina. Este efecto se muestra en las figuras 35.c y 35.d. Mientras que el cruce por cero del Laplaciano de la imagen se desplaza en dirección negativa del eje $x$, la curva $M_L(x, y, \vec{P}) = 05$ se desplaza en la dirección positiva del mismo eje.

3. Para una esquina con un $\alpha > 90^0$ el cruzamiento por cero del Laplaciano se desplaza hacia la curva de contorno $M_L(x, y, \vec{P}) = 0.5$.

- Es evidente que el cruzamiennto por cero del Laplaciano localiza a una esquina en la dirección del piso de la misma y su comportamiento es diametralmente opuesto a nuestro criterio. Más aún, para valores de $\alpha < 45^0$ es posible que el desplazamiento de la esquina se localize fuera de la región que se quiere analizar.

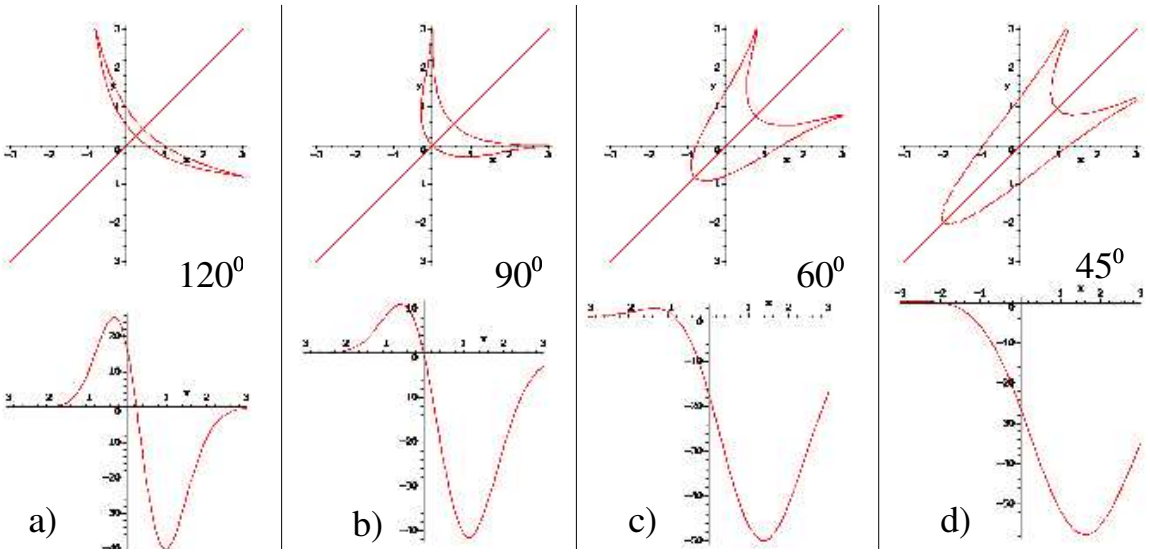



Figura 35: Cruzamiento por cero del Laplaciano de la imagen para distintos ángulos de apertura de la esquina. En los gráficos superiores se muestra el Laplaciano igual a cero ($\nabla^2(M_L) = 0$), la curva de contorno evaluada en 0.5 y la recta $r_{bis}$. En los gráficos inferiores se representa el cruzamiento por cero de $\nabla^2(M_L)$ evaluado en los punto definidos por $r_{bis}$.

### III.5.5 Detector Rohr

Finalmente, se analizará el comportamiento del criterio de ubicación de esquina-L propuesto por Karl Rohr [Rohr, 1992] con respecto a nuestro criterio. Rohr analiza el desplazamiento de una esquina-L, causado por: 1) el proceso de convolución del filtro Gaussiano y una función escalón definida en su modelo, 2) el ángulo de apertura de la esquina y la intensidad en los valores de gris de la imagen, ver sección II.2.3.1 página 29. Afirma que

la posición del punto de esquina se localiza en la línea con la pendiente más escarpada dada por $g_x^2 g_{xx} + 2g_x g_y g_{yy} + g_y^2 g_{yy} = 0$, donde $g$ es una función en niveles de gris y $g_x$, $g_y$, $g_{xx}$, $g_{yy}$ son las primeras y segundas derivadas de $g$ con respecto a $x$ y $y$ respectivamente. Después de su análisis, propone que la posición exacta de la coordenada $x$ de la esquina está dada por la solución de la siguiente ecuación implícita:

$$G(x') - t^2 x' \phi(x') = 0 \tag{III.22}$$

donde:

- $G(x) = \frac{1}{\sqrt{(2\pi)}} e^{-\frac{x^2}{2}}$ es un filtro Gaussiano. Este filtro en su forma bidimensional convolucionado con una función escalón unitaria, Ecuación II.6, es la base de su modelo paramétrico.
- $t = \tan(\frac{\beta}{2})$ es la pendiente del eje de simetría $\xi$ de la esquina con respecto al ángulo de apertura $\beta$.
- $x' = \frac{x}{q}$ y $q = \sqrt{1 + t^2}$.
- $\phi(x) = \int_{-\infty}^{x} G(\xi) d\xi$, Es la función error.

Las observaciones reportadas por Rohr son:

1. El desplazamiento del punto de esquina-L con respecto a su ángulo de apertura $\beta$ está dado por la relación no lineal expresada por la Ecuación (III.22).
2. La posición del punto de esquina es independiente de la altura entre el piso y la cresta de la esquina.
3. Existe una relación lineal entre el factor de difuminado y la posición del punto exacto de la esquina.

Las consideraciones anteriores se observan también en el comportamiento de nuestro criterio de la ubicación exacta de esquina. La Figura 36, muestra el desplazamiento de la posición $x$ con respecto al ángulo de apertura de la esquina dado por la Ecuación III.22, ver línea continua en dicha figura, y a nuestro criterio en diferentes valores de difuminado. En la construcción de esta gráfica se generaron esquinas a partir de $M_L(x, y, \vec{P})$ para los valores siguientes: $\mu_1 = \mu_2 = 0$, esquina centrada en el origen; $\vartheta_2 = 0$, recta $r_2$ sobre la

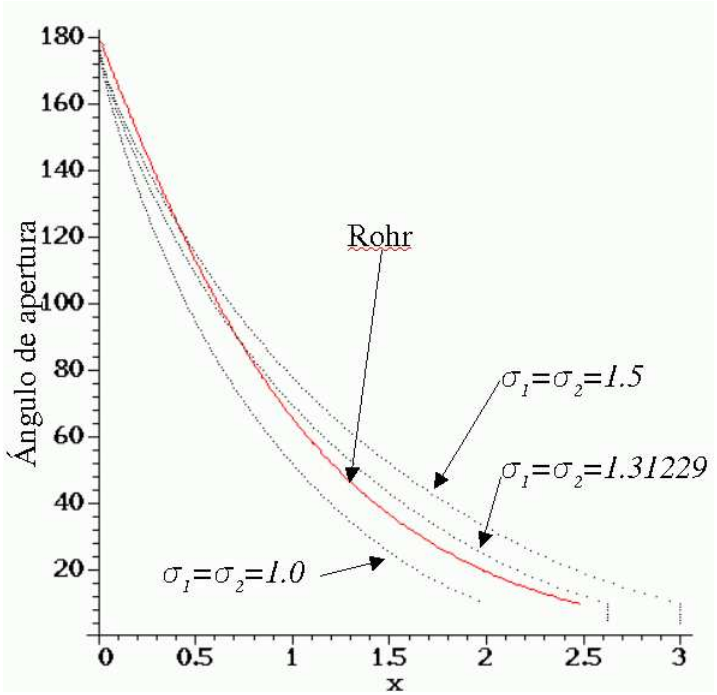



Figura 36: Desplazamiento de la posición de la esquina-L con respecto al ángulo de apertura. Esta gráfica muestra el comportamiento de nuestro modelo para diferentes factores de difuminado y la Ecuación (III.22).

parte positiva del eje $x$; $\vartheta_1$ en el intervalo de $[-89^0, 89^0]$, que representa las variaciones del ángulo de apertura de la esquina entre $[1^0, 179^0]$ con incrementos de $1^0$ entre las diferentes muestras; y tres factores de escala distintos $\sigma_1 = \sigma_2 = 1.0$, $\sigma_1 = \sigma_2 = 1.313229$ y $\sigma_1 = \sigma_2 = 1.5$. Es importante recordar, que nuestro criterio de ubicación se basa en la búsqueda del punto que representa la mínima distancia entre el punto de intersección de las rectas $r_1$ y $r_2$ y la ecuación implicita $M_L(x, y, \vec{P}) = 0.5$. La posición $x$ mostrada en esta gráfica se obtuvo empleando el algoritmo de *"curva de contorno"*, descrito en las secciones previas, y el método *"Down Hill Simplex"*. Para comparar este criterio en igualdad de circunstancias, se asignó $\sigma_1 = \sigma_2$, debido a que el modelo propuesto por Rohr no contempla valores distintos de difuminado. Analizando la Figura 36 se concluye que:

1. La ubicación del punto exacto de esquina con respecto a su ángulo de apertura tiene una relación no lineal. En el caso de una esquina a $90^0$ se encontro que para el valor de $\sigma_1 = \sigma_2 = 1.313229$ el desplazamiento de $x = 0.715669$ es similar en ambos

criterios. En particular, para ángulos superiores a $90^0$ nuestro criterio tiene menor desplazamiento que el propuesto por Rohr, para ángulos inferiores a $90^0$ ocurre lo contrario. En nuestro criterio el desplazamiento con respecto al eje $y$, para los mismos casos presentado en la Figura 36, tiene también un comportamiento no lineal. Este efecto se muestra en la Figura 37. El desplazamiento mayor se localiza cuando el ángulo de apertura de la esquina es de $90^0$. Rohr no consideró en su análisis el comportamiento del punto de esquina con respecto al eje $y$, debido a que en su modelo consideró al eje de simetría $\xi$ ubicado sobre el eje $x$, por tanto no existe desplazamiento. Sin embargo, si consideramos que el eje de simetría $\xi$ no se encuentra sobre el eje $x$, se puede observar como se comporta la coordenada $y$. Dicho comportamiento se muestra en la Figura 37.

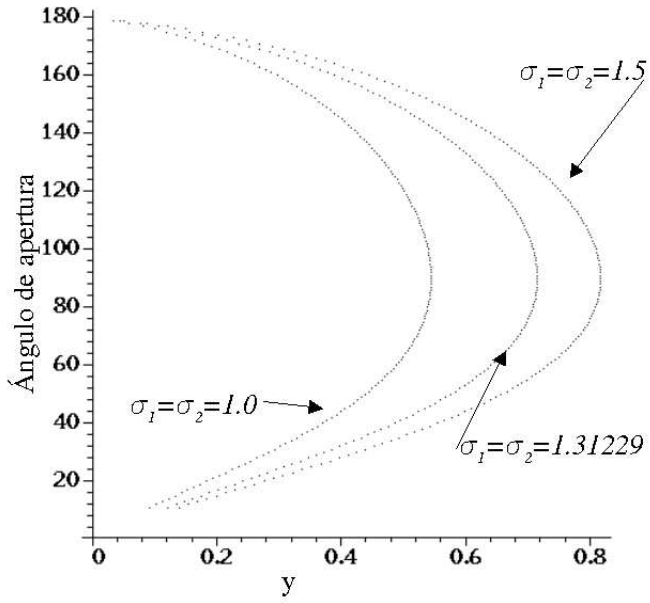



Figura 37: Desplazamiento de la posición $y$ de la esquina-L con respecto al ángulo de apertura. El máximo desplazamiento ocurre cuando el ángulo de apertura es de $\frac{\pi}{2}$.

2. Existe una relación proporcional entre diferentes factores de difuminado y el desplazamiento de la esquina. En la figuras 36 y 37 se observa con claridad que las curvas de ubicación de esquina para diferentes valores no sufren modificación en su forma, unicamente están afectadas por una constante. Por ejemplo, en la Figura 37,

la posición de $y$ para un ángulo de $90^0$, máximo desplazamiento, toma los valores $y(\sigma = 1) = 5.44929553138007e-01$, $y(\sigma = 1.313229) = 7.15617314878440e-01$ y $y(\sigma = 1.5) = 8.1739637022503e-01$ para los valores de $\sigma_1 = \sigma_2$ dados por 1.0, $+1.313229$ y 1.5, respectivamente. Es claro que los valores de la coordenada $y$ están escalados por $\sigma$, esto es, $y(\sigma = 1.5) \approx 1.5y(\sigma = 1)$.

El comportamiento de estos detectores con respecto al punto preciso de esquina $(x_e,y_e)$, localizado en la curva de contorno central $M_L(x, y, \vec{P}) = 0.5$ cuya distancia a $(x_0,y_0)$ es mínima, se resume como sigue:

- Los operadores de Beaudet, Dreschler y Nagel, y Kitchen y Rosenfeld ubican al punto preciso de esquina desplazado hacia la cresta de la esquina.

- El criterio de ubicación de Deriche, ubica al punto de esquina en la dirección del piso de la esquina. La curva que representa el Laplaciano igual a cero de una imagen, se desplaza en sentido opuesto a $M_L(x, y, \vec{P}) = 0.5$ cuando varia el ángulo de apertura de la esquina. El comportamiento del criterio de Deriche es contrario al observado en nuestro modelo.

- La curva de desplazamiento del punto de esquina sobre el eje $x$ de nuestro modelo y el propuesto por Rohr se comportan de forma similar. Más aún, el comportamiento del movimiento del punto exacto de esquina con respecto al ángulo de apertura de la esquina reportado por Rohr, se aplica también al observado en nuestro criterio de ubicación. Ambos criterios se comportan de manera similar, pero su deducción surge de conceptos diferentes; nuestro criterio se basa puramente en conceptos *geométricos* y el criterio de Rohr emplea *geometría diferencial.*

- Nuestro criterio toma en cuenta el desplazamiento causado por diferentes difuminados producidos por el tamaño de las rejillas fotosensibles contenidas en sensores CCD rectangulares. Los descritos anteriormente carecen de esta característica.

## III.6 Conclusiones

Se ha propuesto un modelo paramétrico nuevo de detección precisa de esquinas múltiples, que opera sobre imágenes digitales en tonos de gris. El modelo está basado en la

*parametrización de una función de distrubición Gaussiana*, la cual representa a los bordes de la estructura en estudio. El empleo de esta función como base del modelo es un elemento nuevo en el campo de detectores de esquinas.

Cada uno de los bordes de la esquina, caracterizados completamente por expresiones analíticas denominadas *funciones unitarias de borde*, se combinan a través de *operadores aritméticos simples* (suma, resta, división y multiplicación) con el propósito de representar estructuras más complejas. La facilidad con la que se generan nuevas estructuras es una de sus características principales.

El modelo opera sobre una ventana cuadrangular de la imagen y puede representar esquinas en cualquier dirección y ángulo de apertura. Así mismo, la esquina puede encontrarse en cualquier punto de la región de interés.

Nuestro modelo caracteriza los desplazamientos en las direcciones de los ejes principales del plano del sensor, producidos por el tamaño de las rejillas fotosensibles contenidas en sensores CCD rectangulares, en el momento de discretizar una escena. Este desplazamiento es medido por los diferentes difuminados que puede tener el vértice en estudio. Dicho elemento no ha sido tomado en consideración en la literatura previa.

Finalmente, como resultado del estudio de la geometría de *esquinas-L* se ha propuesto un nuevo criterio de localización precisa. Este criterio para el punto preciso de esquina, se define como: *el punto que representa la mínima distancia euclidiana entre el punto de intersección de las rectas de borde $r_1$ y $r_2$ y las coordenadas que satisfacen a la curva de contorno* $M_L(x, y, \vec{P}) = 0.5$.

# Capítulo IV

# Modelado de Datos y Optimización Multidimensional

## IV.1 Introducción

En el capítulo anterior se ha planteado un nuevo modelo analítico que describe las características de los fenómenos que nos interesa analizar: las esquinas-L y los vértices. Como resultado de nuestro modelo, una esquina-L está representada completamente por 8 parámetros ($\sigma_1$, $\vartheta_1$, $\mu_1$, $\sigma_2$, $\vartheta_2$, $\mu_2$, $A$ y $B$). En este capítulo describiremos los métodos que se utilizaron para ajustar este modelo matemático a un conjunto de datos reales. Estos datos se obtienen con una cámara PULNIX 9710 de $764 \times 484$ pixeles.

Al proceso de ajustar un conjunto de parámetros de un modelo dado, a un conjunto de observaciones se le conoce como *Modelado de Datos*. El modelado de datos se puede formular como un proceso de optimización global. Dicho proceso consiste en buscar el mejor conjunto de valores de los parámetros, dentro del universo de posibles valores que éstos puedan tomar, de tal forma que se ajusten lo mejor posible a nuestras observaciones.

Los datos observados generalmente no son exactos, están sujetos a errores en las mediciones: mala iluminación de escena, ruido inherente del sensor, insuficiente umbral en los niveles de grises que definen los bordes, etc. Típicamente los datos nunca se ajustan exactamente a los valores de los parámetros del modelo una vez que éste es corregido. Por tanto, se requiere cuantificar el error total del modelo. Este valor nos ayuda a conocer

*"que tan bueno"* es el ajuste. Una medida estadística que se emplea para cuantificar *que tan bueno* es un conjunto de parámetros, es el estimador *"chi-cuadrado"* ($\chi^2$). En el presente desarrollo se emplea al estimador $\chi^2$ como una medida de *aptitud* que nos indica *que tan bueno* es cada ajuste. En términos de optimización $\chi^2$ corresponde a nuestro criterio de optimización o función de *mérito*.

Por otro lado, la búsqueda de un extremo (máximo o mínimo), fuera del contexto del modelado de datos, es por si mismo un problema complejo. Si consideramos una simple función $f$ sin discontinuidades que dependa de una o más variables independientes, como la mostrada en la Figura 38, resulta difícil y en ocaciones imposible encontrar cual es el máximo con el valor más grande, aplicando simples evaluaciones de $f$. Lo mismo ocurre si se desea encontrar el mínimo con el valor más pequeño de $f$. Existen diferentes técnicas que se pueden emplear en este proceso, tales como: las basadas en la evaluación de la función de mérito, las basadas en el descenso de gradiente (desarrollada por Newton), las basadas en lo que se conoce como recocido simulado, y más recientemente las basadas en técnicas de computación evolutiva. Si la función de mérito es continua y unimodal, con un sólo máximo, el óptimo puede ser encontrado mediante movimientos definidos en la dirección de los gradientes locales [Salomon, 1998]. Para una revisión de ejemplos de programas y métodos de acelaración de búsqueda, se recomienda la lectura correspondiente a las referencias [Press, 1998] y [Stoer, 1980]. Si la función es multimodal, como la mostrada en la Figura 38, el método de descenso de gradiente sólo garantiza la existencia de un extremo local. Las técnicas evolutivas y de recocido simulado pueden ser aplicables para el problema de funciones multimodales. Si el planteamiento del problema requiere de más de una función de mérito, conocida en la literatura como *"optimización multi-objetivo"* [Gen, 2000], se pueden experimentar con las técnicas evolutivas. En resumen, la técnica de optimización por aplicar depende enteramente del tipo de problema. La idea de la mejor técnica, depende de un análisis particular del problema estudiado, sobre el rendimiento resultante de la implementación y experimentación de varias técnicas con el mismo problema. Este último concepto, es la motivación principal para implementar distintas técnicas de optimización multimensional, sobre nuestro detector preciso de esquinas múltiples.

Finalmente, el objetivo de este capítulo es mostrar el criterio de optimización que se utilizó en el ajuste de datos, así como explicar las técnicas de optimización multidimen-

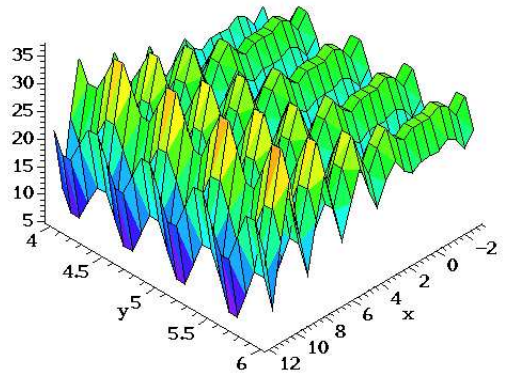



Figura 38: Función $f(x, y) = 21.5 + x\text{sen}(4\pi x) + y\text{sen}(20\pi y)$ con múltiples máximos o mínimos.

sional que se aplicaron en el desarrollo de este trabajo. Este capítulo tiene la estructura siguiente:

En la sección *Conceptos Básicos* se presenta una reseña breve del problema de optimización y algunos métodos que emplearon en la búsqueda de los extremos de una función.

En la sección *Criterio de Optimización* se explica el estimador $\chi^2$ aplicado al método de Levenberg-Marquardt como el criterio que nos permite evaluar *"que tan bueno"* es el ajuste para un conjunto de parámetros dados.

En la sección *Métodos de Optimización* se explican los métodos de *Downhill Simplex* para funciones multimensionales, el método de *Recocido Simulado* y el *Algoritmo Evolutivo* empleado en la obtención del mejor conjunto de parámetros que se ajustan a los datos.

Finalmente, en la sección *Conclusiones*, se presentan las observaciones y sugerencias generadas en la implementación de estos algoritmos.

## IV.2 Conceptos Básicos

En la literatura se han propuesto diversos métodos numéricos para el problema de búsqueda de mínimos o máximos. Dependiendo de cual es el aspecto que se enfatiza, los teóricos (la existencia de condiciones de una solución óptima) o los prácticos (procedimientos para la búsqueda de un óptimo) [Schwefel, 1995]. La *optimización* se clasifica hoy en día como una rama de las matemáticas aplicadas, investigación de operaciones, o como una rama del diseño de sistemas asistidos por computadora en el campo de la ingeniería. En esta sección se presentan los conceptos y las matemáticas básicas involucradas en los procesos de optimización.

**Problema de Optimización**

En optimización, un problema es típicamente especificado mediante un conjunto de $n$ parámetros $(x_1, ..., x_n)$, denotado por $\vec{x}$, y una función objetivo $f$, llamada función de aptitud. La meta del proceso de optimización es encontrar un conjunto de $n$ parámetros $\vec{x}$ tal que la función objetivo otorgue un valor *óptimo*, $f(x_1, ..., x_n) = f(\vec{x}) = \textit{óptimo}$ [Salomon, 1998]. El concepto *óptimo* significa *maximizar* $f$, que es equivalente a *minimizar* $-f$. Por tanto, maximizar o minimizar una función es equivalente a optimizar dicha función. Esta expresión es representada normalmente por,

$$minimizar \quad f(\vec{x})$$

donde $f : R^n \rightarrow R$ es una función real de $n$ variables $f(\vec{x}) = f(x_1, ..., x_n)$. Si $f$ es diferenciable y $\nabla f(\vec{x}) = (\partial f/\partial x_1, ..., \partial f/\partial x_n)$ es el gradiente de $f$, entonces el punto *mínimo* $\vec{x}_{min}$ de $f(\vec{x})$ es óptimo cuando el gradiente de la función $\nabla f(\vec{x}_{min}) = 0$.

**Método de Newton-Raphson**

El método de Newton-Raphson es un método numérico que se emplea generalmente para encontrar las raices de un polinomio. Se basa en la expansión de la serie de Taylor de una función a su primera aproximación. En su forma más simple si $y = f(x)$ y existen las $n + 1$ primeras derivadas en un intervalo alrededor de $x = a$ la expansión de Taylor esta dada por.

$$f(x) = f(a) + \frac{f'(a)}{1!}(x - a) + \frac{f''(a)}{2!}(x - a)^2 + ... + \frac{f^{(n)}(a)}{n!}(x - a)^n + ...$$

Aplicando la primera aproximación de la serie de Taylor para encontrar el punto $f(x) = 0$, donde $x_{actual}$ es la coordenada actual y $x_{actual+1}$ es la siguiente aproximación, se tiene,

$$0 = f(x) = f(x_{actual}) + \frac{f'(x_{actual})}{1!}(x_{actual+1} - x_{actual})$$
$$x_{actual+1} = x_{actual} - \frac{f(x_{actual})}{f'(x_{actual})}$$

Si se aplica el método de Newton al problema de optimización descrito anteriormente se tiene: sea $f(\vec{x}) = f(x_1, ..., x_n)$ una función objetivo con $n$ variables a minimizar, $\nabla f(\vec{x})$ es el gradiente, $\mathbf{H}(\vec{x}) = \left(\frac{\partial^2 f}{\partial x_j \partial x_k}\right), \quad j, k = 1, .., n$ es el Hessiano de $f$ y se sabe que el mínimo se encuentra cuando el gradiente es cero, $\nabla f(\vec{x}) = 0$. En las cercanías de un punto $\vec{x}_i$, se expande $f(\vec{x})$ en la segunda aproximación de Taylor,

$$f(\vec{x}) \approx f(\vec{x}_i) + (\vec{x} - \vec{x}_i) \cdot \nabla f(\vec{x}_i) + \frac{1}{2}(\vec{x} - \vec{x}_i) \cdot \mathbf{H} \cdot (\vec{x} - \vec{x}_i) \tag{IV.1}$$

tal que el gradiente es,

$$\nabla f(\vec{x}) = \nabla f(\vec{x}_i) + \mathbf{H} \cdot (\vec{x} - \vec{x}_i) \tag{IV.2}$$

En el método de Newton se asigna $\nabla f(\vec{x}) = 0$ para determinar el siguiente punto en la iteración,

$$\vec{x} - \vec{x}_i = -\mathbf{H}^{-1} \cdot \nabla f(\vec{x}_i) \tag{IV.3}$$

La parte izquierda de la ecuación representa los pasos finitos que se requieren para extraer el mínimo. La parte derecha de esta expresión implica el cálculo de la matriz del Hessiano. Esta matriz debe ser positiva definida, ya que de lo contrario no se puede calcular su matriz inversa. En general, cerca del mínimo no existe garantía que $\mathbf{H}$ sea positiva definida, por tal motivo existen métodos alternos que garantizan esta característica de $\mathbf{H}$, como por ejemplo, el método *cuasi-Newton* o el método *Levenberg-Marquardt* [Press, 1998], [Stoer, 1980].

**Metodo de Descenso de Gradiente**

Newton desarrollo el *método de descenso de gradiente*. Este método requiere del cálculo de gradiente $\nabla f(\vec{x})$ para poder localizar el mínimo de $f$. El método consiste en iniciar en un punto $\vec{x}_0$. El siguiente punto $\vec{x}_{t+1}$ está dado por una pequeña fracción $\lambda$ en la dirección del gradiente local $\nabla f(\vec{x}_t)$, calculado en el punto actual $\vec{x}_t$. Si $\lambda$ es lo suficientemente pequeña este procedimiento converge a un óptimo local. La siguiente ecuación es conocida como el método de descenso de gradiente:

$$\vec{x}_{t+1} = \vec{x}_t - \lambda \nabla f(\vec{x}_t) \tag{IV.4}$$

El método de descenso de gradiente únicamente puede ser aplicado a funciones objetivo $f(\vec{x})$ continuas y diferenciables. De esta forma, la búsqueda de un extremo está dirigida por una fracción del gradiente en el punto que se está evaluando $x_t$. Esto significa, que si $\lambda$ es grande el método está imposibilitado a alcanzar la cresta o valle de la función. Por otra lado, si $\lambda$ es demasiado pequeña para alcanzar un valle o una cresta, entonces el método de descenso de gradiente se ve imposibilitado de búscar en otros valles o crestas el extremo de $f(\vec{x})$; por ejemplo, vea la función representada en la Figura 38. De esta forma, el método de descenso de gradiente es considerado como un método de optimización local. En las referencias [Press, 1998], [Schwefel, 1995] y [Stoer, 1980] se pueden consultar distintos algoritmos de acelaración del método de descenso de gradiente.

# IV.3 Criterio de Optimización

## IV.3.1 Descripción del Método de Minimos Cuadrados

En teoría de modelado de datos cuando se quiere ajustar un conjunto de $N$ puntos conocidos $(x_i,y_i)$, también llamados datos observados, a cualquier modelo dado con $M$ parámetros ajustables $a_j$, $j = 1, 2, ..., M$, el modelo que predice la relación funcional existente entre las variables independientes y dependientes es,

$$f(x) = f(x; a_1, a_2, ..., a_M);$$

Para efectos de obtener los $a_j$ parámetros que obedecen a las $N$ diferentes mediciones experimentales se genera un sistema de ecuaciones,

$$f(x_i) = f(x_i; a_1, a_2, ..., a_M) \quad para \quad i = 1, 2, ..., N \tag{IV.5}$$

tal que $a_j$ satisfaga a dichas mediciones. En general, se necesita $N \geq M$ ecuaciones para poder obtener los $a_j$ parámetros que representen una solución única. Si $N > M$ el sistema se define como sobredeterminado, para los parámetros desconocidos $a_1, a_2, ..., a_M$, lo que en la práctica es deseable para validar al modelo en una gran diversidad de condiciones. Estos sistemas generalmente no tienen una solución única. Debido a que las valores de las observaciones $y_i$ están pertubadas por errores en las mediciones. Consecuentemente, en

lugar de buscar la solución exacta a (IV.5), el problema se transforma en buscar la *mejor solución* (problema de optimización). En términos estadísticos, es equivalente a encontrar los valores $a_j$ que representen la mínima diferencia cuadrática entre los datos observados y los que aporta el modelo. Este concepto se representa a través de la expresión siguiente:

$$minimizar \quad a_1, a_2, .., a_M : \quad \sum_{i=1}^{M} [y_i - f(x_i; a_1, a_2, ..., a_M)]^2 \tag{IV.6}$$

La Ecuación (IV.6) se conoce como *Ajuste de Mínimos Cuadrados*, o *Minimización del Residual de la Norma Euclidiana*. El ajuste de mínimos cuadrados es una medida estadística del *error* producido entre las observaciones, en nuestro caso los valores de intensidad de la imagen en tonos de gris y la hipótesis, en nuestro caso el modelo de esquina evaluado en los $\vec{P}$ parámetros. Esto implica, que se desea saber si la distribución estadística de un conjunto de $N$ eventos observados, corresponde a la misma función de distribución del valor esperado por la hipótesis. La respuesta está dada por el *estimador* $\chi^2$ representado por la ecuación siguiente:

$$Q = \chi^2 \equiv \sum_{i=1}^{M} \left( \frac{y_i - f(x_i; a_1, a_2, ..., a_M)}{\sigma_i} \right)^2 \tag{IV.7}$$

donde $x_i$ es el conjunto de observaciones y $\sigma_i$ es la desviación estándar. Los valores de la desviación estándar $\sigma_i$ son una medida del error de las *i-ésimas* observaciones, presumiblemente conocida. Si esta medida de error es desconocida, puede asignarse un valor constante de $\sigma_i = 1$, lo cual no altera el comportamiento estadístico de $\chi^2$ [Press, 1998].

Si asignamos $\sigma_i = 1$ en (IV.7), el criterio $\chi^2$ es equivalente al ajuste de mínimos cuadrados (IV.6). Por tanto, se quiere minimizar $Q$ para el conjunto de los $a_j$ parámetros desconocidos, representados por:

$$minimizar \quad Q(a_1, a_2, ..., a_M) = \chi^2 \equiv \sum_{i=1}^{M} [y_i - f(x_i; a_1, a_2, ..., a_M)]^2 \tag{IV.8}$$

El ajuste basado en $Q$ es el *estimador de máxima probabilidad* de los parámetros ajustados, la máxima probabilidad de que los $a_j$ sean los mejores, *si* los errores en las mediciones son independientes y distribuidos en forma normal con una desviación estándar constante.

En el presente trabajo se emplea el estimador $Q$ como la medida principal, que llamaremos *criterio*, el cual nos indica *que tan bueno* es el ajuste para un conjunto de parámetros dados.

### IV.3.2 Estimador $\chi^2$ y Modelo de Esquina-L

El proceso de localización de una esquina, se realiza por medio del ajuste de los parámetros de nuestro modelo $\vec{P}$, a la variación de las intensidades en una región de la imagen. Los parámetros que se desean modelar $\sigma_1$, $\vartheta_1$, $\mu_1$, $\sigma_2$, $\vartheta_2$, $\mu_2$, $A$ y $B$, describen el comportamiento físico y geométrico de los bordes que conforman una esquina-L, ver sección III.3 página 41. En términos del modelado de datos, la expresión $M'_L(x, y, \vec{P})$, Ecuación III.10, representan nuestra hipótesis y la región cuadrada de la imagen en estudio $I(u, v)$, nuestras observaciones.

Aplicando los conceptos de modelado de datos, nuestro problema consiste en encontrar un conjunto de parámetros $\vec{P} = (p_1, p_2, ..., p_n) \in \Re$, tal que para el conjunto de $m \times m$ observaciones $I(u_i, v_j)$, la diferencia cuadrática entre los datos $I(u_i, v_j)$ y nuestro modelo $M'_L(x_i, y_j, \vec{P})$ sea mínima. Esta relación es expresada por el estimador de máxima probabilidad:

$$\chi^2 \equiv Q = F(\vec{P}) = \sum_{j=1}^{m} \sum_{i=1}^{m} \left[ I(u_i, v_j) - M'_L(x_i, y_j, \vec{P}) \right]^2 \qquad \text{(IV.9)}$$
$$minimización \quad sobre \quad p_1, p_2, ..., p_n$$

donde:

$n$ es el número de parámetros a minimizar de nuestro modelo, Ecuación (III.12).

$m$ es el número de datos observados, equivalente al número de columnas multiplicado por el número de renglones de la región en estudio. $m$ en relación con nuestro detector preciso de esquinas múltiples es expresado por:

$$m = 2w + 1 \qquad \text{(IV.10)}$$

recuerde que $m$ es el rango $[-w, w]$ donde está definido el sistema coordenado del modelo $(x, y)$, ver sección III.2.

$\vec{P} = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$ son los parámetros de $M'_L$ que describen el comportamiento de una esquina-L.

$M'_L(x_i, y_j, \vec{P})$ es el modelo de esquina evaluado en los $\vec{P}$ parámetros en el sistema coordenado del modelo ($x$,$y$).

$I(u_i, v_j)$ son los valores de intensidad de una imagen en niveles de gris. $I(u_i, v_j)$ es una subimagen cuadrada de $m \times m$ pixeles, en el sistema coordenado de la imagen ($u$,$v$).

$F(\vec{P})$ es el estimador $\chi^2$.

La expresión descrita por $Q$, Ecuación IV.9, incluye dos sistemas coordenados. El sistema coordenado de imagen $(u, v)$ y el sistema coordenado del modelo $(x, y)$. Los sistemas coordenados de la imagen y del modelo son diferentes. El origen de una imagen, posición (0,0), en el sistema coordenado de la imagen, está ubicada en el extremo superior izquierdo. Apartir de ese punto las coordenadas ($u$,$v$) representan un desplazamiento positivo en números enteros hacia la derecha y hacia abajo respectivamente. Por otro lado, el sistema coordenado del modelo ($x$,$y$) es un sistema coordenado cartesiano, definido en el intervalo $[-w,+w]$, Figura 39. Por tanto, es necesario un conjunto de expresiones que permitan la transformación entre los sistemas coordenados. Para un punto conocido ($u_0$,$v_0$) en el sistema coordenado de la imagen, si se quiere explorar la ventana definida en el intervalo $[-w, +w] \in I$ las ecuaciones de transformación son:

$$\begin{array}{lllll} u_i = u_o + x_i & ; & x_i = i - 1 - w & ; & i = 1, 2, ..., 2w + 1 \\ v_j = v_o - y_i & ; & y_i = -j + 1 + w & ; & j = 1, 2, ..., 2w + 1 \end{array} \qquad \text{(IV.11)}$$

Por ejemplo, si se quieren construir dos matrices de datos **A** y **B**. Donde la matriz **A** representa los valores de intensidad de la imagen u observaciones. **B** representa los valores del modelo o hipótesis. Para una ventana cuadrada en el rango de $[-2, +2]$, en la vecindad del punto conocido $(u_0 = 100, v_0 = 100)$ de la imagen digital, se tiene:

En notación matricial, los elementos de una matriz pueden ser representados por:

$$\mathbf{A}(j, i) \quad i = 1, 2, ..., 2w + 1 \quad j = 1, 2, ..., 2w + 1$$

donde $i$ son las columnas y $j$ los renglones de la matriz **A**. La ventana de interés se define en el rango $[-2, +2]$, por tanto $w = 2$ y $m = 2w + 1 = 5$, ver Ecuación

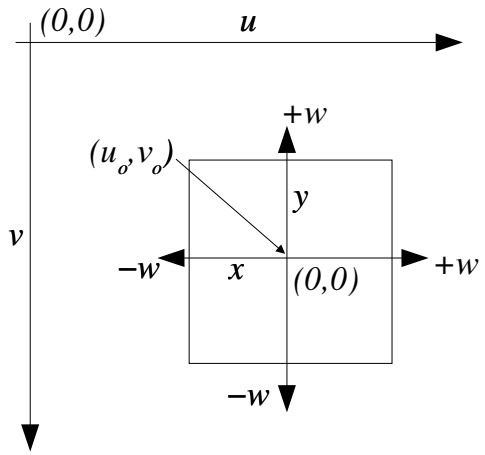



Figura 39: Sistema coordenado de la imagen $(u, v)$ y el sistema coordenado del Modelo $(x, y)$. La relación entre los dos sistemas coordenados esta dada por la Ecuación (IV.11).

(IV.10). Esta misma notación se aplica a la matriz **B**. **A** y **B** son entonces matrices de $5 \times 5$ elementos.

Primero obtenemos la correspondencia entre los subíndices de la matriz $\mathbf{B}(j, i)$, y el sistema coordenado del modelo $(x, y)$. Aplicando (IV.11) obtenemos la siguiente tabla:

Tabla IV.1: Construcción de los subíndices de la matriz de modelo $\mathbf{B}(j, i)$ relacionados con el sistema coordenado del modelo $(x, y)$.

| $i$ | $x_i = i - 1 - w$ | $j$ | $y_i = -j + 1 + w$ |
|---|---|---|---|
| 1 | $1 - 1 - 2 = -2$ | 1 | $-1 + 1 + 2 = 2$ |
| 2 | $2 - 1 - 2 = -1$ | 2 | $-2 + 1 + 2 = 1$ |
| 3 | $3 - 1 - 2 = 0$ | 3 | $-3 + 1 + 2 = 0$ |
| 4 | $4 - 1 - 2 = 1$ | 4 | $-4 + 1 + 2 = -1$ |
| 5 | $5 - 1 - 2 = 2$ | 5 | $-5 + 1 + 2 = -2$ |

Finalmente, cada elemento de $\mathbf{B}(j, i)$ contiene los valores del modelo en la coordenada $(x_i, y_j)$ para los $\vec{P}$ parámetros dados. Por lo tanto, $(x_i, y_j)$ toman los valores

descritos en la Tabla IV.1. Esta relación se expresa como sigue:

$$\mathbf{B}(j,i) = M'_L(x_i, y_j, \vec{P})$$

De forma análoga, se obtiene la correspondencia de los subíndices de la matriz $\mathbf{A}$, con las coordenadas de los pixeles, en el sistema coordenado de la imagen $(u, v)$. La coordenada de imagen que se quiere analizar es (100,100). Alrededor de este punto se contruye una ventana definida en el intervalo $[-2, +2]$. Aplicando la Ecuación IV.11 se obtine dicha relación expresada en la tabla siguiente:

Tabla IV.2: Construcción de los subíndices de la matriz $\mathbf{A}(j,i)$, relacionados con el sistema coordenado de la imagen $(u, v)$, en la vecindad del punto $(100, 100)$.

| $i$ | $u_i = u_0 + x_i$ | $j$ | $v_j = v_0 - y_i$ |
|---|---|---|---|
| 1 | $100 + (-2) = 98$ | 1 | $100 - 2 = 98$ |
| 2 | $100 + (-1) = 99$ | 2 | $100 - 1 = 99$ |
| 3 | $100 + 0 = 100$ | 3 | $100 - 0 = 100$ |
| 4 | $100 + 1 = 101$ | 4 | $100 - (-1) = 101$ |
| 5 | $100 + 2 = 102$ | 5 | $100 - (-2) = 102$ |

Cada elemento de $\mathbf{A}(j,i)$ contiene los valores de intensidad en niveles de gris, de los pixeles $(u_i, v_j)$ de la imagen original. Por lo tanto, $(u_i, v_j)$ toman los valores descritos en la Tabla IV.2. Esta relación se expresa como sigue:

$$\mathbf{A}(j,i) = I(u_i, v_j)$$

En suma, la Ecuación (IV.11) proporciona la relación existente entre los sistemas coordenados del modelo $(x, y)$, la imagen $(u, v)$ y su representación matricial $(j, i)$. Nuestra meta consiste ahora, en encontrar el estimador de máxima probabilidad $Q$ entre las matrices $\mathbf{A}(j,i)$ y $\mathbf{B}(j,i)$ así construidas.

Existen diversos métodos para resolver el problema de mínimos cuadrados, los cuales dependen del tipo de modelo: lineal o no lineal. La estructura representada por $M'_L(x, y, \vec{P})$ es una expresión no lineal, por tanto centraremos la atención en este tipo de modelos.

Finalmente, la Ecuación (IV.9) representa nuestro ***criterio de optimización***. En las secciones subsecuentes la función $Q$ es llamada también *función de mérito*, *función de*

*costo*, *función de aptitud* o *función objetivo*, dependiendo de la técnica de optimización que se este utilizando.

### IV.3.3 Mínimos Cuadrados para Modelos No Lineales

En el presente capítulo se plantean una serie de métodos, tendientes a encontrar *el mejor* conjunto de parámetros $\vec{P}$ que se ajusten a la imagen observada. Los métodos descritos en la sección IV.4 están enfocados a la búsqueda de un óptimo ***global***. En esta sección se muestra el método de Levenberg-Marquardt desarrollado específicamente para el problema de mínimos cuadrados considerando modelos no lineales. El método de Levenberg-Marquardt se encarga de encontrar, *el mejor* conjunto de parámetros, siempre y cuando los valores de $\vec{P}$ estén los suficientemente cercanos a un mínimo. Por esta razón, el método de Levenberg-Marquardt es considerado un método de optimización ***local***.

El método de Levenberg-Marquardt se emplea para acelerar la búsqueda del óptimo global, en la estructura general de nuestro proceso de optimización. En otras palabras, los procesos de optimización Down Hill Simplex, Recocido Simulado y Evolutivo, *proponen* un conjunto de parámetros. Estos valores alimentan al método de Levenberg-Marquardt. Si los valores propuestos están cercanos a un mínino, entonces Levenberg-Marquardt los entregará optimizados. Esto implica, que el proceso de optimización global requiere únicamente de un ciclo de proceso para la búsqueda de un extremo. En caso contrario, el método de Levenberg-Marquardt no modifica los valores propuestos por los procesos de optimización global.

El desarrollo de mínimos cuadrados para modelos no lineales, consiste en buscar un conjunto de expresiones que nos permitan encontrar rápidamente un extremo. Para ello, se emplean los métodos de Newton-Raphson y descenso de gradiente introducidos en la sección IV.2, página 72. Dicho desarrollo lo mostramos a continuación.

#### IV.3.3.1 Ecuaciones Generales

La Ecuación (IV.9) representa nuestro criterio de optimización. Esta expresión puede aproximarse al segundo orden, cerca del mínimo, por:

$$F(\vec{P}) \approx \gamma + \nabla F(\vec{P}) \cdot \vec{P} + \frac{1}{2}\vec{P} \cdot \mathbf{H} \cdot \vec{P} \tag{IV.12}$$

donde $\nabla F(\vec{P})$ es el gradiente de $\chi^2$ y $\mathbf{H}$ es el Hessiano. Si la aproximación es buena, puede emplearse el método de Newton, Ecuación (IV.3) para encontrar el mínimo $\vec{P}_{min}$ a partir de la posición actual $\vec{P}_{act}$ en un sólo salto. Esta expresión es:

$$\vec{P}_{min} = \vec{P}_{act} + \mathbf{H}^{-1} \cdot [-\nabla F(\vec{P}_{act})] \tag{IV.13}$$

A esta expresión se le conoce como *el método de la Inversa del Hessiano.*

Si la aproximación de $\chi^2$ al segundo orden de la serie de Taylor en el punto actual $\vec{P}_{act}$ no es buena, está *lejos* del mínimo local, entonces la búsqueda del mínimo puede intentarse con el método de *descenso de gradiente.* Esto es:

$$\vec{P}_{act+1} = \vec{P}_{act} - \lambda \nabla F(\vec{P}_{act}) \tag{IV.14}$$

El gradiente de la función de mérito $\chi^2$ es:

$$\frac{\partial F(\vec{P})}{\partial \vec{P}_k} = -2 \sum_{j=1}^{m} \sum_{i=1}^{m} [I(u_i, v_j) - M_L(x_i, y_j, \vec{P})] \frac{\partial M_L(x_i, y_j, \vec{P})}{\partial \vec{P}_k} \quad k = 1, 2, ..., n \tag{IV.15}$$

El Hessiano de $\chi^2$ esta dado por:

$$\begin{aligned} \frac{\partial^2 F(\vec{P})}{\partial \vec{P}_k \partial \vec{P}_l} = \quad & 2 \sum_{j=1}^{m} \sum_{i=1}^{m} \left[ \frac{\partial M_L(x_i, y_j, \vec{P})}{\partial \vec{P}_k} \frac{\partial M_L(x_i, y_j, \vec{P})}{\partial \vec{P}_l} \right. \\ & \left. - [I(u_i, v_j) - M_L(x_i, y_j, \vec{P})] \frac{\partial^2 M_L(x_i, y_j, \vec{P})}{\partial \vec{P}_k \partial \vec{P}_l} \right] \\ & k = 1, 2, ..., n \quad l = 1, 2, ..., n \end{aligned} \tag{IV.16}$$

Para simplicar es conveniente definir:

$$\beta_k \equiv -\frac{1}{2} \frac{\partial F(\vec{P})}{\partial \vec{P}_k} \qquad \alpha_{kl} \equiv \frac{1}{2} \frac{\partial^2 F(\vec{P})}{\partial \vec{P}_k \partial \vec{P}_l} \tag{IV.17}$$

Con esta notación y haciendo $[\alpha] = \frac{1}{2}\mathbf{H}$, la expresión de la inversa del Hessiano (IV.13) se convierte en:

$$\sum_{l=1}^{n} \alpha_{kl} \delta \vec{P}_l = \beta_k \tag{IV.18}$$

Este conjunto de ecuaciones es resuelto mediante incrementos $\delta\vec{P}_l$ que, sumada a la aproximación actual, nos proporcionan la siguiente aproximación. Obedeciendo al esquema de *desceso de gradiente* se convierte en:

$$\delta\vec{P}_l = constante \times \beta_l \tag{IV.19}$$

Para evitar el cálculo del Hessiano de la función de mérito $\chi^2$, un método usual es construir el Hessiano a partir de las primeras derivadas parciales [Schwefel, 1995] como sigue:

$$\alpha_{kl} = \sum_{j=1}^{m}\sum_{i=1}^{m}\left[\frac{\partial M_L(x_i, y_j, \vec{P})}{\partial\vec{P}_k}\frac{\partial M_L(x_i, y_j, \vec{P})}{\partial\vec{P}_l}\right] \tag{IV.20}$$

#### IV.3.3.2 El algoritmo de Levenberg-Marquardt

La condición para que $F()$ sea mínima, definida por $\beta_k = 0; \forall k$, es independiente de como $[\alpha]$ es definida. La idea principal de este algoritmo es hacer variaciones suaves entre los extremos del *método de la Inversa del Hessiano* (IV.18) y el *método de descenso de gradiente* (IV.19) a partir de la modificación de la definición de la matriz $[\alpha]$. El término *constante* en el descenso de gradiente (IV.19) es especificado como:

$$\delta\vec{P}_l = \frac{1}{\lambda\alpha_{ll}}\beta_l \qquad ó \qquad \lambda\alpha_{ll}\delta\vec{P}_l = \beta_l \tag{IV.21}$$

La observación principal de Levenberg-Marquardt es que las ecuaciones (IV.21) y (IV.18) pueden ser combinadas si se define una matriz nueva $\alpha'$ de la forma siguiente:

$$\begin{aligned}\alpha'_{jj} &\equiv \alpha_{jj}(1-\lambda)\\ \alpha'_{jk} &\equiv \alpha_{jk} \quad (j \neq k)\end{aligned} \tag{IV.22}$$

entonces los dos métodos dados por (IV.21) y (IV.18) son mezclados aplicando la siguiente expresión:

$$\sum_{l=1}^{n}\alpha'_{kl}\delta\vec{P}_l = \beta_k \tag{IV.23}$$

Cuando $\lambda$ es muy grande, la matriz $\alpha'$ es forzada a convertirse en matriz *diagonal dominante*, de tal forma que la Ecuación (IV.23) tiende a ser idéntica que (IV.21). Por otra lado, si $\lambda$ se aproxima a cero, la Ecuación (IV.23) tiende a convertirse en (IV.18). Dado un conjunto de valores iniciales $n$ para los parámetros $\vec{P}$, el algoritmo de Levenberg-Marquardt se define como sigue:

1. Se calcula el valor de $\chi^2$ dado por los parámetros $\vec{P}$.
2. Se toma un valor modesto de $\lambda$, por ejemplo $\lambda = 0.001$.
3. Se resuelve el sistema de ecuaciones (IV.23) para $\delta\vec{P}$ y se evalúa $\chi^2$ en $F(\vec{P} + \delta\vec{P})$.
4. Si $F(\vec{P} + \delta\vec{P}) \geq F(\vec{P})$, se incrementa $\lambda$ en un factor de 10 y se regresa al paso 3.
5. Si $F(\vec{P} + \delta\vec{P}) < F(\vec{P})$, se reduce $\lambda$ en un factor de 10, se actualiza la solución $\vec{P} \leftarrow \vec{P} + \delta\vec{P}$, y se regresa al paso 3.
6. El proceso termina después de 2 ó 3 iteraciones si el valor de $\chi^2$ decrece en una cantidad despreciable. Esto se puede calcular cuando el valor absoluto de la diferencia entre los valores de $\chi^2$ sea inferior a 0.1. Otro criterio de paro es cuando el valor $\lambda \to 0$, como por ejemplo cuando $\lambda < 1.0e^{-15}$.

Finalmente, el método de Levenberg-Marquardt calcula la matriz de covarianza entre el modelo ajustado y los datos, asignando $\lambda = 0$ al final del proceso.

#### IV.3.3.3 Implementación del Algoritmo de Levenberg-Marquardt

La implementación del algoritmo de Levenberg-Marquardt que se utilizó, está basada en las rutinas del libro *Numerical Recipes in C* [Press, 1998], correspondientes a este algoritmo. Para obtener las derivadas parciales de $M'_L(x, y, \vec{P})$, con respecto a los parámetros $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B$ se empleo el paquete de matemática simbólica MAPLE.

La rutina **Criterio_ch2_L** controla todo el proceso de minimización del algoritmo de Levenberg-Marquardt, para una matriz de datos **Datos[1,...,W][1,...,W]** que representa los valores de intensidad de una región de la imagen y un vector de parámetros **a[1,...,Nmodel+2]** con **Nmodel** números de parámetros a minimizar. Al final del proceso, esta rutina entrega el valor de $\chi^2$ y los mejores valores de $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B$. Es importante mencionar que este algoritmo entrega el valor de un mínimo local, el cual no necesariamente es el mejor. La rutina **Criterio_ch2_L** interactúa con la rutina de **mrqmin**, *Numerical Recipes* y con la rutina **corner**. Un diagrama simple de interacción se muestra en la Figura 40, su código fuente se describe en el Ápendice BLA.

La rutina **mrqmin** realiza una iteración del método de Levenberg-Marquardt, pasos 2 al 5 de la sección IV.3.3.2. Esta rutina llama a **mrqcof** para obtener la matrix $[\alpha]$,

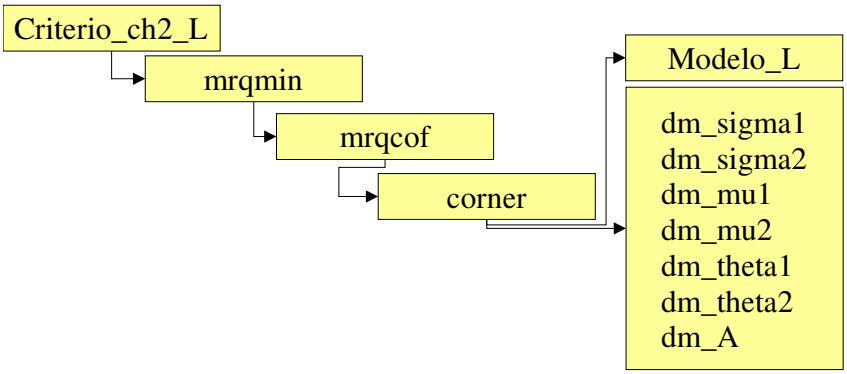



Figura 40: Rutinas utilizadas por **Criterio_ch2_L**.

Ecuación (IV.20), y el vector $\beta$ de las ecuaciones (IV.17) y (IV.18). **mrqcof** llama a la rutina **corner**, para obtener los valores de nuestro modelo.

La rutina **corner** calcula el valor de nuestro modelo $y \equiv M'_L(x_i, y_i, \vec{P})$ y sus derivadas parciales $dyda \equiv \partial M_L(x_i, y_i, \vec{P})/\partial \vec{P}$ evaluados en el punto $(x_i, y_i)$ con los valores de los parámetros $\vec{P}$. La rutina original de *Numerical Recipes* se construye a través de una función $y$, la cual depende de sólo una variable independiente $x$ y de los parámetros del modelo **a**. Esta función es de la forma $y = f(x; \vec{a})$. Sin embargo, nuestro modelo depende de las coordenadas del modelo $(x_i, y_i)$. Nuestro modelo contiene dos variables independientes en lugar de una. Por tanto, nuestra función es de la forma $y \equiv M_L(x_i, y_i; \vec{a})$. Para agregar la coordenada faltante $y_i$, en la rutina **mrqmin** el término $x = x_i$ es reemplazado por el valor del subíndice $x = i$ de las coordenadas $(x_i, y_i)$, de tal forma que en la función **corner** únicamente se inspeccione el índice $indice = x = i$.

La rutina **Criterio_ch2_L** efectúa los siguientes pasos:

1. Variables de Entrada:

   **Datos[1,...,W][1,...,W]**. Matriz de datos correspondientes a los valores de intensidad de una región en tonos de gris de la imagen.

   **a[1,...,Nmodel+2]**. Es el vector de parámetros $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B$ a minimizar. En la primera iteración los valores de **a** están dados por el proceso de optimización global: *DownHill Simplex*, *Recocido Simulado* o *Evolutivo*.

**ia[1,...,Nmodel+2]**. Es un vector de control que nos indica cuales son los parámetros a ajustar. Si $\mathbf{ia}[i] = 0$ significa que el *i-ésimo* parámetro debe ser ajustado. Si $\mathbf{ia}[i] = 1$ significa que el *i-ésimo* parámetro no debe ajustarse.

**W**. Es el tamaño de la ventana o región cuadrada dada por $W = 2w + 1$.

**Nmodel**. Número de parámetros a minimizar.

Los elementos $\mathbf{a}[\mathbf{Nmodel} + \mathbf{1}] = s_{U_1}$ y $\mathbf{a}[\mathbf{Nmodel} + \mathbf{2}] = s_{U_2}$, son los signos en la dirección de apertura de la esquina previamente calculados, ver sección III.4.2 página 47. $\mathbf{ia}[\mathbf{Nmodel} + \mathbf{1}] = \mathbf{ia}[\mathbf{Nmodel} + \mathbf{2}] = \mathbf{1}$, son parámetros que no deben ser ajustados.

2. La matriz **Datos[1,...,W][1,...,W]** se transforma en el vector de observaciones **y[i]**. Los vectores **x_cor[i]** y **y_cor[i]** representan los valores de las coordenadas en el *sistema coordenado del modelo.*

3. El método de Levenberg-Marquardt se deja operar libremente, pasos 1 a 5 de la sección IV.3.3.2.

4. El proceso termina cuando, después de 2 iteraciones, la diferencia entre los valores de $\chi^2$ pueda ser considerado despreciable. En nuestro caso cuando $\chi^2 < 0.1$, o bien $\lambda < 1 \times 10^{-15}$. El proceso se detiene también, si los valores de los parámetros propuestos por **mrqmin** quedan fuera de los límites permitidos, sección III.2. La rutina **Limites_L** es la encargada de este control.

5. Finalemente, **Criterio_ch2_L** entrega el valor de $\chi^2$ y el vector de parámetros ajustados **a**, al término de las iteraciones.

La rutina **corner** evalúa el modelo y sus derivadas parciales. Es importante recordar que el modelo de esquina $M'_L(x, y, \vec{P})$, Ecuación (III.10), puede evaluarse numéricamente empleando una función de error $\phi(z)$. $\phi(z)$, Ecuación (III.6), es una rutina matemática la cual se implementa en el lenguaje de *programación C* como; *erf(z)*. Por simplicidad, hablar de $\phi()$ ó *erf()* es equivalente. Por otro lado, recordemos también que:

$\vec{P} = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$, son los parámetros desconocidos que se desean ajustar a la región de la imagen en estudio. $\vec{P}$ caracteriza completamente el comportamiento físico y geométrico de una esquina-L.

$s_{U_1}$ y $s_{U_2}$ representan la orientación de la esquina, tomando como referencia los cuatro cuadrantes que forma el sistema coordenado del modelo $(x, y)$. $s_{U_1}$ es el signo en la dirección del eje $x$. $s_{U_2}$ es el signo en la dirección del eje $y$, ver sección III.4.2.

$r_1 = x - y\tan(\vartheta_1) - \mu_1$ y $r_2 = y - x\tan(\vartheta_2) - \mu_2$ definen las rectas de los bordes que representan a una esquina-L, vea sección III.2.

Por último, cada una de las expresiones analíticas llamadas por **corner** se codifican a partir de las siguientes rutinas:

**Modelo_L**. Evaluación del modelo de esquina $M_L(x, y, \vec{P})$, en un punto $(x,y)$ para los $\vec{P}$ parámetros.

$$\textbf{Modelo\_L} = \left(s_{U_1}\frac{1}{2}\phi\left(\frac{\sqrt{2}r_1}{2\sigma_1}\right) + \frac{1}{2}\right)\left(s_{U_2}\frac{1}{2}\phi\left(\frac{\sqrt{2}r_2}{2\sigma_2}\right) + \frac{1}{2}\right)A + B \qquad \text{(IV.24)}$$

**dm_sigma1**. Derivada parcial del modelo con respecto al factor de difuminado $\sigma_1$ en la dirección de la línea de borde $r_1$: $\partial M_L(x, y, \vec{P})/\partial\sigma_1$.

$$\textbf{dm\_sigma1} = -\frac{\sqrt{2}As_{U_1}}{2\sqrt{\pi}\sigma_1^2}r_1 e^{-\left(\frac{r_1^2}{2\sigma_1^2}\right)}\left(s_{U_2}\frac{1}{2}\phi\left(\frac{\sqrt{2}r_2}{2\sigma_2}\right) + \frac{1}{2}\right) \qquad \text{(IV.25)}$$

**dm_mu1**. Derivada parcial del modelo con respecto al punto central $\mu_1$: $\partial M_L(x, y, \vec{P})/\partial\mu_1$.

$$\textbf{dm\_mu1} = -\frac{\sqrt{2}As_{U_1}}{2\sqrt{\pi}\sigma_1}e^{-\left(\frac{r_1^2}{2\sigma_1^2}\right)}\left(s_{U_2}\frac{1}{2}\phi\left(\frac{\sqrt{2}r_2}{2\sigma_2}\right) + \frac{1}{2}\right) \qquad \text{(IV.26)}$$

**dm_theta1**. Derivada parcial del modelo con respecto al ángulo de rotación $\vartheta_1$ correspondiente a la línea de borde $r_1$: $\partial M_L(x, y, \vec{P})/\partial\vartheta_1$.

$$\textbf{dm\_theta1} = -\frac{\sqrt{2}As_{U_1}}{2\sqrt{\pi}\sigma_1}y(1 + \tan^2(\vartheta_1))e^{-\left(\frac{r_1^2}{2\sigma_1^2}\right)}\left(s_{U_2}\frac{1}{2}\phi\left(\frac{\sqrt{2}r_2}{2\sigma_2}\right) + \frac{1}{2}\right) \qquad \text{(IV.27)}$$

**dm_sigma2**. Derivada parcial del modelo con respecto al factor de difuminado $\sigma_2$ en la dirección de la línea de borde $r_2$: $\partial M_L(x, y, \vec{P})/\partial\sigma_2$.

$$\mathbf{dm\_sigma2} = -\frac{\sqrt{2}As_{U_2}}{2\sqrt{\pi}\sigma_2^2} r_2 e^{-\left(\frac{r_2^2}{2\sigma_2^2}\right)} \left( s_{U_1} \frac{1}{2}\phi\left(\frac{\sqrt{2}r_2}{2\sigma_1}\right) + \frac{1}{2}\right) \tag{IV.28}$$

**dm_mu2**. Derivada parcial del modelo con respecto al punto central $\mu_2$: $\partial M_L(x, y, \vec{P})/\partial\mu_2$.

$$\mathbf{dm\_mu2} = -\frac{\sqrt{2}As_{U_2}}{2\sqrt{\pi}\sigma_2} e^{-\left(\frac{r_2^2}{2\sigma_2^2}\right)} \left( s_{U_1} \frac{1}{2}\phi\left(\frac{\sqrt{2}r_1}{2\sigma_1}\right) + \frac{1}{2}\right) \tag{IV.29}$$

**dm_theta2**. Derivada parcial del modelo con respecto al ángulo de rotación $\vartheta_2$ correspondiente a la línea de borde $r_2$: $\partial M_L(x, y, \vec{P})/\partial\vartheta_2$.

$$\mathbf{dm\_theta2} = -\frac{\sqrt{2}As_{U_2}}{2\sqrt{\pi}\sigma_2} x(1+\tan^2(\vartheta_2)) e^{-\left(\frac{r_2^2}{2\sigma_2^2}\right)} \left( s_{U_1} \frac{1}{2}\phi\left(\frac{\sqrt{2}r_1}{2\sigma_1}\right) + \frac{1}{2}\right) \tag{IV.30}$$

**dm_A**. Derivada parcial del modelo con respecto al valor de amplitud de la esquina $A$: $\partial M_L(x, y, \vec{P})/\partial A$.

$$\mathbf{dm\_A} = \left( s_{U_1} \frac{1}{2}\phi\left(\frac{\sqrt{2}r_1}{2\sigma_1}\right) + \frac{1}{2}\right) \left( s_{U_2} \frac{1}{2}\phi\left(\frac{\sqrt{2}r_2}{2\sigma_2}\right) + \frac{1}{2}\right) \tag{IV.31}$$

**dm_B**. Derivada parcial del modelo con respecto al piso de la esquina $B$: $\partial M_L(x, y, \vec{P})/\partial B$.

$$\mathbf{dm\_B} = 1 \tag{IV.32}$$

# IV.4 Métodos de Optimización

## IV.4.1 Estrategia de Nelder y Mead

Nelder y Mead (1965) desarrollaron un método de minimización de funciones multidimensionales llamado *Downhill Simplex* [Press, 1998] o estrategia *Simplex* [Schwefel, 1995]. Este método no tiene relación con el método de programación lineal simplex propuesto

por Dantzing en 1966. La idea original se centra en reducir lo más posible, el número de pruebas simultáneas que se identifican experimentalmente para el diseño de problemas de orden factorial [Schwefel, 1995]. El número mínimo de situaciones simultáneas es $n+1$ eventos. El método *Downhill Simplex* toma entonces $n+1$ situaciones simultáneas, llamadas vértices, para cada uno de los $n$ parámetros desconocidos.

Un *simplex* es una figura geométrica en un espacio *n-dimensional*, compuesta por $n+1$ puntos o vértices y todos los segmentos de interconexión entre ellos. En dos dimensiones un simplex es un triángulo, en tres dimensiones es un tetraedro, en n-dimensiones es un poliedro. El método comienza con la generación de un simplex, lo que es equivalente a generar $n+1$ vértices. Se evalúa la función objetivo en cada uno de estos puntos. A partir de esta etapa se generan una serie de movimientos geométricos de los vértices tendientes a conservar el volumen del simplex (proceso no degenerativo) definidos en las siguientes reglas de cada iteración, para el caso de minimización:

- El vértice cuyo valor de la función objetivo sea el más grande, *"punto alto"*, se mueve a través de la cara opuesta del simplex donde se encuentre el *"punto bajo"*, a este movimiento se le llama *reflexión*, Figura 41.b. El *"punto bajo"* es el véritce con el valor más pequeño de la función objetivo, el ***mejor punto***, y el *"punto alto"* representa al ***peor punto***. El movimiento de reflexión es útil para buscar una región más promisoria en el espacio de búsqueda.

- Si el movimiento de reflexión mejora al ***peor punto***, entonces el simplex se *expande* en esa misma dirección, Figura 41.d, lo que equivale a explotar exhaustivamente la región más promisoria.

- Si se alcanza un "valle" el último vértice calculado puede ser también el ***peor punto***. En este caso se intenta una *contracción* del simplex, punto medio entre el peor y el mejor vértice, Figura 41.c, con el objeto de buscar una salida del valle. La prueba consiste en comparar si el movimiento de *contracción* mejora el valor del ***peor punto*** con respecto al ***segundo peor***.

- Si la prueba anterior falla, el simplex efectúa una *contracción múltiple* alrededor del *"punto bajo"*, Figura 41.e, el cual reduce su volumen para intentar pasar a través del valle.

La búsqueda termina cuando la magnitud del vector que define la distancia entre el mejor y el peor vértice es inferior a cierta tolerancia. La Figura 41, muestra los movimientos de la estrategia de optimización *simplex*.

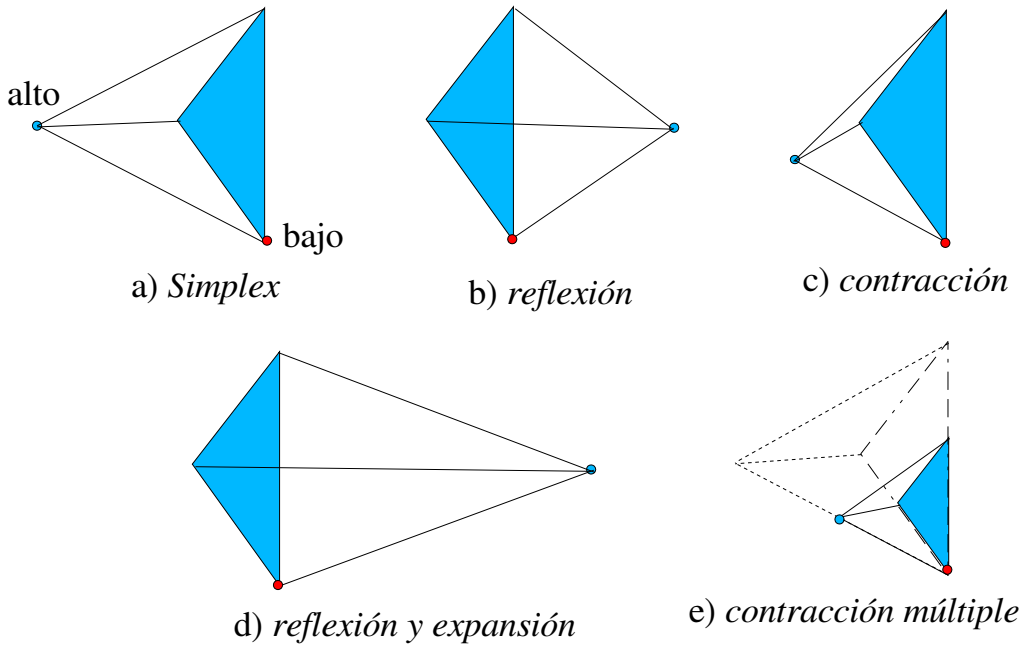



Figura 41: Movimientos geométricos definidos en el método *Downhill Simplex*. a) *Simplex* de 3 parámetros en el inicio de cada iteración, representado por un tetraedro. El *simplex* al final de una iteración puede ser uno de los casos siguientes: b) *Reflexión* del punto "alto" en la dirección del "bajo"; c) *Contracción* de la cara del *simplex* definido por el punto "alto" y "bajo"; d) *Reflexión y expansión* sobre el punto "alto"; e) *Contracción múltiple* alrededor del punto "bajo".

El procedimiento básico consiste en:

1. Dada una matriz $p_{(i,j)}, i = 1, \ldots, n+1; j = 1, \ldots, n$ que representa los $n+1$ vértices de un *simplex* con $n$ parámetros iniciales, $y_{(i)} = F(p_{(i,j)})$ es el vector que contiene los valores de la función objetivo $F()$ en el *i-ésimo* vértice y una tolerancia $\varepsilon > 0$ (i.e., $\varepsilon = 10^{-12}$).

2. Se determinan los índices $(b, w, s)$ que representan el ***mejor***, el ***peor*** y el ***segundo peor*** vértice, respectivamente

$$y_{(b)} = \min\{y_{(i)}, i = 1, \ldots, n+1\}$$
$$y_{(w)} = \max\{y_{(i)}, i = 1, \ldots, n+1\}$$
$$y_{(s)} = \max\{y_{(i)}, i = 1, \ldots, n+1, i \neq w\}$$

3. Se contruye un vector de peso

$$\bar{c} = \frac{1}{n} \sum_{i=1, i\neq w}^{n+1} p_{(i,k)} \quad k = 1, \ldots, n$$

y se efectúa un movimiento de reflexión normal

$$p' = 2\bar{c} - p_{(w,k)}$$

En los subsecuentes pasos, el subíndice $k = 1, \ldots, n$ representa a los $n$ parámetros.

Si $F(p') < F(p_{(b,k)})$ entonces se va al paso 5.

4. Si la reflexión no mejora el valor del ***peor*** punto, se compara el movimiento de reflexión con respecto al ***segundo peor*** vértice.

Si $F(p') \geq F(p_{(s,k)})$ entonces intente una contracción, vaya al paso 6, de lo contrario asigne $p_{(w,k)} = p'$ y ejecute el paso 8.

5. Expansión. Se explota exhaustivamente la región más promisoria, construyendose

$$p'' = 2p' - \bar{c}.$$

Si $F(p'') < F(p_{(b,k)})$ entonces $p_{(w,k)} = p''$, de lo contrario $p_{(w,k)} = p'$. En cualquier caso, ejecute el paso 8.

6. Contracción. Se contruye,

$$p'' = 0.5(\bar{c} + p_{(w,k)})$$

Si $F(p'') \geq F(p_{(w,k)})$ realice una contracción múltiple, ejecute el paso 7, de lo contrario asigne $p_{(w,k)} = p''$ y ejecute el paso 8.

7. Contracción múltiple alrededor del **mejor** punto $b$.

$$p_{(i,k)} = 0.5(p_{(b,k)} + p_{(i,k)}) \quad i = 1, \ldots, n+1, i \neq b$$

8. Terminación. El proceso finaliza cuando una fracción de la magnitud del vector de distancia entre el ***peor*** y el ***mejor*** punto son inferiores a cierta tolerancia.

$$2\frac{|y_{(w)} - y_{(b)}|}{|y_{(w)}| + |y_{(b)}|} \leq \varepsilon$$

   Si el criterio de terminación no se cumple entonces regresa al paso 2

En general, la estrategia de optimización *simplex* tiene las siguientes características:

- Requiere únicamente de evaluaciones de la función objetivo, por tanto, es inecesario el cálculo del gradiente o el Hessiano de la función de mérito. Esta característica es deseable si la obtención de las derivadas parciales con respecto a los parámetros de la función objetivo es difícil o imposible de calcular.
- El número de hipótesis que se prueban en una iteración es de $n + 1$.
- El criterio de búsqueda del mínimo es geométrico. El mínimo se encuentra cuando el volúmen del poliedro que se forma a través de los vértices es pequeño. El poliedro tiende a convertirse en un punto. Si la función objetivo no es muy compleja, que contenga un gran número de extremos cercanos, esta estrategia puede llegar a converger rápidamente a un extremo (local al menos), en caso contrario no se recomienda su empleo.

#### IV.4.1.1 Implementación de la estrategia de Nelder y Mead

La estrategia *simplex* aplicada a nuestro problema se emplea en dos procesos de optimización:

1. Como un proceso de optimzación global para buscar el mejor conjunto de parámetros $\vec{P}$, pertenecientes a $M'_L(x, y, \vec{P})$, que se ajusten mejor a nuestras observaciones (una ventana de la imagen). En este caso, la función objetivo o de mérito es el valor de $\chi^2$, Ecuación (IV.9). La rutina **Down_hill_L** se encarga de este proceso.

2. Como un proceso de optimización para encontrar el punto exacto ($x_e$,$y_e$) donde se ubica la esquina. Recuerde, que nosotros definimos el punto exacto de esquina $(x_e, y_e)$, como el punto que representa la mínima distancia euclidiana $D_{min}$ entre el punto de intersección $(x_0, y_0)$ de las rectas de borde $r_1$ y $r_2$, con la curva de contorno definida por $M_L(x, y, \vec{P}) = 0.5$, ver sección III.4.4 página 53. En este caso la función objetivo es la distancia entre ($x_0$, $y_0$) y un punto en la curva de contorno central de $M_L(x, y, \vec{P})$. La rutina **Modelo_L_Curvatura** es la responsable de este proceso.

Ambos procesos se basan en el código propuesto en las rutinas **amoeba** y **amotry** del libro *Numerial Recipes in C* [Press, 1998]. Las rutinas originales **amoeba** y **amotry** suponen que la función objetivo tiene una variable independiente de la forma $y = f(x)$. Sin embargo, nuestro problema consta de dos variables independientes, las cuales son las coordenadas $(x, y)$ del sistema coordenado del modelo. Por tanto el código fuente se modificó para acoplarlo a los criterios de minimización $Q$ y $D_{min}$. Una descripción detallada de estas modificaciones se incluyen en la sección V.3.5.

**Rutina Down_hill_L**

La rutina **Down_hill_L** inicializa el primer *simplex* dada una función objetivo $Q$, sobre la que se realiza el ajuste de mínimos cuadrados de la región de la imagen. La región de la imagen está representada por la matriz **Datos[1,...,W][1,...,W]** para el vector de parámetros a minimizar **P_M[1,..,N_par]**. $Q$ se obtiene por medio de la rutina **Criterio_ch2_L**, la cual ejecuta el método de Levenberg-Marquardt. El valor de $Q$ para cada uno de los vértices es almacenado en el vector **yd[1,...,N_par+1]**. La matriz **pd[1,...,N_par+1][1,...,N_par+2]** contiene los **N_par+1** vértices del *simplex*, para los **N_par** parámetros desconocidos; recuerde que los índices **N_par+1** y **N_par+2** son los signos en la dirección del ángulo de apertura de la esquina calculados previamente, vea la sección IV.3.3.2, y no son considerados parámetros desconocidos. Un esquema simple de interacción con otras rutinas se muestra en la Figura 42.

**Down_hill_L** contruye el *simplex* de inicio, donde cada uno de los vértices se encuentra a la misma distancia uno de otro. Esto significa, que el *simplex* inicial representa una muestra uniforme entre las cotas superior e inferior, de los valores permitidos para cada uno de los parámetros del modelo. En forma específica el *simplex* inicial se construye de la forma siguiente:

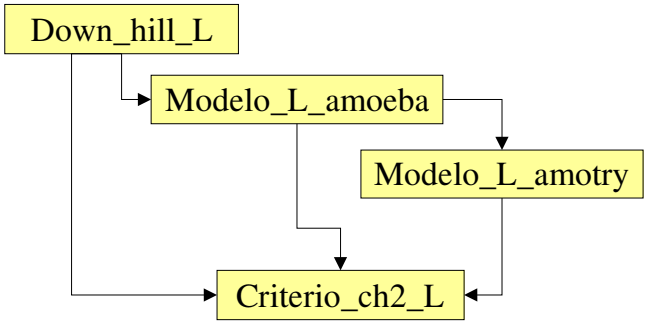



Figura 42: Rutinas utilizadas por **Down_hill_L**.

1. El vector de parámetros a minimizar **P_M** está definido de la forma siguiente:

$$\mathbf{P\_M} = (\sigma_1, \vartheta_1, \mu_1, \sigma_2, \vartheta_2, \mu_2, A, B)$$

donde el número de parámetros a minimizar es **N_par**=8

2. Las cotas superior e inferior de cada uno de los parámetros están contenidas en los vectores **P_max[1,...,N_par+2]** y **P_min[1,...,N_par+2]** respectivamente. Las cotas son definidas por el usuario considerando los siguientes límites: 1) $\sigma_1$ y $\sigma_2$ están definidos en el intervalo $(0, w]$, 2) $\mu_1$ y $\mu_2$ en el intervalo $(-w, w)$, 3) $\vartheta_1$ y $\vartheta_2$ en el intervalo $(-\frac{\pi}{2}, +\frac{\pi}{2})$, 4) $A$ y $B$ en el intervalo $[0, 255]$. Estos límites son los valores extremos de las cotas. Sin embargo, el usuario puede reducir estos límites para efectos de aumentar la velocidad de respuesta del algoritmo de optimización. Para ello, el usuario debe tener un conocimiento *a priori* de la estructura que se estudiará.

3. Se calcula un vector del tamaño de los incrementos **inc[1,...,N_par]** que debe tener cada uno de los vértices. Este vector es de la forma,

$$\mathbf{inc}(j) = \frac{\mathbf{P\_max}(j) - \mathbf{P\_min}(j)}{\mathbf{N\_par} + \mathbf{1}} \quad j = 1, \ldots, \mathbf{N\_par}$$

4. Cada uno de los vértices **pd** representan una muestra uniforme en el intervalo de búsqueda definido por el usuario. **pd** se construye a través de la expresión siguiente:

$$\mathbf{pd}(i, j) = \mathbf{P\_min}(j) + (i - 1)\mathbf{inc}(j) \quad i = 1, \ldots, \mathbf{N\_par} + 1; j = 1, \ldots, \mathbf{N\_par}$$

donde, el índice $i$ representa a los 9 vértices del simplex y el índice $j$ representa los 8 parámetros a minimizar.

El simplex inicial construido de esta forma lo llamaremos *vértice inicial uniformemente muestreado*, ver tablas V.5, V.7, V.8.

Las rutinas **Modelo_L_amoeba** y **Modelo_L_amotry** se encargan de realizar los movimientos propuestos por la estrategia de Nelder y Mead y corresponden a los procedimientos **amoeba** y **amotry** modificados con el objeto de poder operar a partir del método de minimización de Levenberg-Marquardt.

**Rutina Modelo_L_Curvatura**

Una vez que se tiene el mejor conjunto de parámetros $\vec{P}$, que se ajustan a la región de la imagen, se ejecuta la rutina **Modelo_L_Curvatura** para localizar el punto exacto $(x_e, y_e)$ donde se ubica la esquina. Este proceso localiza el punto $(x_e, y_e)$ que pertenece a $M_L(x, y, \vec{P}) = 0.5$ y representa la mínima distancia euclidiana $D(a, b)$ con respecto al punto de intersección $(x_0, y_0)$ de las rectas de borde $r_1$ y $r_2$ que definen a una esquina-L, ver sección III.4.4.

La rutina **Modelo_L_Curvatura** genera el *simplex* inicial para una función objetivo $D(a, b)$. El número de elementos desconocidos es **NP**=2 que representa los valores de la coordenada $(x_e, y_e)$. Por tanto, el número total de vértices es **MP**=3. El vector **y[1,..,MP]** almacena el valor de la función objetivo para las coordenadas $(x_s, y_s)$ propuestas por los movimientos del método de Nelder y Mead. La matriz **p1[1,..,MP][1,...,NP]** contiene las coordenadas de cada vértice del *simplex*. La función objetivo $D(a, b)$ requiere del cálculo del punto $a = (x_0, y_0)$ y otro punto arbitrario $b = (x_s, y_s)$ con las siguientes características.

- $a = (x_0, y_0)$ es el punto de intersección de $r_1$ y $r_2$, obtenido a través de los valores ya conocidos de $\vec{P}$. La rutina **sol** entrega el valor de $(x_0, y_0)$ apartir de la siguiente expresión, Ecuación (III.13):

$$\mathbf{sol} = \begin{cases} x_0 = -\frac{\tan(\vartheta_1)\mu_2+\mu_1}{\tan(\vartheta_2)\tan(\vartheta_1)-1} \\ \\ y_0 = -\frac{\tan(\vartheta_2)\mu_1+\mu_2}{\tan(\vartheta_2)\tan(\vartheta_1)-1} \end{cases}$$

- $b = (x_s, y_s)$ debe ser un punto que pertenece a $M_L(x, y, \vec{P}) = 0.5$. Además, la línea que forma con $(x_0, y_0)$ debe estar orientada entre las rectas de borde $r_1$ y $r_2$. En

otras palabras, se encuentra entre el ángulo de apertura de la esquina. El algoritmo que denominamos *"Curva de Contorno"*, es ejecutado por la rutina **Intercepta**, a fin de localizar a la coordenada $(x_s, y_s)$. Este algoritmo es descrito ampliamente en la sección III.4.3.

Una vez calculados los puntos $a = (x_0, y_0)$ y $b = (x_s, y_s)$, la rutina **distancia** devuelve el valor de la distancia entre dichos puntos:

$$\textbf{distancia} = (x_s - x_0)^2 + (y_s - y_0)^2$$

La Figura 43 muestra un diagrama de interacción de **Modelo_L_Curvatura** con otras rutinas.

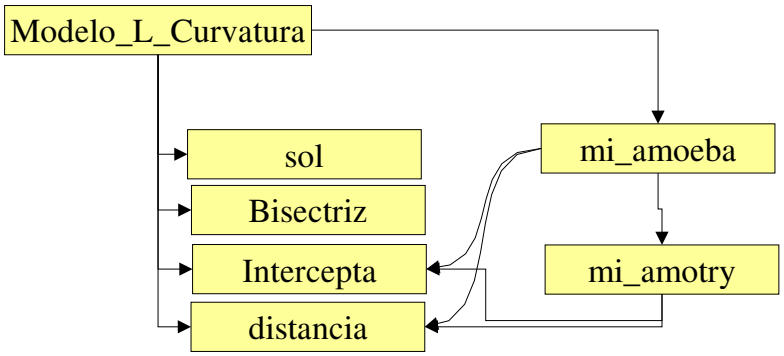



Figura 43: Rutinas utilizadas por **Modelo_L_Curvatura**.

Este algoritmo se ejecuta a través de los siguientes 5 pasos:

1. Se genera una coordenada $(x_l, y_l)$ que obedece a los movimientos de la estrategia *Simplex*. Este punto generalmente no pertenece a la curva de contorno $M_L(x, y, \vec{P}) = 0.5$.

2. Se calcula el ángulo $\beta$ de la recta generada por los puntos $(x_l, y_l)$ y $(x_0, y_0)$.

3. El algoritmo *"Curva de Contorno"*, ejecutado por la rutina **Intercepta**, genera una recta $r_s$ con el ángulo $\beta$ y el punto $(x_0, y_0)$. Si $r_s$ está orientada entre $r_1$ y $r_2$ entonces regresa el punto $(x_s, y_s)$ que satisface a $M_L(x, y, \vec{P}) = 0.5$. En caso contrario penaliza a la coordenada propuesta por el *simplex* y le asigna un valor de coordenada sobre las rectas de borde $r_1$ o $r_2$ dependiendo de cual de los ángulos, definidos por ambas rectas, sea el más cercano a $\beta$. Este algoritmo regresa siempre un punto $(x_s, y_s) \in M_L(x, y, \vec{P}) = 0.5$.

4. Se evalúa la función objetivo $D(a, b)$ con esta coordenada nueva, $b = (x_s, y_s)$.

5. Se continúa con el método Downhill Simplex hasta lograr la convergencia. Para cada vértice nuevo propuesto por el método Downhill Simplex, se regresa al paso 1. Con estos pasos se garantiza que cada vértice satisface a la curva de contorno que nos interesa analizar.

Los tres vértices iniciales del *simplex* en la rutina **Modelo_L_Curvatura**, se construyen mediante la siguiente lógica: 1) el primer vértice $C_1$ se genera con el ángulo de la recta bisectriz $r_{bis}$, Ecuación (III.21), calculado por la rutina **Bisectriz**, 2) el segundo vértice $C_2$ es un punto en la recta de borde $r_1$ y 3) el último vértice $C_3$ corresponde a un punto en la recta de borde $r_2$. La Figura 44, muestra el *simplex* inicial a partir de esta lógica.

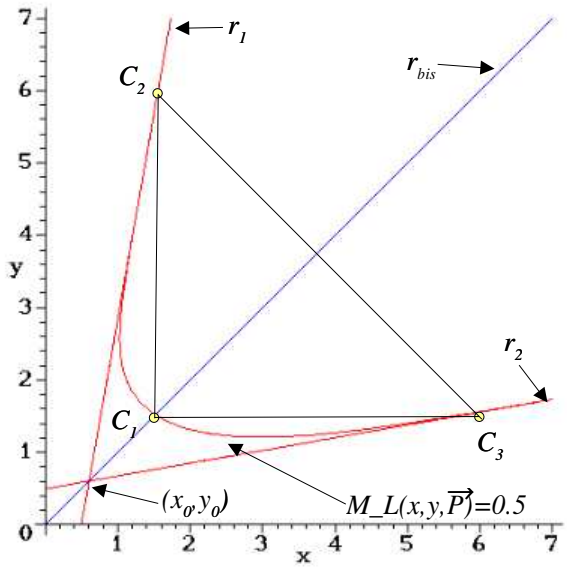



Figura 44: *Simplex* de inicio generado por la rutina **Modelo_L_Curvatura**. Los vértices $C_1$, $C_2$, $C_3$ delimitan la figura geométrica del *simplex* inicial.

### IV.4.2 Estrategia de Recocido Simulado

El método *Recocido Simulado* es una técnica empleada en problemas de optimización de gran escala, especialmente cuando el mínimo global se encuentra embebido o escondido dentro de un ambiente de extremos locales cercanos. En la práctica, este método se ha aplicado en forma eficiente al problema clásico del *"agente viajero"* [Press, 1998]. Dicho problema consiste en encontrar el itinerario que produce la ruta menos costosa para que un viajero pueda visitar N-ciudades distintas.

La idea central del método proviene de la análogia con el problema de recocido o forja de los metales, también conocido como tratamiento térmico del acero, a través del cual se obtienen diferentes propiedades físicas. Cuando un metal se calienta hasta su estado líquido, altas temperaturas, las moléculas se mueven libremente unas con respecto a otras. Seguido, se somete a un proceso de enfriado, provocando que el metal baje su temperatura más o menos rápidamente. A este proceso se le conoce como forja o "annealing" en inglés. Dependiendo del esquema de enfriamiento los átomos o moléculas tienen más o menos tiempo para encontrar su posición en un patrón ordenado, por ejemplo una estructura tipo cristal. El patrón más ordenado, que corresponde al mínimo global de la energía libre, puede ser obtenido únicamente cuando el proceso de enfriamiento es lo **suficientemente lento**. En caso contrario el patrón de congelamiento corresponde a uno u otro mínimo local de energía. Por tanto, *encontrar el equilibrio entre el tiempo necesario para que las moléculas se ordenen a una temperatura y la velocidad de enfriamiento dadas*, es el objetivo a lograr.

Boltzmann fue el primero en formular una *ley de probabilidad*, que relaciona la temperatura con las frecuencias relativas de una gran cantidad de microestados o configuraciones posibles de las interacciones partícula a partícula, con objeto de obtener un equilibrio térmico. Metropolis *et al.* (1953) simularon el equilibrio térmico de un sólido en una bañera caliente. Ellos emplearon el promedio extraído del método de Monte-Carlo, para generar configuraciones nuevas donde las partículas tengan menos energía. La energía libre nueva $E_{nueva}$ se compara con la que fue formada en la antigua configuración $E_{anterior}$. Si $E_{nueva} \leq E_{anterior}$ entonces la nueva configuración "sobrevive" y forma la base para la siguiente perturbación. El nuevo estado puede "sobrevivir" también, si $E_{nueva} > E_{anterior}$ pero debe cumplir con una cierta probabilidad $w$ dada por:

$$w = \frac{1}{c} e^{\left(\frac{E_{anterior} - E_{nueva}}{KT}\right)} \tag{IV.33}$$

donde $K$ es la constante de Boltzmann, $T$ es la temperatura actual y $c$ es una constante empleada para normalizar la distribución de probabilidad. El algoritmo de Metropolis ofrece los siguientes elementos:

1. Una descripción de las posibles configuraciones del sistema.

2. Un generador aleatorio de cambios en la configuración. Estos cambios son "opciones" presentadas al sistema.

3. Un función objetivo $E$, energía del sistema, que debe ser minimizada al término del proceso.

4. Un parámetro de control $T$, temperatura, y un *esquema de recocido*, el cual nos indica la forma de "enfriar" el sistema. El esquema de recocido responde, *al como* después de muchos cambios aleatorios en la configuración la temperatura $T$ desciende al siguiente "paso" y ofrece información de que tan largo es dicho "paso".

Kirkpatrick, *et al.*, (1983) y Cerny (1985) publicaron métodos de optimización basados en el algoritmo de simulación de Metropolis. Estos métodos son usados frecuentemente y se les conoce como *procedimientos de Recocido Simulado* (SA[1]). Existen dos lazos principales en un proceso SA.

- Lazo externo, que representa el esquema de descenso de temperatura

  $$T_{nueva} = f(T_{anterior}) < T_{anterior}$$

  generalmente se asigna un esquema del tipo $T_{nueva} = \alpha T_{anterior}, \quad 0 < \alpha < 1$. En nuestra implementación $\alpha = 0.8$.

- Lazo interno de espera hasta que el estado de equilibrio sea encontrado. La simulación de Metropolis se realiza a $T = constante$, hasta que no ocurra alguna configuración que mejore el rendimiento del sistema.

[1] *Simulated Annealing* por sus siglas en inglés

Schwefel (1995) reporta el siguiente pseudocódigo que expresa en más detalle el proceso SA.

1. Inicialización:
   Dada una posición de inicio $x^{(0,0)}$
   una temperatura inicial $T^{(0)}$
   una longitud o "ancho" $d^{(0)}$ para las variaciones de $x$
   Asignar $x^* = \tilde{x} = x^{(0,0)}, \quad k = 0, \ell = 1.$

2. Simulación de Metropolis.
   Construir $x^{(k,\ell)} = \tilde{x} + zd^{(k)}$,
   donde $z$ está distribuida uniformemente para todos los componentes $z_i$, $i = 1, \ldots, n$ en el rango $z_i \in \left[-\frac{1}{2}, +\frac{1}{2}\right]$, o distribuida en su forma normal acorde a $w(z_i) = \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{1}{2}z_i^2\right)}$.

   Si $F(x^{(k,\ell)}) < F(x^*)$, asignar $x^* = x^{(k,\ell)}$
   Si $F(x^{(k,\ell)}) < F(\tilde{x})$, ejecute el paso 4;
   en caso contrario genere un número aleatorio, $r$, en el intervalo $[0, 1]$.
   Si $r \leq e^{\left(\frac{F(x^{(k,\ell)}) - F(\tilde{x})}{T^{(k)}}\right)}$ , (existe una probabilidad que el nuevo estado sobreviva) ejecute el paso 4.

3. Revisión de equilibrio.
   Si $F(x^*)$ no mejora con respecto al último N-punto de prueba, entonces ejecute el paso 5.

4. Lazo interno de espera hasta encontrar el equilibrio.
   Asigne $\tilde{x} = x^{(k,\ell)}$, incremente $\ell \leftarrow \ell + 1$, y ejecute el paso 2.

5. Criterio de terminación.
   Si $T^{(k)} \leq \varepsilon$, termina la búsqueda. El resultado es expresado por $x^*$.

6. Esquema de enfriamiento, equivalente al lazo externo.
   Asigne $\tilde{x} = x^{(k+1,0)} = x^*$, y
   $T^{(k+1)} = \alpha T^{(k)}, \quad 0 < \alpha < 1.$
   Eventualmente decrezca $d^{(k+1)} = \beta d^{(k)}, \quad 0 < \beta < 1.$
   Asigne $\ell = 1$, incremente $k \leftarrow k + 1$, y ejecute el paso 2.

Dos importantes cuestionamientos se presentan al observar el pseudocódigo del algoritmo SA. El primero en relación al proceso de terminación: ¿Qué tan grande debe ser la última fase en el estado de equilibrio? o bién, ¿Cúal criterio debe ser empleado para detener la búsqueda de un óptimo a una temperatura dada? El segundo en relación al proceso de equilibrio térmico: ¿Cúal es la magnitud o escala en cada paso del proceso de enfriamiento? Existen muchos esquemas empíricos que se han propuesto para responder parcialmente estas preguntas [Schwefel, 1995]. Por ejemplo, un resumen de convergencias puede ser consultado en el libro de Van Laarhoven y Aarts (1987). Sin embargo, es necesario evaluar el algoritmo para distintos valores de inicialización y observar su comportamiento o convergencia, en función del problema en estudio. Dicha tarea puede resultar difícil de implentar si el problema es complejo. Específicamente, se requiere decidir los valores de inicialización de los siguiente parámetros libres:

$T^{(0)}$. Temperatura de inicio.
$d^{(0)}$. Tamaño inicial de cada paso.
$\alpha$. Factor de enfriamiento.
$\beta$. Factor de reducción de cada paso.
$N$. El número de puntos o estados a evaluar en el estado de equilibrio.
$\varepsilon$. La cota inferior de temperatura, donde se encuentra el estado de mínima energía libre.

En conclusión, la característica más importante del algoritmo SA reside en la habilidad de escapar de un óptimo local a través de una degeneración permitida de sus configuraciones con una cierta probabilidad. En contraparte, la determinación de la velocidad de enfriamiento y el tiempo de equilibrio a fin de lograr una rápida convergencia, soló puede lograrse a partir de la experimentación.

#### IV.4.2.1 Implementación de la estrategia de Recocido Simulado

En la implementación del algoritmo de *Recocido Simulado* se utilizaron las rutinas **amebsa** y **amotsa** propuestas por Press *et al.* [Press, 1998]. Este proceso se basa en la estrategia *DownHill Simplex*, descrita en la sección IV.4.1, con las consideraciones siguientes:

- El número de configuraciones a evaluar $N$, donde se busca un equilibrio "térmico", esta dado por el número de vértices del *simplex*, $N \leftarrow N + 1$. Recuerde que $N$ es

el número de parámetros a optimizar.

- Los valores del vector de parámetros estan sujetos a los movimientos definidos por el *simplex* (reflexión, expansión y contracción). Por tanto, el *simplex* es el esquema de equilibrio térmico simulado por Metropolis. El factor de reducción de cada paso $\beta$ es substituido por estos movimientos.

- Por otro lado, el algoritmo de Metropolis se modifica de la siguiente manera:

  Se *suma* una variable de distribución aleatoria logarítmica positiva, proporcional a la temperatura $T$, a cada uno de los valores de la función objetivo para todos los vértices.

  Para cada nuevo valor de la función objetivo se resta una variable aleatoria construida de manera similar.

  Estas perturbaciones que escalan la temperatura $T$, simulan un movimiento térmico "Browniano", donde cada muestra nueva o vértice es aproximadamente aleatoria. Este movimiento equivale al proceso estocástico descrito en la ecuación IV.33. Así, al igual que Metropolis, el método acepta pasos de "descenso de colina" (downhill step) y algunas veces acepta pasos de "ascensos de colina" (uphill step).

  Adicionalmente, son dos las formas de "escapar" del estado de equilibrio. La primera, es a partir de la cota inferior de temperatura $\varepsilon$ y la segunda, por la longitud o número de pasos que se le permite ejecutar al *simplex*.

- La cota inferior de temperatura $\varepsilon$ está controlada por una tolerancia. La tolerancia se compara con la distancia del vector formado por el mejor y el peor vértice del *simplex*.

- El esquema de enfriamiento es similar al propuesto en el "lazo externo", descrito con antelación, para $\alpha = 0.8$.

A diferencia del algoritmo SA descrito por Schwefel (1995), el algoritmo propuesto por Press *et al.* (1998) requiere de dos parámetros controlados por el usuario: la temperatura de inicio $T$ y el número de pasos $I$ permitidos para el *simplex*.

**Rutina Recocido_Simulado_L**

La rutina **Recocido_Simulado_L**, controla todo el proceso de recocido simulado. Esta rutina se encarga de generar un *simplex* aleatorio de inicio dentro del rango de los valores permitidos para cada uno de los parámetros del modelo. Así mismo, controla el esquema de temperatura. Para asegurar que el proceso no quede en un lazo infinito, se define un cota máxima de pasos (5000) que puede ejecutar el *simplex*, sin importar la temperatura $T$ en la que se encuentre. **Recocido_Simulado_L** se comunica con las rutinas: **Modelo_L_amebsa**, encargada de la simulación modificada de Metropolis y, **Criterio_ch2_L** nuestro criterio de minimización. Un diagrama simple de interacción con otras rutinas se muestra en la Figura 45.

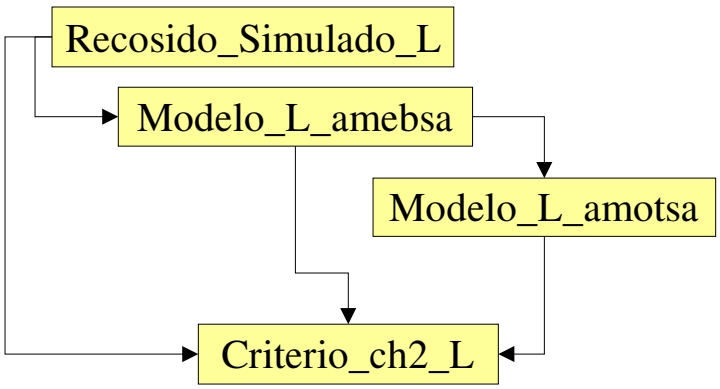



Figura 45: Rutinas utilizadas por **Recocido_Simulado_L**.

Las rutinas **Modelo_L_amebsa** y **Modelo_L_amotsa**, son similares a las rutinas originales **amebsa** y **amotsa** respectivamente, pero modificadas para soportar el criterio $\chi^2$ para funciones multidimensionales. Una descripción detallada de estos modificaciones se incluyen en la sección V.3.7. Las variables de entrada de **Recocido_Simulado_L** son:

**Datos[1,...,W][1,...,W]**. Ventana de la imágen original.
**W**. Tamaño de la ventana.
**signo_x, signo_y**. Signos en la dirección de la apertura de la esquina.
**P_m[1,...,N_par]**. Parámetros del modelo $M'_L(x, y, \vec{P})$ para su ajuste.
**N_par**. Número de parámetros a ajustar.
**ftol**. Tolerancia, equivalente a $\varepsilon = 1 \times 10^{-9}$.
**iteracion**. Número de iteraciones o longitud del paso de equilibrio, dado por el

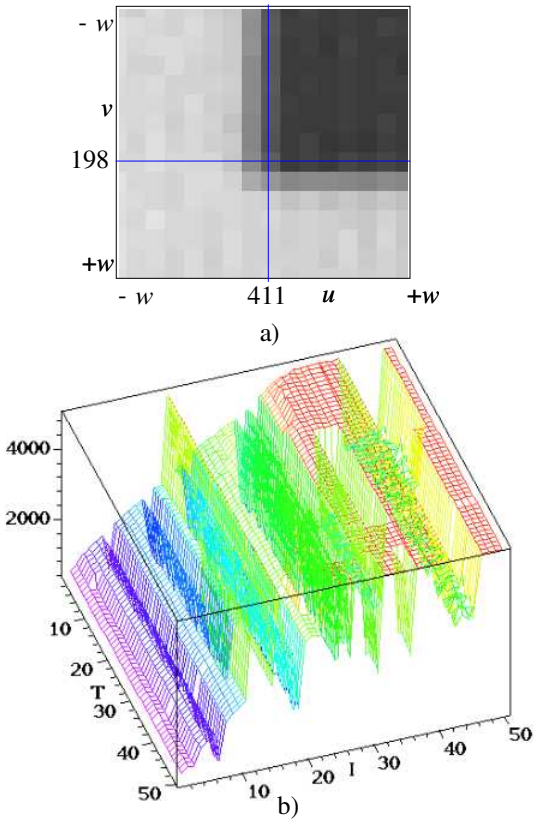



Figura 46: Parámetros de control para el algoritmo de *Recocido Simulado*. a) Ventana de una imagen real ubicada en las coordenadas de la imagen (411,198) y $w = 8$. b) Comportamiento de la Temperatura $T$ y de la Longitud del estado de Equilibrio $I$, con respecto al total de pasos.

usuario.

**temperatura**. Valor de la temperatura inicial $T$, dada por el usuario.

Dado que no se conoce *a priori* la magnitud de los valores de **iteracion** $I$ y **temperatura** $T$, se ejecutó el proceso SA variando $I$ y $T$ en el rango de $[1, 50]$ con incrementos de 1 ($50 \times 50$ veces). Este proceso se aplicó para obtener el punto de esquina de la estructura representada en la Figura 46.a. La Figura 46.a es una ventana de una imagen real adquirida con una cámara Pulnix 9701 en las coordenadas del sistemas de imagen (411,198) y una longitud de $2w+1 = 17$ pixeles por lado. Como resultado obtenemos una malla tridimensional, Figura 46.b, del comportamiento de estas variables con respecto al número total de ejecuciones de SA. A fin de visualizar mejor este comportamiento se

generan las gráficas, Figura 47 y Figura 48, que representan el perfil de la temperatura y la longitud de los pasos de equilibrio, para valores fijos de $I$ y $T$ respectivamente.

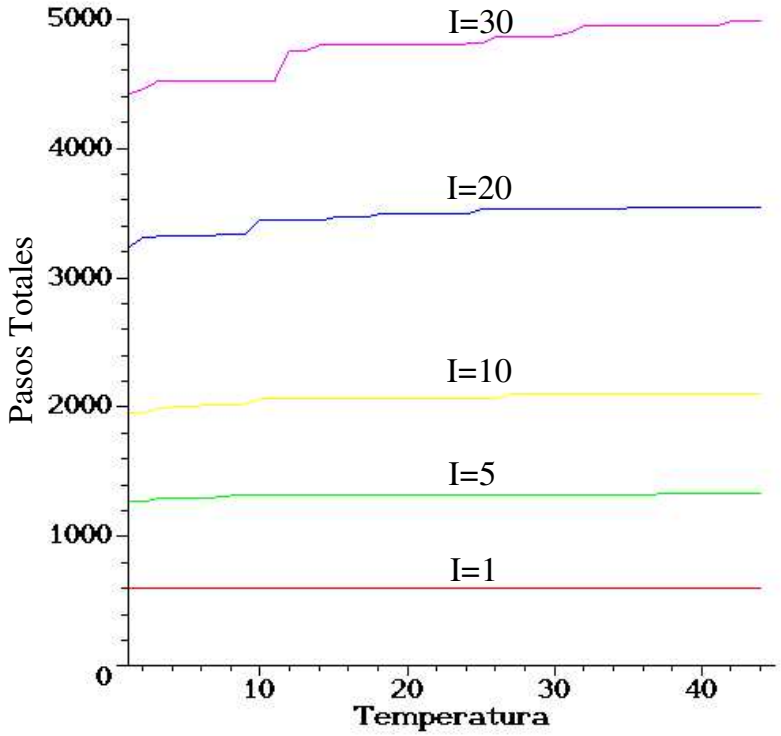



Figura 47: Número total de pasos para el algoritmo de *Recocido Simulado*. $T$ varía en el rango de $[1, 50]$ para los valores de $I = 1$, $I = 5$, $I = 10$, $I = 20$, $I = 30$. El rendimiento del proceso SA es *poco* sensible a la temperarura $T$.

Analizando las Figuras 46.b, 47 y 48 observamos que:

1. El rendimiento del proceso SA para el caso de nuestro modelo $M'_L(x, y, \vec{P})$, en relación con el número total de pasos para localizar los mejores valores de los $\vec{P}$ parámetros, es *poco* sensible a la temperatura de inicio $T$. Para un valor de $I$ constante, y $T$ variando de $[1, 50]$ el número total de pasos permanece constante. Por ejemplo, si $I = 1$ el número total de pasos es de aproximadament 600. Para el caso $I = 20$, los pasos totales son aproximadamente 3600, ver Figura 47. Esto se

debe a que $T$ representa una pertubación escalada por un logaritmo aleatorio, que puede verse como el "radio" de búsqueda de cada vértice del *simplex*. Sin embargo, este radio es corregido en el momento de evaluar la función objetivo $\chi^2$. Recuerde que el criterio $\chi^2$ implica una optimización local basada en el descenso de gradiente, ver sección IV.3.3.2.

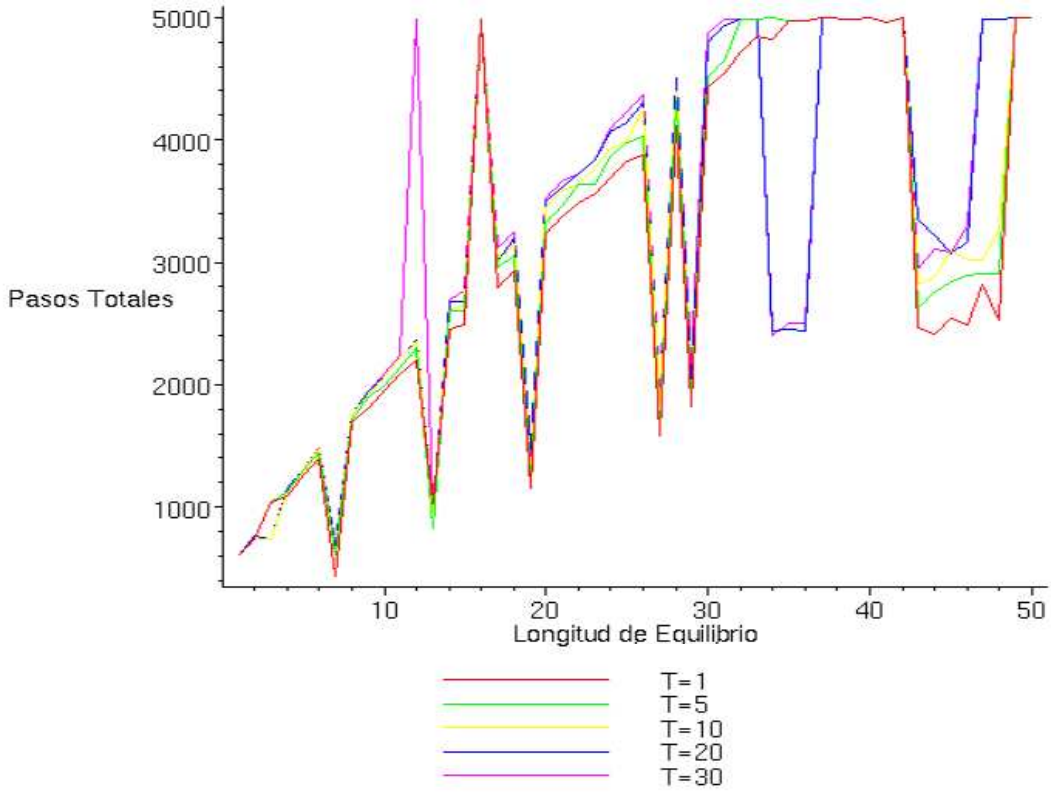



Figura 48: Número de pasos totales del proceso de *Recocido Simulado*, variando $I$ en el rango de $[1, 50]$, para los valores de $T = 1$, $T = 5$, $T = 10$, $T = 20$, $T = 30$.

2. En el caso de la longitud del estado de equilibrio $I$, utilizada en la estructura de la Figura 46.a, observamos que el número de ejecuciones del proceso de *Recocido Simulado* varía drásticamente en función del parámetro $I$. Las curvas mostradas en la Figura 48 muestran un comportamiento errático, reflejado en zonas con muchos extremos, zonas de linealidad y valles. Un ejemplo de este comportamiento errático, se observa en los valores de $I = 7$, $I = 13$ e $I = 19$, donde el número total de pasos del método SA es considerablemente inferior a su antecesor y predecesor más

próximo. Un caso extremo se encuentra cuando $I = 13$ y $T = 30$. El predecesor $I = 12$ reporta un total de 4992 pasos, el cual está cercano al máximo total de pasos permitidos, y en la siguiente longitud del estado de equilibrio $I = 13$ el número de pasos es de 975, el cual representa un mínimo local en esa vecindad. En partícular el mínimo inferior se localiza en $I = 7$ y $T = 1$, reportando 437 pasos totales. Por otro lado en algunos rangos, i.e., el mostrado en la zona comprendida entre $20 \leq I \leq 26$, el comportamiento es lineal. Por tanto, resulta difícil encontrar una relación entre la heurística propuesta por Metropolis, el criterio geométrico de Nelder y Mead y el efecto de descenso de gradiente de Levenberg-Marquardt.

Tomando en consideración estas observaciones el valor de la temperatura inicial es $T = 1$ y el número de pasos en el estado de equilibrio es $I = 7$, para la ejecución del proceso SA. En el Cápitulo de ***Resultados*** se muestra una tabla comparativa del rendimiento de este método con respecto a las otras dos estrategias de optimización global.

### IV.4.3 Estrategias Evolutivas

Desde la década de los 60s se ha incrementado el interés por imitar el proceso de evolución natural de la vida, gracias al continuo desarrollo de las computadoreas (velocidad de proceso y capacidad de almacenamiento). Este proceso evolutivo se fundamenta en dos grandes estudios: 1) la *Teoría de Evolución de las Especies*, postulada por Charles Darwin en 1859, la cual se basa en la supervivencia de la especie que mejor se adapte a su medio ambiente, y 2) en la mecánica de evolución postulada por diversos principios biológicos extraidos de la Genética y la Biología [Gen, 1996] [Holland, 1992] [Koza, 1992] [Michalewicz, 1992] [Olague, 1994]. La simulación de estas teorías ha traido como resultado el desarrollo de algoritmos y técnicas de optimización numérica nuevas y poderosas [Gen, 2000]. Al conjunto de estas técnicas se les conoce, hoy en día, como *Computación Evolutivia* (EC[2]) . En la literatura especializada, son 4 las divisiones más representativas en EC que se conocen: 1) *Algoritmos Genéticos* (GA), propuesto por Holland [Holland, 1992], creador del *Teorema Fundamental* donde demuestra matemáticamente la factibilidad de estas técnicas; 2) *Estrategías Evolutivas* (ES[3]), desarrolladas por Rechenberg [Rechenberg, 1973] y Schwefel [Schwefel, 1995], que centran su atención en la forma

[2]por sus siglas en inglés *evolutionary computation*.

[3]por sus siglas en inglés *evolutionary strategies*.

de explotar el proceso evolutivo (selección, apareamiento, mutación y auto-adaptación); 3) *Programación Evolutiva* (EP[4]), desarrollado por Fogel *et al.* [Fogel, 1966], que aplica estos conceptos a una máquina de estado finito o autómatas, y por último ; 4) *Programación Genética* (GP), propuesto por Koza [Koza, 1992], que propone la explotación de una estructura de árbol aplicando estos elementos.

El proceso de selección natural postula "que sólo el individuo dentro de una población que se adapte mejor a los cambios de su medio ambiente, es el que sobrevive". En este sentido, Rechenberg propone la hipótesis "que el método de evolución orgánica representa la estrategia óptima para la adaptación y supervivencia de cosas dentro de su ambiente", y concluye, "por tanto, esto debiera tener validez mientras se tomen los principios biológicos de la evolución para la optimización de sistemas".

#### IV.4.3.1 Algoritmo Básico

El algoritmo genético es una técnica estocástica, basada en el mecanísmo de selección orgánica e incluye en su terminología diversos principios biológicos. En general, son 5 los componentes básicos de este algoritmo y Olague [Olague, 1994] los sintetiza en la forma siguiente:

1. Una *representación genética* de la solución del problema.
2. Una forma o técnica de crear una población inicial de las soluciones.
3. Una *función de evaluación* o clasificación de las soluciones en términos de su *aptitud*.
4. *Operadores genéticos* que alteren la composición genética de los hijos durante la reproducción.
5. Valores de los parámetros del algoritmo genético.

A partir de estos elementos el algoritmo genético se encarga de mantener a la población de los individuos, en cada una de las generaciones. Primero, se genera un conjunto aleatorio de soluciones llamada *población*. Cada individuo en la población es llamado un *cromosoma*, que representa por él mismo una solución del problema. El cromosoma

[4]por sus siglas en inglés *evolutionary programming*.

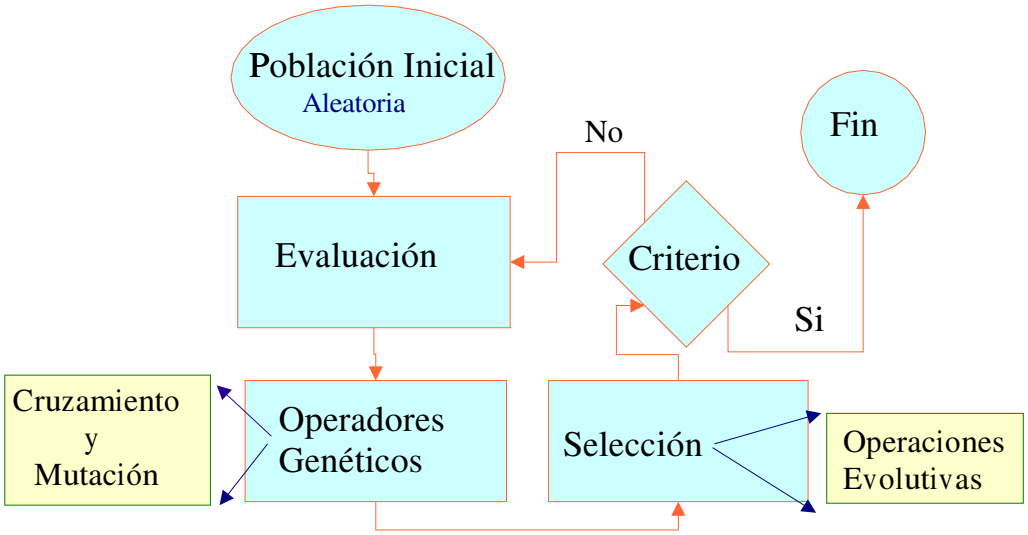



Figura 49: Diagrama básico de un *Algoritmo Genético*.

evoluciona a través de iteraciones sucesivas llamadas *generaciones*. Durante cada generación, el cromosoma es *evaluado*, empleando su valor de *aptitud*. Para crear nuevos cromosomas llamados *descendencia* y producir nuevas genereraciones, existen dos operadores genéticos; *mutación* que es el proceso de generar un individuo nuevo a partir de la modificación de la estructura de uno ya existente y *apareamiento* que genera dos individuos nuevos a partir de dos padres mediante el intercambio o cruzamiento de parte de la estructura genética de cada padre, *intercambio genético*. Finalmente se seleccionan aquellos que se encuentren mejor adaptados al medio, en base a su valor de aptitud o en base a diferentes procesos estocásticos. A estos procesos se les conoce como *operaciones evolutivas*. Después de varias generaciones el proceso converge a la mejor solución, que representa el ***óptimo*** o ***subóptimo*** del proceso. En la Figura 49 se presenta en forma esquemática este proceso.

***Representación Genética.***

El proceso de **Codificación** se refiere a como *representar o codificar* una solución de un problema dentro de un cromosoma y como *decodificar* un cromosoma en términos de la solución del mismo. En este sentido, un algoritmo genético requiere de dos espacios: Un ***espacio de soluciones*** donde cada elemento es llamado *fenotipo*; y un ***espacio***

***codificado*** llamado *genotipo*. Un cromosoma es entonces el *genotipo* de una solución del problema. En general, el algoritmo trabaja en el espacio de soluciones y en el espacio codificado alternativamente. Los operadores genéticos, mutación y apareamiento, trabajan en el genotipo del problema y el proceso evolutivo, selección y evaluación en el espacio de soluciones. La codificación se ha investigado bajo diferentes aspectos, los cuales son:

1. *Clasificación de la Codificación.*

   Dependiendo del tipo de dato que contiene un cromosoma, la codificación se clasifica en:

   - *Binaria.* Un cromosoma es representado por una una cadena de "1" y "0". Este tipo de representación no consume muchos recursos de memoria y sus operaciones no son costosas, sólo requiere de manipulaciones binarias sobre variables binarias. Sin embargo, se necesita encontrar una función que nos permita efectuar una correspondencia entre el espacio de soluciones y su representación genética. Adicionalmente, se requiere una segunda función que nos permitar regresar al espacio de soluciones. A este último proceso se le conoce como *morfogénesis*. La búsqueda de estas funciones podría resultar una tarea complicada. Los GA utilizan generalmente la representación binaria, vea por ejemplo los trabajos de Holland y algunos ejemplos propuestos por Gen y Cheng [Gen, 1996].
   - *Entero o Alfabético.* Un cromosoma es representado por una cadena de números enteros o alguna letra del alfabeto. Para problemas de optimización puramente combinatorios ésta es considerada la mejor representación genética. En esencia el problema combinatorio busca la mejor permutación o combinación de los elementos sujetos a ciertas restricciones, por lo tanto esta codificación es la mejor forma de representar el problema.
   - *Reales.* Cada cromosoma se representa como un vector de números reales. Este tipo de representación es empleado ampliamente en problemas de optimización numérica [Gen, 1996], dado que el proceso de morfogénesis no es necesario, esto es, el espacio codificado y de soluciones es el mismo. En nuestro desarrollo se utiliza la representación del cromosoma con números reales.

- *Estructura de Datos.* Cada cromosoma representa una o más estructuras de datos. Un ejemplo de este concepto se observa en el trabajo de Koza (Programación Genética) [Koza, 1992], donde cada cromosoma representa una estructura de árbol; las hojas se definen como un conjunto de *terminales* (constantes) unidas entre si por un conjunto de *funciones* (operadores boleanos, aritméticos, trascendentales, condicional o de transferencia).

Dependiendo del contenido de la codificación del cromosoma, se clasifican en:

- *Sólo-la-Solución.* El cromosoma sólo contiene información de la solución del problema. La gran mayoria de los GA están inmersos en esta clasificación, en particular, el algoritmo genético mostrado en el presente trabajo pertenece a este tipo.
- *Solución + Parámetros.* Cada cromosoma consiste de dos partes: la primera está dada por la solución del problema y la segunda está dada por los parámetros de la estrategia evolutiva, principalmente la varianza y covarianza de la distribución normal para la mutación. Esta técnica es introducida por Rechenberg y Schwefel [Schwefel, 1995], en las estrategias evolutivas. El propósito de incorporar parámetros de estrategia dentro de la representación genética de los individuos, es la facultad de auto-adaptación evolutiva de estos parámetros mediante la aplicación de operadores evolutivos [Gen, 2000].

2. *Ilegalidad e infactibilidad de la codificación.*

   En un algoritmo genético, los operadores de mutación y apareamiento trabajan directamente en el genotipo del problema. Dichos operadores manipulan directamente el código de la estructura de los cromosomas, mediante ciertos procesos estocásticos. Por otro lado, el proceso evolutivo opera directamente en el espacio de la solución, evaluando y seleccionando los cromosomas que se adapten mejor al medio ambiente del problema, a través de funciones de aptitud y heurísticas de clasificación. Es por ello, que la correspondencia entre el genotipo y el fenotipo a partir de funciones de codificación y decodificación de ambos espacios tiene una considerable influencia en el rendimiento de un algoritmo genético. Asi mismo, debe distiguirse entre dos conceptos; *ilegalidad* e *infactibilidad.*

El término *infactibilidad* se refiere al fenómeno ocurrido cuando la solución decodificada a partir de un cromosoma, morfogénesis, no pertenece a una región factible en el espacio de soluciones, pero sí al universo de soluciones, ver Figura 50. En problemas de optimización con restricciones, el óptimo ocurre típicamente en el límite definido por el espacio factible y el infactible, en cuyo caso, el empleo de *estrategias de penalidad* es una técnica empleada para solventar este problema. La aplicación de la *penalidad* forza a la búsqueda genética a probar la existencia del óptimo en esta región.

El término *ilegalidad* se referie al fenómeno producido por la morfogénesis cuando el cromosoma no representa una solución, ver Figura 50. Este problema es originado por la naturaleza del proceso de codificación. Una forma básica de evitarlo, es replanteando las ecuaciones de codificación y decodificación de las soluciones. No obstante, existen las técnicas llamadas *estrategias de reparación* de cromosomas como una herramienta adicional para atacar este fenómeno. En el libro escrito por Gen y Cheng [Gen, 1996] ambas estrategias son explicadas con detalle.

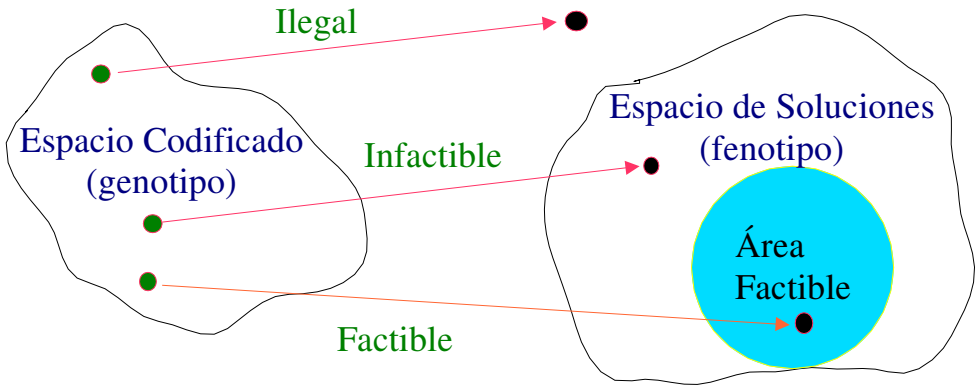



Figura 50: *Ilegalidad* e *Infactivilidad* de la representación genética.

3. *Correspondencia de la codificación*

La correspondencia entre el espacio de soluciones y el espacio codificado puede ser de 3 tipos, ver Figura 51:

- $1 - a - 1$
- $n - a - 1$

- $1 - a - n$

La correspondencia $1 - a - 1$ es el caso más deseable de la morfogénesis en un algoritmo genético, no existe ambigüedad en la representación y no hay operaciones triviales en el momento de generar descendencia. A esta característica se le conoce como la propiedad de *unicidad* o *no-redundancia* de la representación.

Para el caso de $n - a - 1$, dos o más cromosomas estructuralmente distintos, tienen la misma representación con respecto a su fenotipo. Esto provoca que el número de individos en cada espacio sea distinto, por tanto el universo de selección, *diversidad* genética, se puede reducir significativamente. El efecto final sobre el algoritmo es una convergencia prematura que no alcanza el óptimo global.

El último caso, $n - a - 1$, es el peor caso para estos algoritmos. Dado que un cromosoma representa $n$ soluciones, resulta imposible discernir que cromosoma representa la mejor solución, en otras palabras, un cromosoma tiene distintos valores de la función de aptitud, por consiguiente no se podra obtener una solución. En esta situación es preferible replantear el proceso de morfogénesis del algoritmo.

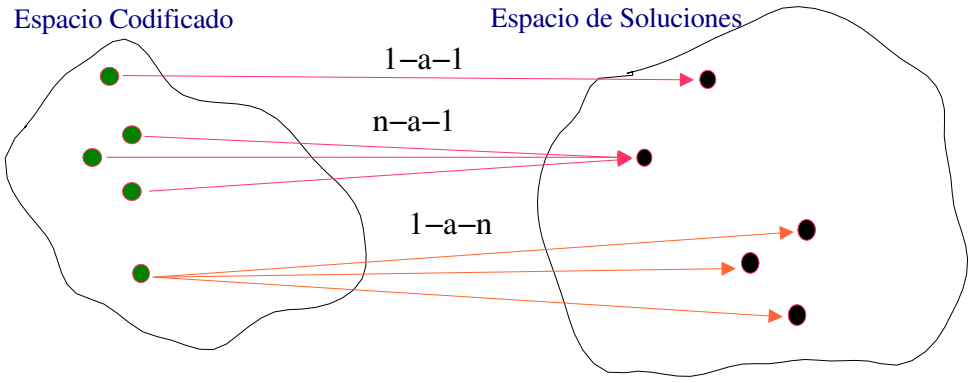



Figura 51: *Correspondencia* entre el espacio codificado y el espacio de soluciones.

### *Operadores Genéticos.*

La forma de generar nuevos individuos dentro de cada generación esta controlada por dos operadores genéticos que son el *apareamiento* y la *mutación*. Primeramente, explicaremos

cual es la estructura de un cromosoma. Un cromosoma es la representación genética de una solución del problema definida mediante una estructura de la forma siguiente:

$$x = [x_1, x_2, \ldots, x_l, \ldots, x_n]$$

donde $n$ es el tamaño o longitud del cromosoma que permanece constante durante todo el proceso evolutivo. $x_1, x_2, \ldots, x_l, \ldots, x_n$ se conocen como *genes* y representan a la unidad básica de la estructura del cromosoma, por tanto un cromosoma es un conjunto finito de genes. $x_l$ puede representar un número binario, entero, alfabético o real, vea la sección de ***Representación Genética***. Un *alele* es un subconjunto no vacío de genes de $x$ que representa una característica de la solución y es la unidad básica de información en el fenotipo del problema. Un alele guarda siempre su posición y número de elementos dentro del cromosoma y son subconjuntos disjuntos entre si. Un cromosoma entonces, puede contener uno o más aleles. Por ejemplo si $x$ es la representación binaria de las variables de una función bidimensional $f(a, b)$ y consideramos que $a$ se representa con $r$ elementos binarios y $b$ con $s$ elementos binarios, entonces $x$ contiene dos aleles $a$ y $b$ con la estructura siguiente:

$$x = [\underbrace{x_1, x_2, \ldots, x_r}_{a}, \underbrace{x_{r+1}, x_{r+2}, \ldots, x_{n=r+s}}_{b}]$$

Un paso inicial para la aplicación de las técnicas evolutivas es la determinación de la longitud total de $x$, la cual depende de la complejidad de la representación de la solución del problema.

El ***apareamiento***, también llamado *recombinación*, consiste en la generación de dos individuos nuevos a partir de dos padres. Si representamos a un par de cromosomas como:

$$x = [x_1, x_2, \ldots, x_n]$$
$$y = [y_1, y_2, \ldots, y_n]$$

donde $x$ y $y$ son dos cromosomas con $n$ elementos. Si el apareamiento es en un sólo punto despúes de la *k-ésima* posición, donde $k$ es un número aleatorio generado en el rango $[1, n]$, los descendientes o hijos estarán expresados como sigue:

$$x' = [x_1, x_2, \ldots, x_k, y_{k+1}, y_{k+2}, \ldots, y_n]$$
$$y' = [y_1, y_2, \ldots, y_k, x_{k+1}, x_{k+2}, \ldots, x_n]$$

La forma de apareamiento descrita anteriormente es empleada ampliamente en los algoritmos con representación binaria, entera o alfabética y es conocida como cruzamiento **convencional** en *un punto de corte*. Este estilo de apareamiento puede ser ampliado a *dos puntos de corte* o *multiples puntos de corte*. Existen otros tipo de apareamiento como son:

- *Apareamiento aritmético.* Propuesto por Gen y Cheng [Gen, 1996], el apareamiento esta basado en el concepto de combinaciones lineales de vectores a partir de la teoría de espacios convexos. Esta técnica se explicará en más detalle en la sección IV.4.3.2 ya que es el empleado en el desarrollo de nuestro algoritmo evolutivo para la obtención de los parámetros de $M'_L(x, y, \vec{P})$.

- *Apareamiento direccional.* Este estilo de apareamiento emplea los valores de la función objetivo para determinar la dirección de la búsqueda genética y únicamente produce un descendiente a partir de dos cromosomas padres [Michalewicz, 1992].

El operador de ***mutación*** altera el valor de uno o más genes con una cierta probabilidad. Para la representación binaria el proceso es simple; se selecciona aleatoriamente un conjunto de genes de toda la población, dichos genes cambian su valor, si el valor del gen es "1" se reemplaza por "0" y viceversa. Para la representación con números reales el procedimiento es similar. Supongase que el cromosoma $x = [x_1, x_2, \ldots, x_n]$ es seleccionado para mutación. Obtenemos un número aleatorio $k \in [1, n]$ y producimos entonces un descendiente $x' = [x_1, \ldots, x'_k, \ldots, x_n]$, tal que $x'_k$ es un valor aleatorio uniformemente distribuido en el rango $[x^I_k, x^S_k]$. $x^I_k$ y $x^S_k$ son los valores de la cota inferior y superior de la *k-ésima* variable respectivamente, generalmente dadas por el dominio del problema. Por otro lado existen otros tipos de mutación como son:

- *Mutación dinámica.* Este operador es conocido como *mutación no uniforme*, y fue propuesto por Michalewicz [Michalewicz, 1992]. Una mutación dinámica es diseñada con el propósito de lograr una "sintonía fina" con objeto de obtener precisiones altas en el proceso evolutivo. En secciones posteriores se explicará con más detalle este tipo de mutación ya que es el empleado en el algoritmo genético del presente trabajo.

- *Mutación direccional.* Al igual que en el apareamiento direccional, la mutación direccional toma en cuenta el gradiente de una función objetivo y con la cual se

define el valor de la heurística del cambio. Esta técnica fue propuesta por Gen y Cheng [Gen, 1996].

En un proceso de optimización, los operadores evolutivos explotan exhaustivamente el espacio de *búsqueda* del dominio del problema. Típicamente existen dos tipos de búsqueda: la *búsqueda aleatoria* explora todo el espacio de soluciones y tiene la capacidad de "escapar" de un óptimo local. Por otro lado, la *búsqueda local* explota la mejor solución y es capaz de ascender o descender rápidamente hacia un valle o colina para obtener un óptimo local. Estas dos habilidades son mutuamente complementarias en un proceso de búsqueda. Un algoritmo genético es entonces, una clase de método de búsqueda de propósito general que combina elementos de búsqueda direccional y estocástica, y puede efectuar un buen balance entre la exploración y explotación del espacio de soluciones.

El operador de apareamiento es usado para explotar el espacio de búsqueda. Holland propone la hipótesis que a partir de dos padres con un valor de aptitud mayor que los otros integrantes de la población, se produce una descendencia igual o superior. Esta teoría está basada en el análisis de esquemas realizado por Holland y es la base que sustenta la convergencia de este tipo de algoritmos. Una explicación clara en español de este concepto puede consultarse en [Olague, 1994]. En algoritmos genéticos convencionales enfocados a la optimización, el cruzamiento es el principal operador del sistema evolutivo y por ende se le asigna un valor de probabilidad relativamente alto. Por otro lado, el operador de mutación provoca cambios espontáneos en la composición genética de los cromosomas, lo que implica que explore otras áreas del espacio de búsqueda. Sin embargo, está confirmado que la mutación puede algunas veces jugar un papel más importante que el apareamiento en una búsqueda genética [Schwefel, 1995].

***Selección.***

El proceso de selección se encarga de simular los postulados Darwinianos de la evolución orgánica. La selección es la fuerza de conducción del algoritmo genético y la *presión* que ejerce sobre la evolución generacional, es critica en el proceso evolutivo. En un extremo, si la presión es alta, la búsqueda puede terminar prematuramente, por otro lado, si la presión es baja el progreso del algoritmo puede ser más lento del necesario. Generalmente se emplea una presión baja al iniciar el algoritmo. Esto permite que se explore en forma amplia el espacio de soluciones. Una presión alta es recomendable

al final del proceso, con objeto de explotar exhaustivamente la región más promisoria en el espacio de búsqueda. La selección es empleada en dos procesos: para seleccionar los individuos candidatos a mutación y apareaminento y para seleccionar los individuos que pertenecerán a la siguiente generación. Existen una gran variedad de métodos de selección, los más comunes son:

- *Selección por Ruleta.* Propuesto por Holland, es uno de los métodos de selección *estocástica* empleado más comúnmente en GA. La idea principal es determinar la probabilidad de supervivencia de cada individuo de forma proporcional a su valor de aptitud. En esta estrategia se construye una *rueda de ruleta* segmentada. Cada segmento de la rueda es propocional al valor de aptitud del cromosoma, y el número de segmentos es igual al *tamaño* o *longitud* de la población. Se gira la rueda, y se "lanza" un "dardo" para seleccionar un cromosoma. Existe una mejor probabilidad de seleccionar un cromosoma con un valor de apitud mayor. En otras palabras, con una área mayor en el segmento de la rueda de la ruleta, se tiene una mejor probabilidad de selección que otro cromosoma con valor menor en su función de mérito. La rueda se gira tantas veces como número de individuos hay en una población. En cada paso, el cromosoma seleccionado formará parte de la nueva generación. En este método el universo de selección es constante, donde los padres seleccionados para apareamiento o mutación son reemplazados por sus hijos. A esta estrategia de selección se le conoce como *reemplazo generacional.*

- *Selección $(\mu + \lambda)$ y $(\mu,\lambda)$.* Propuesto por Bäck [Bäck, 1994], es un proceso *determinístico* donde los mejores individuos, padres y descendencia, son seleccionados para formar a la nueva generación, a partir de su valor de la función de mérito o aptitud. Para el caso de $(\mu + \lambda)$, los $\mu$ padres y los $\lambda$ descendientes compiten para sobrevivir y los mejores $\mu$, de todos los padres más su descendencia, formarán a la nueva generación. $(\mu,\lambda)$ seleciona los mejores $\lambda$ descendientes, como individuos de la nueva generación a partir de los $\mu$ padres. En estos métodos determinísticos se prohibe la selección de cromosomas duplicados de la población. Los criterios determinísticos más comunes son:

  *Truncamiento.* Se ordenan todos los cromosomas tomando como base su valor de aptitud. Se define un umbral $T$ que representa un porcentaje de los mejores.

Seguido, sólo se seleccionan los mejores $T$ cromosomas para pasar a la siguiente generación.

*Elitismo.* Únicamente los mejores cromosomas pasan a formar parte de la próxima generación.

*Reproducción de estado-equilibrado.* Consiste en reemplazar los peores $\lambda$ cromosomas de la población original por los $\mu$ descendientes. Esta estrategia es una modificación del reemplazo generacional.

- *Selección por Torneo.* La idea general de esta estrategia es escoger aleatoriamente un conjunto de cromosomas para competir en un torneo. Por ejemplo, se substraen los mejores cromosomas para su reproducción empleando el método de la ruleta, los cuales serán los integrantes del torneo. Al número de cromosomas escogidos para el torneo se le llama *longitud del torneo*. Este método *híbrido*, combina elementos estocásticos y determinísticos, como los descritos en la selección por ruleta y la selección $(\mu + \lambda)$ respectivamente.

- *Selección por escalamiento y ranking.* En las estrategias anteriores la selección o probabilidad de selección de un individuo es proporcional a su valor de aptitud. Este esquema puede provocar características no deseadas. Después de varias generaciones existe la tendencia a generar *supercromosomas*, que dominan el proceso de selección y por tanto se pierde la *diversidad genética* de nuestra población. Para minimizar estos problemas se han propuesto los mecanísmos de *escalamiento y ranking*. El escalamiento efectúa una correspondencia entre el valor de la función objetivo hacia un valor positivo real. La probabilidad de supervivencia será tomada a partir de este valor. En general, si $f'_k$ es la función de escalamiento basada en una función objetivo original $f_k$ para un cromosoma $k$, el escalamiento se expresa como:

$$f'_k = g(f_k) \tag{IV.34}$$

donde $g(\cdot)$ es la función de tranformación de función de aptitud original dentro de la aptitud escalada. La función $g(\cdot)$ puede tomar distintas formas, tales como: *escalamiento lineal*, *escalamiento dinámico lineal*, *truncamiento en sigma*, *escalamiento en potencias*, *escalamiento logarítmico*, *por ventana*, *normalizada* y *escalamiento basado en Boltzmann*. Los parámetros de escalamiento de estos métodos son dependientes del problema.

La selección por ranking ignora el valor actual de la función objetivo. En su lugar emplea una clasificación definida por la posición que guardan los cromosomas en la población, para determinar la probabilidad de sobrevivencia. Existen dos métodos comúnmente usados; *ranking lineal* y *ranking exponencial* propuestos por Baker y Michalewicz respectivamente. Una buena revisión bibliográfica del escalamiento y ranking puede ser consultada en [Gen, 1996].

En resumen, un proceso evolutivo es una heurística que contiene elementos estocásticos y determinísticos, que intenta lograr un buen balance entre la explotación y exploración del espacio de búsqueda, definido por el dominio del problema a optimizar, a través de los principios biológicos de la evolución orgánica. La Figura 52 sintetiza la estructura general de un GA.

#### IV.4.3.2 Implementación de la estrategia evolutiva

En la teoría de algoritmos evolutivos existen diferentes formas de representar a cada uno de los individuos, ver sección IV.4.3.1, y su manipulación mediante operadores genéticos (cruzamiento y mutación) para producir nuevas generaciones. Michalewicz (1992) argumenta que la mejor manera de resolver un problema es buscar la representación más apropiada que se asemeje a las operaciones especificas del comportamiento del fenómeno. Nosotros empleamos una representación en **números reales** (cada cromosoma es codificado como un vector de números reales $\vec{y}_i$) y desarrollamos una forma de como llevar a cabo un conjunto de transformaciones con una simple operación algebraica. Estas características y las que a continuación se describen están codificadas en la rutina **Evolutivo_L**.

**Apareamiento**

Muchos operadores de cruzamiento han sido introducidos dentro del nombre de aritmética de operadores, los cuales han sido diseñados tratando de imitar el concepto de combinaciones lineales de vectores en el campo de la "teoría de espacios convexos[5]". Los operadores de cruzamiento son combinados para producir descendencia, mediante el cálculo del peso promedio de 2 vectores $\vec{y}_1$ y $\vec{y}_2$ como sigue:

$$\begin{aligned} \vec{y}_1' &= \lambda_1 \vec{y}_1 + \lambda_2 \vec{y}_2 \\ \vec{y}_2' &= \lambda_2 \vec{y}_1 + \lambda_1 \vec{y}_2 \end{aligned} \tag{IV.35}$$

[5]en inglés convex sets theory.

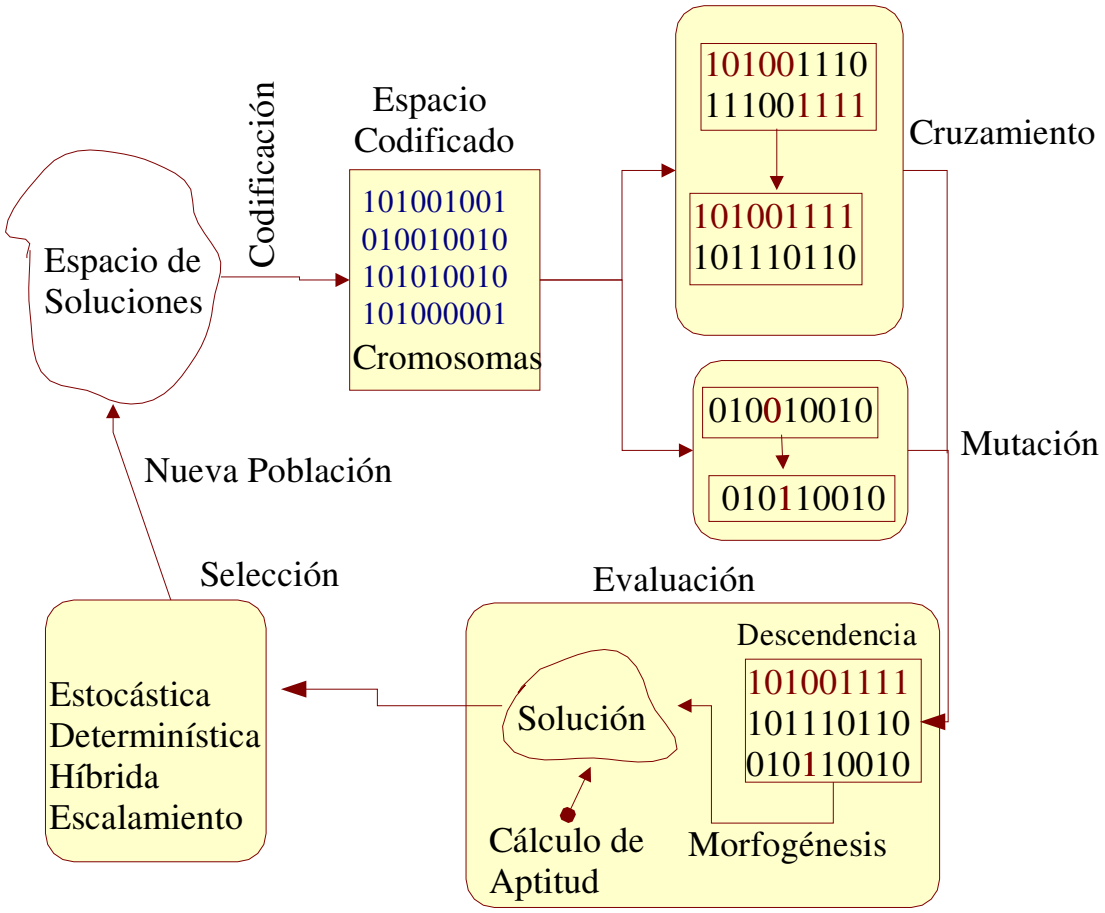



Figura 52: Estructura general de un algoritmo genético.

si los multiplicadores están restringidos a:

$$\lambda_1 + \lambda_2 = 1, \quad \lambda_1 > 0, \quad \lambda_2 > 0$$

la expresión es conocida como una combinación convexa. Por otro lado, si la condición de no negatividad en los multiplicadores es respetada, se le conoce como una combinación afin y si los multiplicadores siempre están contenidos en el espacio real, la combinación es conocida como una combinación lineal [Gen, 1996].

Existen diferentes tipos de combinaciones convexas, las cuales dependen de la forma en que se generan $\lambda_1$ y $\lambda_2$. Por ejemplo, en una "combinación promedio" $\lambda_1 = 0.5$ y $\lambda_2 = 0.5$, en una "combinación afin" $\lambda_1 = 1.5$ y $\lambda_2 = -0.5$. En particular, nosotros empleamos

una *combinación intermedia afin extendida*, propuesta por Mühlenbein y Schlierkamp-Voosen, ver [Gen, 1996]. En la cual, $\lambda_1$ es un número aleatorio en el intervalo $[-d, 1+d]$, tal que $\lambda_1 + \lambda_2 = 1$. El algoritmo evolutivo se programó para $d = 0.5$.

**Mutación**

Para efectos de generar descendientes que busquen aleatoriamente soluciones independientes a la convergencia que impone el proceso de selección, nosotros empleamos el concepto de "mutación dinámica" introducido por Janikow y Michalewicz [Michalewicz, 1992], el cual tiene la capacidad de producir ajustes finos que nos ayudan a obtener precisiones altas. Las mutaciones dinámicas están expresadas como sigue: dado un padre $\vec{y}$, si el *k-ésimo* elemento $y_k$ es seleccionado para una mutación, el descendiente generado es $\vec{y'} = (y_1, \ldots, y'_k, \ldots, y_n)$. Donde $y'_k$ es un número aleatorio seleccionado a través de dos posibilidades:

$$\begin{array}{lcl} y'_k & = & y_k + \Delta(t, y_k^U - y_k) \\ y'_k & = & y_k - \Delta(t, y_k - y_k^L) \\ \Delta(t,y) & = & yr\left(1 - \frac{t}{T}\right)^b \end{array} \qquad \text{(IV.36)}$$

donde $y_k^U$ y $y_k^L$ son las cotas superior e inferior del *k-ésimo* elemento respectivamente. $r$ es un número aleatorio en el rango de $[0, 1]$. $t$ es la generación actual. $T$ es el número máximo de generaciones permitidas. $b$ es el parámetro que determina el grado de no uniformidad en la función $\Delta(t, y)$. La función $\Delta(t, y)$ regresa un valor en el rango de $[0, y]$ tal que el valor de $\Delta(t, y) \to 0$ cuando $t$ crece. Esto causa que en las generaciones iniciales ($t$ pequeño) el operador de mutación busque uniformemente en el espacio de soluciones y explote exhaustivamente la región más promisoria al final del proceso.

**Transformación afin sobre operadores genéticos**

Los operadores de mutación y cruzamiento pueden ser encapsulados dentro de una simple transformación compleja. Esta idea fue propuesta por Olague (1999), la cual consiste básicamente en lo siguiente: aplicando las propiedades geométricas de planos afines, si suponemos que $\Re_a$ es una representación de un plano *afin*, todas las transformaciones de la clase entera $\Re_A$ son derivadas aplicando transformaciones del tipo:

$$\begin{array}{lcl} Y'_1 & = & b_{11}Y_1 + b_{12}Y_2 + C_1 \\ Y'_2 & = & b_{21}Y_1 + b_{22}Y_2 + C_2 \end{array} \qquad \text{(IV.37)}$$

donde cada una de la líneas representan una ecuación invariante y $Y = \left(\frac{y_1}{y_0}, \frac{y_2}{y_0}\right)$ suponiendo $y_0 = 0$ es un par de coordenadas no homogéneas que representan el punto actual en el plano afin, y los coeficientes son números reales arbitrarios con la condición $|b_{rs}| \neq 0$, entonces para el caso en dos dimensiones todas las $n$ variables (aleles de los cromosomas) las podemos representar como:

$$\begin{pmatrix} Y'_{1_1} & Y'_{1_2} & \dots & Y'_{1_n} \\ Y'_{2_1} & Y'_{2_2} & \dots & Y'_{2_n} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & C_1 \\ \underbrace{b_{21} \quad b_{22}}_{crcuzamiento} & & \underbrace{C_2}_{mutación} \end{pmatrix} \begin{pmatrix} Y_{1_1} & Y_{1_2} & \dots & Y_{1_n} \\ Y_{2_1} & Y_{2_2} & \dots & Y_{2_n} \\ 1 & 1 & \dots & 1 \end{pmatrix} \tag{IV.38}$$

La Ecuación (IV.38) puede ser expandida para aplicarse a la población completa. Las ventajas de esta representación son:

- El tratamiento de todas las transformaciones son estándares.
- Transformaciones complejas implican simples multiplaciones de matrices.
- En lugar de aplicar transformaciones sucesivas a todos los puntos de un escenario tridimensional, sólo se puede calcular la matriz de transformación total, misma que se multiplica con las coordenadas homegéneas de cada punto.
- Inversiones en las transformaciones son realizadas mediante inversiones de matrices.
- Cálculo rápido. Operaciones matriciales están soportadas en el hardware de estaciones de trabajo con alto rendimiento gráfico.

**Selección**

La rutina evolutiva **Evolutivo_L** está diseñada con dos métodos de selección: *selección por ruleta* y *selección por torneo.* Ambos métodos se integran para aumentar la presión en la convergencia de la población. Esto es, se emplea el método de selección por ruleta para escoger a los dos cromosomas que competirán en un torneo. Despúes, se escoge el cromosoma con un mejor valor en su función de aptitud. Recuerde que la función de aptitud es nuestro criterio $\chi^2$. Combinando ambos métodos obtenemos una selección estocástica, selección por ruleta, y determínistica, selección por torneo. Así mismo, se emplea el *reemplazo generacional* en la obtención de la nueva población. El algoritmo general de selección por ruleta es el siguiente:

1. Se calcula el valor de aptitud $v_k$ para cada cromosoma $\vec{y}_k$:

$$v_k = Q(\vec{y}_k)$$

2. Se calcula la aptitud total de la población:

$$a = \sum_{k=1}^{pop} v_k$$

donde *pop* es el número total de individuos en la población.

3. Se calcula la probabilidad de selección $p_k$ para cada cromosoma $\vec{y}_k$:

$$p_k = \frac{v_k}{a}, \qquad k = 1, 2, \ldots, pop$$

Esta expresión implica que mientras mayor sea el valor de aptitud de un cromosoma, éste tendrá una mejor probabilidad de ser seleccionado y por tanto contenga mayor área en la construcción de la ruleta. Para problemas de maximización este criterio es suficiente. Sin embargo, en nuestro problema requerimos minimizar la función objetivo $Q$. Esto es, necesitamos que valores pequeños de $v_k$ ocupen una mayor área en la rueda de la ruleta. Para logar esto, nosotros invertimos el valor de probabilidad $p_k$ como sigue:

$$p'_k = p_k^{-1} = \frac{a}{v_k}$$

y normalizamos a uno el nuevo valor $p'_k$

$$\begin{aligned} a' &= \textstyle\sum_{k=1}^{pop} p'_k \\ p''_k &= \frac{p'_k}{a'} \qquad k = 1, 2, \ldots, pop \end{aligned}$$

donde $p''_k$ es el inverso normalizado de la probabilidad $p_k$.

4. Se calcula la probabilidad acumulada $q_k$ para cada cromosoma $\vec{y}_k$:

$$q_k = \sum_{j=1}^{k} p''_k, \qquad k = 1, 2, \ldots, pop$$

$q_k$ representa la rueda de la ruleta.

5. La rueda de ruleta se hace girar tantas veces como *pop* individuos existan en una generación. En cada una de los giros se selecciona un cromosoma en la forma siguiente:

(a) Se genera un número aleatorio $r$ en el rango de $[0, 1]$.

(b) Si $r \leq q_1$ entonces el primer comosoma $\vec{y}_1$ es seleccionado; de otra forma, se selecciona el *k-ésimo* cromosoma $\vec{y}_k$ tal que $q_{k-1} < r \leq q_k$.

Finalmente, aplicamos la *selección por torneo*. Esto es, de dos cromosomas seleccionados por el método de la *ruleta*, el individuo que mejor valor de aptitud contenga será selecionado para continuar el proceso evolutivo.

**Criterio de paro**

De manera similar que las estrategias de Nelder y Mead y *Recocido Simulado*, el criterio de paro en un algoritmo *Evolutivo* es un proceso que depende de las características del problema. Esto es, resulta difícil determinar cuando un algoritmo de optimización ha encontrado el máximo o mínimo global. Nosotros empleamos dos criterios:

1. Cuando la función objetivo $Q \to 0$. Este criterio es aplicable sólo a imágenes perfectas (imágenes sintéticas). En particular, **Evolutivo_L** termina la optimización si $Q < \varepsilon^2$.

2. Cuando un porcentaje de la población converga, calculado mediante el criterio siguiente: el proceso finaliza cuando una fracción de la magnitud del vector de distancia entre el *peor* $v_w$ y el *mejor* $v_b$ valor de aptitud de los cromosomas, correspondientes a un porcentaje de la población dado, es inferior a cierta tolerancia:

$$2\frac{|v_w - v_b|}{|v_w| + |v_b|} \leq \varepsilon \tag{IV.39}$$

para nuestro caso, $\varepsilon = 1 \times 10^{-9}$.

**Rutina Evolutivo_L**

La rutina **Evolutivo_L** es la responsable de todo el proceso evolutivo, acorde a los esquemas antes planteados. De manera similar que las estrategias anteriores de optimización, **Evolutivo_L** determina los $\vec{P}$ parámetros de $M'_L(x, y, \vec{P})$ que se ajusten mejor a una ventana de la imagen original $\mathbf{Datos}[\mathbf{1}, \ldots, \mathbf{W}][\mathbf{1}, \ldots, \mathbf{W}]$. La función de aptitud $Q$ está dada por la rutina **Criterio_ch2_L**. Un diagrama simple de interacción con otras rutinas se muestra en la Figura 53. La descripción de cada una de las variables de entrada puede ser consultada en la sección V.3.8.

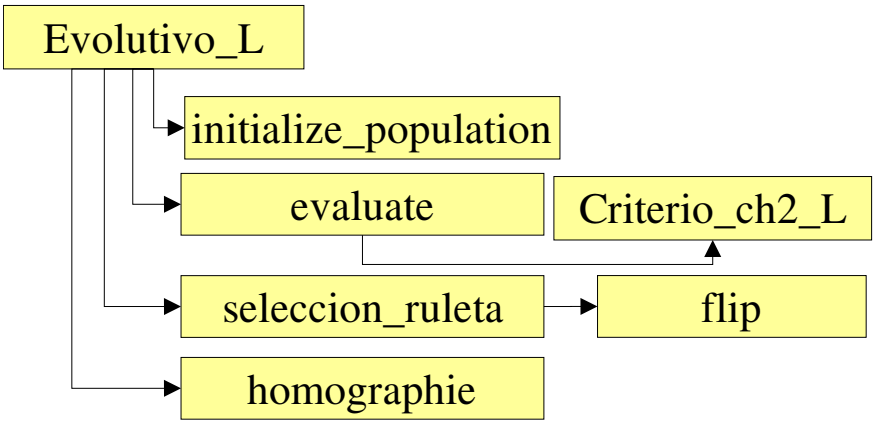



Figura 53: Rutinas utilizadas por **Evolutivo_L**.

**Evolutivo_L** se comunica con las rutinas siguientes:

1. **initialize_population**. Genera un población inicial, donde los aleles de cada cromosoma están uniformemente distribuidos entre las cotas superior e inferior, de los valores permitidos para cada uno de los parámetros del modelo, ver algoritmo ubicado en la página 92.

2. **evalute**. Evalúa a cada cromosoma a través del estimador de máxima probabilidad $Q$, rutina **Criterio_ch2_L**.

3. **seleccion_ruleta**. Criterio de selección, equivalente al método de la ruleta explicado en los parráfos anteriores.

4. **flip**. Determina si un cromosoma es candidato para la mutación o apareamiento.

5. **homographie**. Transformación afin sobre los operadores genéticos, ver parráfos anteriores.

En un algoritmo evolutivo el porcentaje de apareamiento ($pa$), mutación ($pm$) y convergencia de la población ($pc$) y el número de individuos en una población ($pop$), son variables que impactan directamente en el rendimiento de **Evolutivo_L**. Para conocer cual es el comportamiento de nuestro problema en relación con estas variables, se generó la prueba siguiente: se ejecutó **Evolutivo_L** a la estructura mostrada en la Figura 46.a, para las mismas condiciones expresadas en el proceso *Recocido Simulado*; la población

inicial es fija en $pop = 10$; $pc$ se varía en el rango de $[0.6, 0.8]$ en incrementos de 0.1; $pa$ se varía en el rango de $[0.5, 1.0]$ en incrementos de 0.05; $pm$ se varía en el rango de $[0.0, 0.5]$ en incrementos de 0.05. Como resultado de la prueba obtenemos los gráficos $a \rightarrow d$ de la Figura 54. La Figura 54.a, es una malla tridimensional del número total de pasos requeridos por el algoritmo *Evolutivo*, para la obtención de los $\vec{P}$ parámetros que se ajustan mejor a la estructura en estudio. Las Figuras 54.b $\rightarrow$ 54.d muestran como se comporta $pa$ para los porcentajes de convergencia $pc = 0.6, 0.7, 0.8$ respectivamente. Así mismo, se grafica $pc$ para los porcentajes de mutación $pm = 0.00, 0.05, 0.10, 0.15$. Analizando la Figura 54 se observa que:

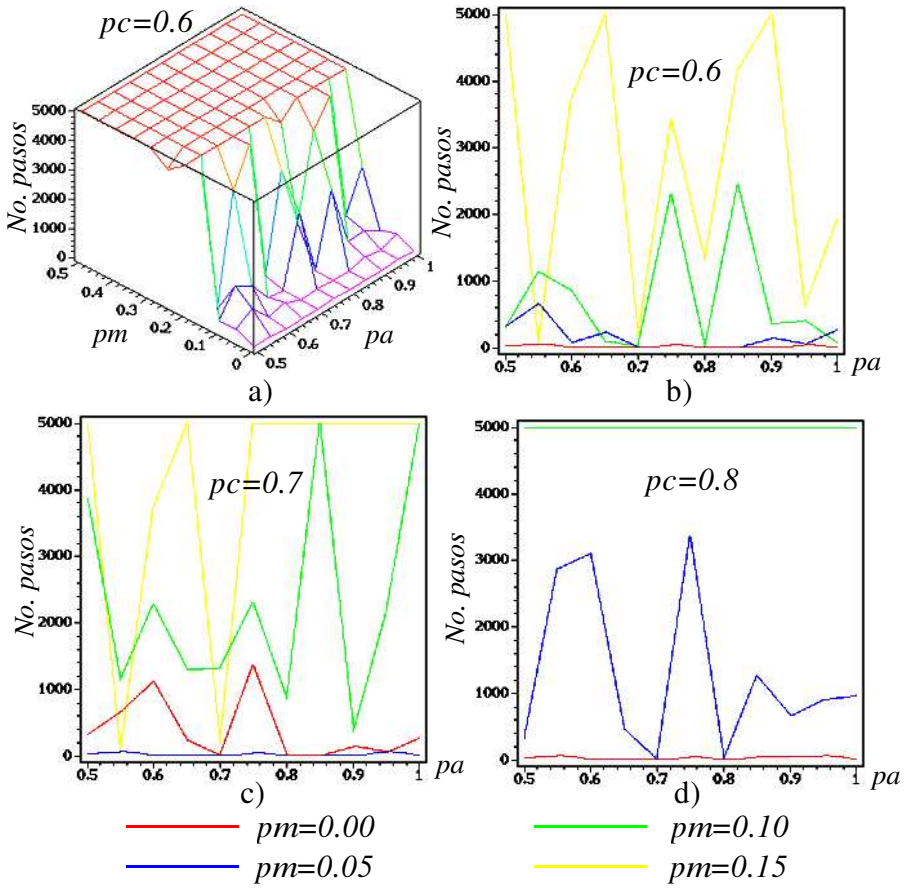



Figura 54: Comportamiento porcentaje de apareamiento ($pa$), mutación ($pm$) y convergencia de la población ($pc$) del algoritmo *Evolutivo* con respecto a $M'_L(x, y, \vec{P})$.

1. La probabilidad de mutación $pc$ de nuestro algoritmo *Evolutivo* debe ser inferior al 10%. Obsérvese como para valores superiores al 10%, **Evolutivo_L** requiere de más de 5000 pasos para lograr obtener el óptimo. En particular, 5000 pasos significa que el algoritmo finaliza por el número máximo de pasos permitidos y no por el criterio de convergencia de los individuos en la población, ver Ecuación (IV.39). Esto implica, que nuestro problema requiere de un porcentaje de exploración bajo $(0.0 < pm \leq 0.1)$, en la búsqueda del óptimo global.

2. Un porcentaje de apareamiento de alrededor de un 80% es considerado un valor aceptable para nuestro problema. Obsérvese que existe un mínimo de pasos para los valores de $pa = 0.7$ y $pa = 0.8$, ver figuras 54.b., 54.c., y 54.d. Recuerde, que el porcentaje de cruzamiento, está relacionado con el grado de explotación en una región específica del espacio de soluciones, acotada por los mejores individuos de una población. Es por esto, que nuestro problema requiere de un valor de alrededor de un 80% de la heurística de cruzamiento (*combinación intermedia afin extendida*) para determinar el óptimo global.

3. El porcentaje de convergencia $pc$ es, en escencia, el critero de paro de nuestro algoritmo evolutivo, ver Ecuación (IV.39). Esto es, en imágenes reales es imposible encontrar el caso $\chi^2 = 0$, debido a que una imagen digital no es perfecta, ver Capítulo I. Debido a ello, la determinación del mejor valor de $pc$ depende del objetivo de la ejecución de **Evolutivo_L**. Si deseamos que este algoritmo de optimización, busque exhaustivamente un óptimo global es recomendable asignar $pc = 1$ (100% de la población) para efectos de encontrar un "supercromosoma" al final del proceso. Por otra lado, si queremos inspeccionar cuales son los posibles valores de $\vec{P}$, por ejemplo, para reasignar las cotas iniciales de $\vec{P}$ en una ejecución posterior de **Evolutivo_L**, entonces es recomendable porcentajes bajos de convergencia $pc < 0.6$ para efectos de terminar rápidamente el proceso evolutivo. En la práctica, porcentajes de $70\% \leq pc \leq 90\%$ son recomendables para la obtención de $\vec{P}$.

## IV.5 Conclusiones

En este capítulo se han planteado los métodos y los algoritmos empleados para la obtención del punto exacto $(x_e, y_e)$ donde se ubica una esquina-L. La determinación de $(x_e, y_e)$ a

través de nuestro modelo $M'_L(x, y, \vec{P})$ se ha planteado como un problema de optimización. Esto es, encontrar el mejor conjunto de parámetros $\vec{P} = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$, que se ajusten mejor a una ventana cuadrada de $2w-1 \times 2w-1$ pixeles alrededor de un pixel $(u_0, v_0)$ en una imagen digital. Los elementos utilizados para este fin, son los siguientes:

- Una función objetivo o de mérito que llamamos *Criterio de Optimización*. El criterio de optimización ($Q = \chi^2$), es el estimado de máxima probabilidad, conocido como *Ajuste de Mínimos Cuadrados*, que representa al conjunto de $\vec{P}$ que se adapta mejor a la región en estudio de la imagen digital. El valor de $\chi^2$ se obtiene por medio del método propuesto por Levenberg-Marquardt. El método de Levenberg-Marquardt es un algoritmo de optimización local.

- Un conjunto de ecuaciones de tranformación, ver Ecuación (IV.11), que relaciona el sistema coordenado del modelo $(x, y)$, el sistema coordenado de la imagen $(u, v)$ y su notación matricial $(j, i)$.

- Un conjunto de estrategias de optimización global con distintas características entre ellos. Estos métodos son:

    1. Estrategia *Down Hill Simplex*. La búsqueda de un óptimo global, en este método, obedece a movimientos geométricos previamente definidos de una estructura llamada *simplex*. Para nuestro caso, el simplex es un poliédro con 9 vértices.

    2. Estrategia *Recocido Simulado*. La búsqueda de un óptimo global, en este método, obedece a la simulación del tratamiento térmico de los metales, llamado forja. Donde se definen un esquema de descenso de temperatura de la estructura y un esquema de equilibrio. Para nuestro caso: 1) el esquema de descenso de temperatura es $T_{nueva} = 0.8 \times T_{anterior}$, 2) el esquema de equilibrio está dado por los movimientos geométricos de un *simplex* de 9 vértices, perturbados por un movimiento Browniano.

    3. Estrategia *Evolutiva*. La búsqueda de un óptimo global, en este método, obedece a la evolución generacional de los integrantes de una población acorde al proceso de "seleccion natural" propuesto por Charles Darwin y conceptos Biológicos de la Genética. Para nuestro caso se emplea: 1) un apareamiento a través de una *combinación intermedia afin extendida*, 2) una mutación

*dinámica*, 3) una selección por *torneo* y *ruleta*. Los operadores de mutación y apareamiento se integran en un conjunto de transformaciones, ver ecuaciones (IV.37) y (IV.38), que proporcionan un tratamiento eficiente de los operadores evolutivos.

- Como resultado de aplicar $Q$ a las estrategias *Down Hill Simplex*, *Recocido Simulado* y *Evolutivo*, nosotros contamos con un método de optimización local inmerso en un proceso de optimización global. Por consecuencia, nuestros procesos de optimización global están acelerados por el método de Levenberg-Marquardt.

- Finalmente, para determinar el punto exacto de esquina-L $(x_e, y_e)$ a partir de nuestro *Criterio de Ubicación Exacta de esquina*, ver sección III.4.4, se realiza a través de la estragia propuesta por Nelder y Mead.

# Capítulo V

# Arquitectura del Sistema

## V.1 Introducción

En el capítulo anterior se plantearon los métodos de optimización multidimensional *Down Hill Simplex*, *Recocido Simulado* y *Evolutivo* empleados para encontrar los valores de los $\vec{P}$ parámetros de $M'_L(x, y, \vec{P})$ que se ajusten mejor a una ventana de una imagen digital. En otras palabras, se describieron los recursos teóricos necesarios para llevar a cabo nuestro trabajo.

En el presente capítulo se describe "como" estos métodos se codificaron y organizaron para poder obtener la ubicación exacta de una esquina-L $(x_e, y_e)$. En particular se generaron dos grandes grupos de programas: el primero, es un conjunto de programas codificados para detectar el punto exacto de esquina, que llamaremos *Sistema Integral para la Detección Exacta de Esquinas* (**SIDEE**), y el segundo, es un conjunto de programas que representan la interfaz con el usuario, el sistema de adquisición y el manejo de los distintos formatos de una imagen digital. Los primeros, se agrupan en un conjunto de *librerías*, que llamaremos ***Librerías de SIDEE***, que pueden ser accesadas desde cualquier programa escrito en el lenguaje de programación $C$ ó $C++$. Al segundo grupo lo llamaremos ***Programas de Explotación***. La parte medular del desarrollo es la codificación de las *Librerías de SIDEE* y están intimamente relacionados con los Capítulos III y IV. Los *Programas de Explotación* hacen uso de las *Librerías de SIDEE* y se encargan de resolver el problema de la interacción de los métodos de optimización con el *sistema de adquisición de imágenes*, ver Capítulo I, los difrentes formatos de imágenes digitales

existentes (PGM, GIF, TIFF, BMP) y la interfaz con el usuario, por ejemplo, desplegado de imágenes, control del sistema apuntador (ratón), etc.

Este capítulo está organizado en las secciones siguientes:

*Algoritmo General de Búsqueda.* Se presenta un esquema general de la secuencia de pasos requeridos para la obtención del punto exacto de esquina-L.

*Librerías de SIDEE.* Se presenta la arquitectura general de SIDEE donde se describen las rutinas que lo componen y la relación existente entre ellas.

*Programas de Explotación.* Se describen los programas generados para adquirir una imagen en tiempo real, así como los creados para analizar la sensibilidad de $M_L(x, y, \vec{P})$, a través de las librerías de SIDEE.

*Conclusiones.* Se presenta un resumen estadístico de la arquitectura del sistema y los características más relevantes del mismo.

## V.2 Algoritmo General de Búsqueda

El objetivo de SIDEE es encontrar la ubicación exacta de una esquina-L $(x_e, y_e)$ contenida en una región de la imagen $I(u, v)$, en la vecindad del pixel $(u_0, v_0)$ proporcionado por el usuario. Recuerde, que la ubicación de $(x_e, y_e)$ está definida como la mínima distancia euclidiana entre el punto de cruce $(x_0, y_0)$ de las rectas de borde $r_1$ y $r_2$ y la curvatura central del modelo $M_L(x, y, \vec{P}) = 0.5$, ver sección III.4. La región cuadrangular en análisis está definida por $w$, también proporcionada por el usuario. Por tal motivo, la subventana sobre la cual se aplicarán todos los procesos de optimización, queda acotada por $[+w, -w]$ pixeles alrededor de $(u_0, v_0)$. $(u_0, v_0)$ representa el origen del sistema coordenado del modelo $(x, y)$. Por otro lado, son 8 los parámetros de $M'_L(x, y, \vec{P})$ que se requieren ajustar a las variaciones de intensidad de $I(u, v)$. Estos parámetros son: $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B$. Adicionalmente, se necesita determinar los signos en la dirección de apertura de la esquina $s_{U_1}$ y $s_{U_2}$, para que $M'_L(x, y, \vec{P})$ quede definida completamente. A partir de estas consideraciones el algoritmo general de búsqueda de $(x_e, y_e)$ es el siguiente:

1. Dada una imagen $I(u, v)$, una posición inicial $(u_0, v_0)$ y un tamaño de ventana $2w + 1$. Se extrae una matriz de datos **DATOS**$[j, i]$, para los $j = 1, \cdots, 2w + 1$

renglones y las $i = 1, \cdots, 2w + 1$ columnas. **DATOS**$[j, i]$ contiene los valores de intensidad en tonos de gris de $I(u, v)$ alrededor del pixel $(u_0, v_0)$, obtenida mediante la ecuación de transformación (IV.11). Es importante enfatizar que **DATOS**$[j, i]$ está expresada en función del sistema coordenado de la imagen $(u, v)$ y el sistema coordenado del modelo $(x, y)$.

2. Se determina el valor de $s_{U_1}$ y $s_{U_2}$ a partir del cálculo de los promedios de los pixeles que se encuentran en la parte positiva y negativa de cada uno de los ejes del sistema coordenado del modelo $(x, y)$, ver sección III.3.

3. Se aplican los métodos de optimización global para modelos no lineales en la determinación de los valores de $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B$. Estos métodos son:

   (a) *Down Hill Simples*, discutido en la sección IV.4.1.

   (b) *Recocido Simulado*, discutido en la sección IV.4.2.

   (c) *Evolutivo*, discutido en la sección IV.4.3.

   Estos métodos se aplican uno a la vez, para determinar a $\vec{P}$. Por otro lado, el procedimiento para calcular el valor de la función de aptitud $Q$, en cada uno de los métodos anteriores, se realiza mediante el algoritmo de Levenberg-Marquardt, ver sección IV.3.3.2. Levenberg-Marquardt es un algoritmo de minimización local que nos ayuda a acelerar el proceso de búsqueda del óptimo global. Por tanto, nosotros *empleamos un algoritmo de optimización local, inmerso en un proceso de optimización global.*

4. Finalmente, se calcula el punto exacto de esquina $(x_e, y_e)$, acorde a nuestro criterio de ubicación exacta de esquina, mediante el proceso siguiente:

   (a) Se calcula el punto de cruze $(x_0, y_0)$ entre las rectas de borde $r_1$ y $r_2$, a partir de la Ecuación (III.13).

   (b) Se calcula la mínima distancia euclidiana entre $(x_0, y_0)$ y un punto $(x_s, y_s) \in M_L(x, y, \vec{P}) = 0.5$. Para resolver la ecuación implícita $M_L(x, y, \vec{P}) = 0.5$ aplicamos el algoritmo *curva de contorno* descrito en la sección III.4.3. Para determinar la mínima distancia entre $(x_0, y_0)$ y $(x_s, y_s)$ se utiliza el método de optimización propuesto por Nelder y Mead, ver sección IV.4.1.1.

(c) Una vez terminado este segundo proceso de optimización, el algoritmo reporta las coordenadas $(x_e, y_e)$ en función del sistema coordenado del modelo $(x, y)$ y el sistema coordenado de la imagen $(u, v)$. En forma específica, el punto exacto de esquina $(u_e, v_e)$ en coordenadas de la imagen a un nivel de ***subpixel*** está indicado por la expresión siguiente:

$$(u_e = u_0 + x_e, v_e = v_0 - y_e)$$

El esquema de este algoritmo se muestra en la Figura 55.

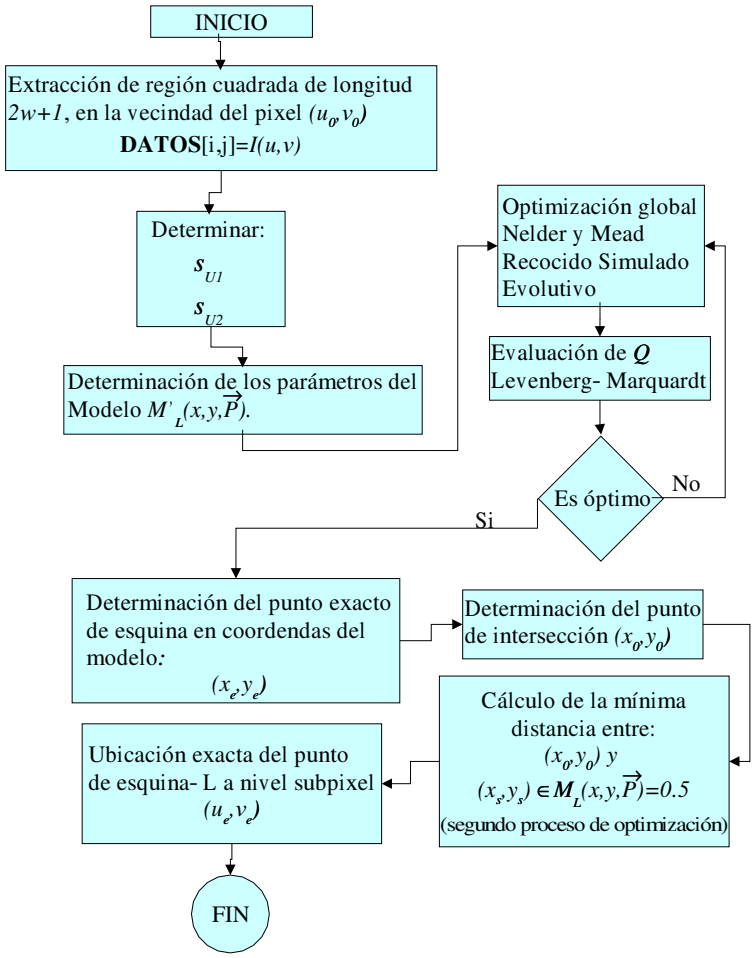



Figura 55: Algoritmo general de búsqueda del punto exacto de esquina-L $(x_e, y_e)$.

SIDEE está organizado de la forma siguiente:

- Un conjunto de librerías compartidas, que incluyen los métodos de optimización global y la geometría de nuestro modelo de esquinas múltiple. A este conjunto de librerías las llamaremos *Librerías de SIDEE.*

- Un conjunto de programas encargados de los procesos de adquisición en tiempo real de las imágenes, desplegado de imágenes en pantalla y manejo de las imágenes previamente almacenadas en archivo. A este conjunto de programas los llamaremos *Programas de Explotación.* Los programas de explotación emplean las librerías de SIDEE para determinar el punto exacto de esquina-L.

En las siguientes secciónes se describirá el contenido de cada una de las librerías y los programas de explotación que se codificaron.

## V.3 Librerías de SIDEE

SIDEE se codificó a través de un conjunto de librerías dinámicas compartidas que aglutinan todos los aspectos necesarios para su desarrollo. Dichos aspectos son: 1) funciones de evaluaciones del modelo $M'_L(x, y, \vec{P})$ y sus derivadas parciales con rescpecto a $\vec{P}$, 2) métodos de optimización global, 3) evaluciónes del criterio de optimización $Q$, 4) determinación del punto exacto de esquina $(x_e, y_e)$, 5) diversas funciones que nos apoyan en los procesos anteriores.

SIDEE está compuesto por librerías dinámicas compiladas con el lenguaje de programación *C* y *C++*. El formato empleado para asignar el nombre de estas librerías es el siguiente:

**lib***nombre***.so**

**lib***nombre***_c.so**

donde **lib** y **.so** indican al compilador que son librerías dinámicas compartidas. *nombre* es el nombre de cada librería. El subfijo **_c** indica que la librería está preparada para ser usada con programas escritos y compilados con el lenguaje *C*. La ausencias de **_c** indica que la librería está preparada para ser usada con *C++*.

SIDEE es un conjunto de librerías, por tanto es necesario generar programas que hagan uso de ellas (*programas de explotación*). Esta forma de organización de SIDEE nos permite lo siguiente:

- Aislar los métodos de optimización, la evaluación del modelo y sus derivadas, y a nuestro criterio de ubicación precisa de esquina-L, de la complejidad del manejo y adquisición de una imagen digital. Por ejemplo: 1) el manejo de distintos formatos de una imagen digital, 2) los mecanísmos de desplegado de una imagen en el ambiente X-Windows, 3) el manejo de las tarjetas de adquisición en la obtención de escenas del mundo real.
- Segmentar las funciones o rutinas tal que puedan ser usadas desde diferentes programas (código reusable).
- Agreagar, rutinas o funciones que permitan nuevas habilidades de SIDEE.

Las librerías de SIDEE son las siguientes.

1. **libfunciones.so**: evaluación de $M'_L(x, y, \vec{P})$ y manejo de la geometría de una esquina-L.
2. **libderivadas_L.so** y **libderivadas_L_c.so**: derivadas parciales de $M'_L(x, y, \vec{P})$, con respecto a los parámetros $\vec{P}$.
3. **libCriterio_ch2_L.so** y **libCriterio_ch2_L_c.so**: evaluación de $Q$ a través del método de Levenberg-Marquardt.
4. **libmisnr.so** y **libmisnr_c.so**: rutinas propuestas en el libro Numerical Recipes [Press, 1998], utilizadas por SIDEE.
5. **libDown_hill_L.so** y **libDown_hill_L_c.so**: método de optimización propuesto por Nelder y Mead.
6. **libModelo_L_Curvatura.so** y **libModelo_L_Curvatura_c.so**: criterio de ubicación exacta del punto de esquina-L.
7. **libRecocido_Simulado_L.so** y **libRecocido_Simulado_L_c.so**: método de optimización *Recocido Simulado*.
8. **libEvolutivo_L.so**: método de optimización *Evolutivo*.

### V.3.1 Librería libfunciones.so

Esta librería se encarga de evaluar el modelo de esquina $M'_L(x, y, \vec{P})$ y proporcionar los elementos de la geometría de esquina-L. Estos elementos son: 1) punto de intersección de las rectas de borde $(x_0, y_0)$, 2) ángulo de apertura de la esquina $\alpha$, 3) ángulo de la recta bisectriz de la esquina $r_{bis}$, 4) determinación de un punto $(x_s, y_s)$ que pertence a la curva de contorno central $M_L(x, y, \vec{P}) = 0.5$, 5) cálculo de los signos $s_{U_1}$ y $s_{U_2}$, en la dirección de apertura de la esquina, 6) manejo de las cotas inferior y superior de cada uno de los parámetros $\vec{P}$. **libfunciones.so** consta de 17 rutinas descritas en la Tabla V.1. La Figura 56 muestra un diagrama de interelación de estas rutinas.

Tabla V.1: Funciones asociadas a **libfunciones.so**

| | | |
|---|---|---|
| V.1.1 | Modelo_L | Cálculo de $M'_L(x, y, \vec{P})$ |
| | Variables de entrada | $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B, s_{U_1}, s_{U_2}$<br>$(x, y)$: coordenada que se quieren evaluar. |
| | Variable de salida | Valor flotante de precisión doble. |
| | Definición | double Modelo_L(double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double A, double B, double x, double y, double sgx, double sgy); |
| | Referencias | Secciones III.3, B.2 |
| V.1.2 | ABS_Modelo_L | Cálculo del valor absoluto del modelo unitario de esquina centrado en el plano $\overline{XY} = 0$.<br>$\lvert M_L(x, y, \vec{P}) - 0.5 \rvert$ |
| | Variables de entrada | $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, s_{U_1}, s_{U_2}$<br>$(x, y)$: coordenada que se quieren evaluar. |
| | Variable de salida | Valor flotante de precisión doble. |
| | Definición | double ABS_Modelo_L(double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double 1, double -0.5, double x, double y, double sgx, double sgy); |
| | Referencias | Secciones III.4.3, B.2 |

*continúa en la siguiente página*

Tabla V.1: **libfunciones.so** *(continuación)*

| | | |
|---|---|---|
| V.1.3 | NO_ABS_Modelo_L | Cálculo del modelo unitario de esquina centrado en el plano $\overline{XY} = 0$. $M_L(x, y, \vec{P}) - 0.5$ |
| | Variables de entrada | $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, s_{U_1}, s_{U_2}$<br>$(x, y)$: coordenada que se quieren evaluar. |
| | Variable de salida | Valor flotante de precisión doble. |
| | Definición | double ABS_Modelo_L(double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double 1, double -0.5, double x, double y, double sgx, double sgy); |
| | Referencias | Secciones III.4.3, B.2 |
| V.1.4 | M_f | Cálculo del modelo unitario de esquina centrado en el plano $\overline{XY} = 0$. Esta rutina es similar a V.1.3, con la diferencia que los parámetros de la función son vectores. Esto facilita el llamado de esta función por los procesos de optimización donde los parámetros $\vec{P}$ se definen como vectores. |
| | Variables de entrada | $k\_param = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, s_{U_1}, s_{U_2})$<br>$k\_min = (x, y)$ |
| | Variable de salida | Valor flotante de precisión doble. |
| | Definición | double M_f(double k_min[], double k_param[]); |
| | Referencias | Secciones III.4.3, B.2 |
| V.1.5 | sol | Cálculo del punto de intersección $(x_0, y_0)$ de las rectas de borde $r_1$ y $r_2$. $(x_0, y_0)$ son variables de precisión doble. $\left(x_0 = -\frac{\tan(\vartheta_1)\mu_2 + \mu_1}{\tan(\vartheta_2)\tan(\vartheta_1) - 1}, y_0 = -\frac{\tan(\vartheta_2)\mu_1 + \mu_2}{\tan(\vartheta_2)\tan(\vartheta_1) - 1}\right)$ |
| | Variables de entrada | $\mu_1, \vartheta_1, \mu_2, \vartheta_2$ |

*continúa en la siguiente página*

Tabla V.1: **libfunciones.so** *(continuación)*

| | | |
|---|---|---|
| | Variable de salida | $x$: coordenada $x_0$.<br>$y$: coordenada $y_0$. |
| | Definición | void sol(double *x, double *y, double mu1, double theta1, double mu2, double theta2); |
| | Referencias | Secciones III.4.1, B.2 |
| V.1.6 | Bisectriz | Cálculo del ángulo de la recta bisectriz $r_{bis}$ en la dirección del ángulo de apertura de la esquina-L $\alpha$.<br>$\vartheta_2 + \frac{\alpha}{2}$<br>$\alpha = \arccos\left(\frac{\vec{r_1}\cdot\vec{r_2}}{\lvert\vec{r_1}\rvert\lvert\vec{r_2}\rvert}\right)$ |
| | Variables de entrada | $\mu_1, \vartheta_1, \mu_2, \vartheta_2, s_{U_1}, s_{U_2}$<br>$(x_0, y_0)$: punto de intersección entre $r_1$ y $r_2$ |
| | Variable de salida | Valor flotante de precisión doble. |
| | Definición | double Bisectriz( double X0, double Y0, double mu1, double theta1, double mu2, double theta2, double sgx, double sgy); |
| | Referencias | Secciones III.4.2 III.4.4, B.2 |
| V.1.7 | Recta | Ecuación de una recta en su forma punto pendiente. recta.<br>$y - y_0 = m(x - x_0)$ |
| | Variables de entrada | $\beta$: ángulo cualesquiera.<br>$(x_0, y_0)$: punto arbitrario.<br>$x$: valor de una coordenada. |
| | Variable de salida | Valor flotante de precisión doble. |
| | Definición | double Recta( double X0, double Y0, double beta, double x); |
| | Referencias | Sección B.2 |

*continúa en la siguiente página*

Tabla V.1: **libfunciones.so** *(continuación)*

| | | |
|---|---|---|
| V.1.8 | distancia | Cálculo de la distancia entre los puntos $(x_s, y_s)$ y $(x_0, y_0)$. $$(x_s - x_0)^2 + (y_s - y_0)^2$$ |
| | Variables de entrada | $P\_xy = (x_0, y_0)$: punto intersección entre $r_1$ y $r_2$.<br>$k\_min = (x_s, y_s) \in M_L(x, y, \vec{P}) - 0.5 = 0$. |
| | Variable de salida | Valor flotante de precisión doble. |
| | Definición | double distancia( double k_min[], double P_xy[]); |
| | Referencias | Secciones III.4.4, IV.4.1.1, B.2 |
| V.1.9 | Brac | Valores iniciales del algoritmo de *cotas y bisección*. $P_1$ y $P_2$ son dos puntos que pertenecen a una recta $r_s$. Busco el valor de $P_1$ y $P_2$ tal que $M_L(P_1, \vec{P}) - 0.5$ y $M_L(P_2, \vec{P}) - 0.5$ tengan signos opuesto. |
| | Variables de entrada | $k\_p = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, s_{U_1}, s_{U_2})$<br>$\beta$: ángulo de $r_s$.<br>$P_1$ y $P_2$ |
| | Variable de salida | Esta rutina proporciona dos estados. En el caso de no encontrar un cambio de signo al evaluar nuestro modelo en $P_1$ y $P_2$, la rutina entrega el valor entero dos. De otra forma entrega el valor cero. |
| | Definición | int Brac( double *P1, double *P2, double beta, double k_p[]); |
| | Referencias | Secciones III.4.3, B.2 |
| V.1.10 | Intercepta | Búsqueda del punto $(x_s, y_s) \in M_L(x, y, \vec{P}) - 0.5 = 0$. Esta rutina corresponde al algoritmo de *Curva de Contorno*. El algoritmo consiste en generar una recta de búsqueda $r_s$ definida entre $r_1$ y $r_2$ que pasa por $(x_0, y_0)$. Finalmente, se calcula numéricamente el punto de interecepción $(x_s, y_s)$ entre $r_s$ y $M_L(x, y, \vec{P}) - 0.5 = 0$. |

*continúa en la siguiente página*

Tabla V.1: **libfunciones.so** *(continuación)*

| | | |
|---|---|---|
| | Variables de entrada | $k\_param = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, s_{U_1}, s_{U_2})$.<br>$\beta$: ángulo de $r_s$.<br>$(x_0, y_0)$: punto de intersección de las rectas de borde.<br>*d_brac*: es la distancia entre las cotas iniciales donde se inspecciona el cambio de signo. |
| | Variable de salida | $(xi, yi) = (x_s, y_s)$.<br>*N_iter* es el número de pasos que da este algoritmo.<br>Esta rutina maneja dos estados: "0" sin encuentra exitosamente a $(x_s, y_s)$ y "2" en caso contrario. |
| | Definición | int Intercepta( double X0, double Y0, double beta, double d_brac, double *xi, double *yi, int *N_iter, double k_param[]); |
| | Referencias | Secciones III.4.3, B.2 |
| V.1.11 | Manejo_error_2 | Maneja el error "2". Cuando no se encuentra un cambio de signo, entonces se regresa un punto sobre $r_1$ ó $r_2$. Si el ángulo $\beta$ de $r_s$ está más cercano a $r_1$ que a $r_2$, entonces el punto $(x_s, y_s)$ pertenece a $r_1$. En caso contrario, $(x_s, y_s)$ pertenecia a $r_2$. Este fenómeno ocurre generalmente cuando el método de *Downhill Simplex* propone vértices nuevos en el cálculo de nuestro *criterio de ubicación exacta de esquina-L*. |
| | Variables de entrada | k_param=$(\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, s_{U_1}, s_{U_2})$<br>$angulo = \beta$<br>$cotay = \vartheta_1$<br>$cotax = \vartheta_2$<br>$(x_0, y_0)$<br>*delta_x*: constante en la dirección de la apertura de la esquina. |
| | Variable de salida | $(xi, yi) = (x_s, y_s)$. |

*continúa en la siguiente página*

Tabla V.1: **libfunciones.so** *(continuación)*

| | | |
|---|---|---|
| | Definición | void Manejo_error_2( double X0, double Y0, double angulo, double cotagy, double cotagx, double delta_x, double *xi, double *yi, double k_param[]); |
| | Referencias | Secciones IV.4.1.1, B.2 |
| V.1.12 | Limites_L | Inspecciona que los $\vec{P}$ parámetros no exedan los límites definidos por el modelo. En cuyo caso el valor de $\vec{P}$ es substituido por la cota superior o inferior según corresponda. |
| | Variables de entrada | $N\_par$: número de parámetros del modelo.<br>$Win\_size$: tamaño de la ventana en análisis. |
| | Variable de salida | $P = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$.<br>En caso de existir alguna modificación a $\vec{P}$, esta rutina regresa el valor de "$-1$". En caso contrario regresa el valor cero. |
| | Definición | int Limites_L( double *P , int N_par, int Win_size); |
| | Referencias | Secciones IV.4.1.1, B.2 |
| V.1.13 | Kernel_gauss | Cálcula un núcleo Gaussiano bidimensional.<br>$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{(-\frac{x^2+y^2}{2\sigma^2})}$$ |
| | Variables de entrada | $(x, y)$: coordenada.<br>$\sigma$: factor de suavizado del núcleo Gaussiano. |
| | Variable de salida | Valor flotante de precisión doble. |
| | Definición | double Kernel_gauss( double x, double y, double sigma); |
| | Referencias | Sección B.2 |
| V.1.14 | Promedio_Matrix | Promedio de una región de una matriz.<br>$$\frac{1}{(y_{ini} - y_{fin})(x_{ini} - x_{fin})} \sum_{r=y_{ini}}^{y_{fin}} \sum_{c=x_{ini}}^{x_{fin}} M(r,c)$$ |

*continúa en la siguiente página*

Tabla V.1: **libfunciones.so** *(continuación)*

| | | |
|---|---|---|
| | | $M$: matriz. |
| | Variables de entrada | $[y_{ini}, y_{fin}]$: cotas de los $r$ renglones. |
| | | $[x_{ini}, x_{fin}]$: cotas de las $c$ columnas. |
| | Variable de salida | Valor flotante de precisión doble. |
| | Definición | double Promedio_Matrix (double **M, int xini, int xfin, int yini, int yfin); |
| | Referencias | Algoritmo ubicado en la página 43. Sección B.2. |
| V.1.15 | Cual_cuadrante | Cálculo de los valores de $s_{U_1}$ y $s_{U_2}$. El cuadrante cuyo valor sea superior determina los signos de $s_{U_1}$ y $s_{U_2}$. |
| | Variables de entrada | $M$: matriz de valores de intensidad en tonos de gris.<br>$W$: tamaño de la ventana. |
| | Variable de salida | $sgx = s_{U_1}$<br>$sgy = s_{U_2}$ |
| | Definición | void Cual_cuadrante( double **M, int W, double *sgx, double *sgy); |
| | Referencias | Algoritmo ubicado en la página 43. Sección B.2. |
| V.1.16 | Cambia_background | Invierte los valores de los pixeles en tonos de gris de la matriz de datos $M$: $M(r,c) = 255 - M(r,c)$. |
| | Variables de entrada | $W$: tamaño de la ventana. |
| | Variable de salida | $M$: matriz de valores de intensidad en tonos de gris. |
| | Definición | Cambia_background( double **M, int W); |
| | Referencias | Sección B.2 |
| V.1.17 | Si_background_blanco | Para efectos de visualizar a una esquina, donde el fondo de la estructura represente el piso de nuestro modelo, en ocaciones es necesario invertir los valores de intensidad. Esta rutina invierte los valores de intensidad si el número de pixeles con valor superior al promedio total es mayor que el número de pixeles con valor inferior a dicho promedio. |
| | Variables de entrada | $W$: tamaño de la ventana. |

*continúa en la siguiente página*

Tabla V.1: **libfunciones.so** *(continuación)*

| | |
|---|---|
| Variable de salida | $M$: matriz de valores de intensidad en tonos de gris. |
| Definición | void Si_backgraund_blanco( double **M, int W); |
| Referencias | Sección B.2 |

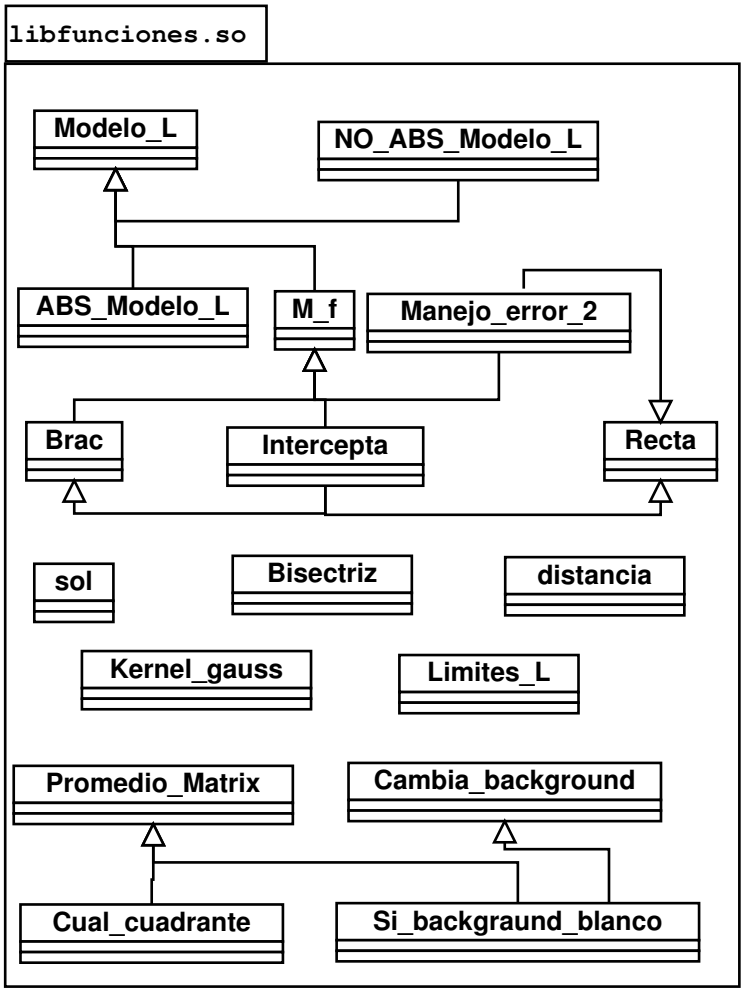



Figura 56: Rutinas que componen **libfunciones.so**.

### V.3.2 Librería libderivadas_L.so

Esta librería se encarga de evaluar las derivadas parciales del modelo de esquina-L $M'_L(x, y, \vec{P})$ con respecto a los parámetros $\vec{P}$. La siguiente tabla muestra las derivadas de los 8 parámetros del modelo. En la Figura 57 se muestra el diagrama de las rutinas que integran a **libderivdas_L.so**.

Tabla V.2: Rutinas asociadas a **libderivadas_L.so**

| | | |
|---|---|---|
| | | Las variables de entrada y de salida son las mismas para cada una de las siguientes rutinas, así como sus referencias. Por tanto se describirán una vez. |
| | Variables de entrada | $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B, s_{U_1}, s_{U_2}$<br>$(x, y)$: coordenada de un punto. |
| | Variables de salida | Valor flotante de precisión doble. |
| | Referencias | Secciones IV.3.3.3, B.2 |
| V.2.1 | dm_mu1 | $\frac{\partial M'_L(x, y, \vec{P})}{\partial \mu_1}$ |
| | Definición | double dm_mu1( double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double A, double B, double x, double y, double sgx, double sgy); |
| V.2.2 | dm_mu2 | $\frac{\partial M'_L(x, y, \vec{P})}{\partial \mu_2}$ |
| | Definición | double dm_mu2( double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double A, double B, double x, double y, double sgx, double sgy); |
| V.2.3 | dm_theta1 | $\frac{\partial M'_L(x, y, \vec{P})}{\partial \vartheta_1}$ |

*continúa en la siguiente página*

Tabla V.2: **libderivadas_L.so** *(continuación)*

| | | |
|---|---|---|
| | Definición | double dm_theta1( double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double A, double B, double x, double y, double sgx, double sgy); |
| V.2.4 | dm_theta2 | $\frac{\partial M'_L(x,y,\vec{P})}{\partial \vartheta_2}$ |
| | Definición | double dm_theta2( double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double A, double B, double x, double y, double sgx, double sgy); |
| V.2.5 | dm_sigma1 | $\frac{\partial M'_L(x,y,\vec{P})}{\partial \sigma_1}$ |
| | Definición | double dm_sigma1( double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double A, double B, double x, double y, double sgx, double sgy); |
| V.2.6 | dm_sigma2 | $\frac{\partial M'_L(x,y,\vec{P})}{\partial \sigma_2}$ |
| | Definición | double dm_sigma2( double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double A, double B, double x, double y, double sgx, double sgy); |
| V.2.7 | dm_A | $\frac{\partial M'_L(x,y,\vec{P})}{\partial A}$ |
| | Definición | double dm_A( double sigma1, double mu1, double theta1, double sigma2, double mu2, double theta2, double A, double B, double x, double y, double sgx, double sgy); |

*continúa en la siguiente página*

Tabla V.2: **libderivadas_L.so** *(continuación)*

| | | |
|---|---|---|
| V.2.8 | dm_B | $\frac{\partial M'_L(x, y, \vec{P})}{\partial B}$ |
| | Definición | #ifndef dm_B<br>#define dm_B 1.0<br>#endif |

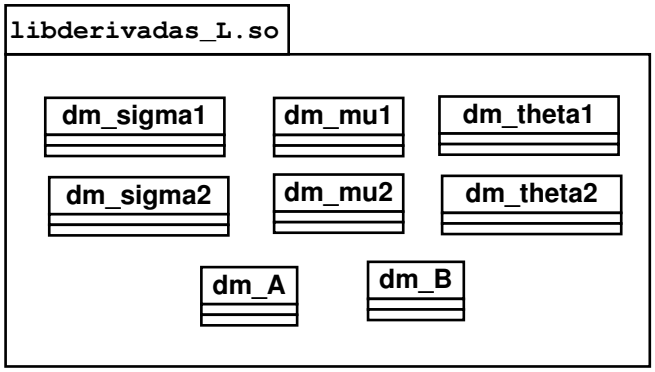



Figura 57: Rutinas que componen **libderivadas_L.so**.

### V.3.3 Librería libCriterio_ch2_L.so

La librería **libCriterio_ch2_L.so** se encarga de evaluar la función de aptitud $Q$ a través del algoritmo de Levenberg-Marquardt. Esta librería contiene 2 rutinas descritas en la Tabla V.3. Adicionalemnte, **libCriterio_ch2_L.so** hace uso de **libfunciones.so** y **libmisnr.so**. En la Figura 58 se muestran las rutinas que pertenecen a **libCriterio_ch2_L.so**, así como el nombre de las rutinas expecíficas llamadas por **libCriterio_ch2_L.so** que pertenecen a otros módulos (**libfunciones.so** y **libmisnr.so**).

Tabla V.3: Rutinas asociadas a **libCriterio_ch2_L.so**

| | | |
|---|---|---|
| | | Las observaciones y la hipótesis son manejadas como vectores por el método de Levenberg-Marquart. Por otro lado, nuestras observaciones ($Datos[][]$) y nuestras hipótesis ($M'_L(x, y, \vec{P})$) son manejadas como matrices. Por tal motivo, la matriz $Datos[][]$ se vectoriza en el vector $y[]$ y las coordendas $(x, y)$ correspondientes a dicha matriz se vectorizan en los vectores $x\_cor[]$ y $v\_cor[]$. Finalmente, $x\_cor[]$ y $v\_cor[]$ se definen como variables globales de todo este proceso. |
| | Variables globales | $x\_cor[]$ y $v\_cor[]$. |
| | Definición | static double *x_cor, *v_cor; |
| | Referencias | Secciones IV.3.3, B.2. |
| V.3.1 | corner | Evaluación del modelo de esquina $M'_L(x, y, \vec{P})$ y las derivadas parciales $\partial M'_L(x, y, \vec{P})/\partial \vec{P}$. |
| | Variables de entrada | $a[] = \sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B, s_{U_1}, s_{U_2}$<br>*indice*: manejo del índice de $x\_cor[]$ y $v\_cor[]$.<br>*na* no utilizada.<br>$dyda[]$: derivadas parciales del modelo. |
| | Variable de salida | $y$: evaluación del modelo en las coordendas $x\_cor[]$ y $v\_cor[]$. |
| | Definición | void corner( double indice, double a[], double *y, double dyda[], int na); |
| V.3.2 | Criterio_ch2_L | Cálculo de $Q$ |
| | Variables de entrada | $Datos[][]$: matriz de los valores de intensidad de la región en estudio.<br>$ia[]$: vector de control de parámetros ($ia[k] = 1$ significa que el parámetro $k$ debe ajustarse, de otra forma $ia[k] = 0$).<br>$W$: tamaño de la ventana.<br>$Nmodel$: número de parámetros a ajustar. |

*continúa en la siguiente página*

Tabla V.3: **libCriterio_ch2_L.so** *(continuación)*

| | |
|---|---|
| | $Q$: Valor de $\chi^2$ en precisión doble. |
| Variable de salida | $a[] = \sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B, s_{U_1}, s_{U_2}$: parámetros ajustados correspondientes a $Q$. |
| Definición | double Criterio_ch2_L( double **Datos, double *a, int *ia, int W, int Nmodel); |

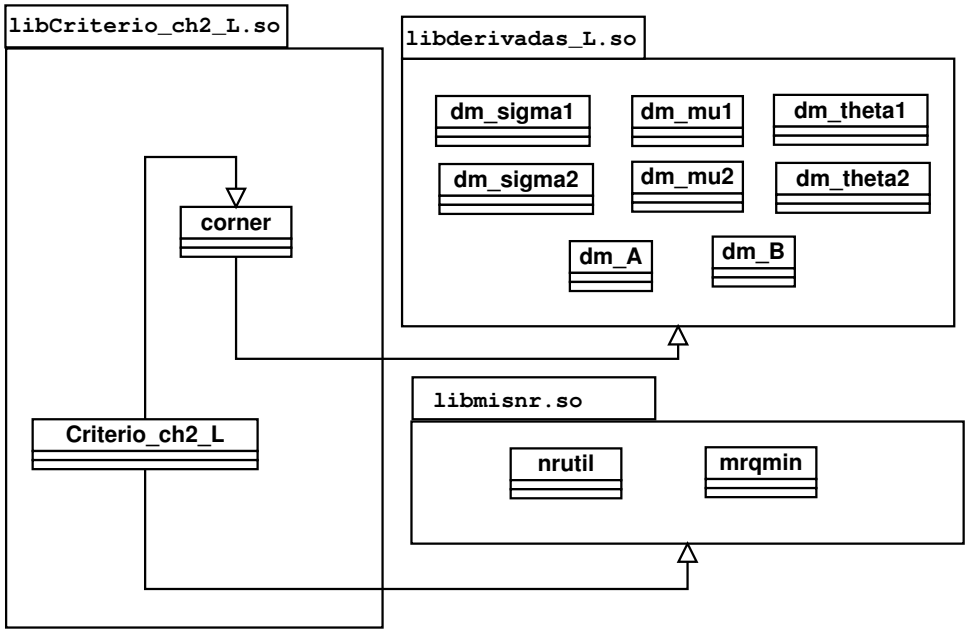



Figura 58: Rutinas que componen **libCriterio_ch2_L.so**.

### V.3.4 Librería libdmisnr.so

La librería **libdmisnr.so** contiene únicamente las rutinas correspondies al libro *Numerica Recipes in C* [Press, 1998] que emplea SIDEE. Dichas rutinas están expresadas en su forma original. La Tabla V.4 y la Figura 59 muestran las rutinas y su interrelación asociadas a **libdmisnr.so**.

Tabla V.4: Rutinas asociadas a **libmisnr.so**

| | | |
|---|---|---|
| V.4.1 | nrutil | Asignación dinámica de espacio en memoria de vectores matrices y cubos de datos en diferentes tipos de varibales (enteras, reales en precisión doble y caracter). A continuación se citan las rutinas empleadas más frecuentemente. Donde $i_1, j_1, k_1$ son los índices iniciales y $i_2, j_2, k_2$ son el último índice de los vectores o matrices según sea el caso. |
| | Funciones | vector$(i_1, i_2)$ : Vector en precisión doble.<br>free_vector$(a)$ : Libera memoria del vector $a$.<br>matrix$(i_1, i_2, j_1, j_2)$: Matriz en precisión doble.<br>free_matrix$(a)$ : Libera memoria de la matriz $a$.<br>f3tensor$(i_1, i_2, j_1, j_2, k_1, k_2)$: Cubo de datos en prcisión doble.<br>free_f3tensor$(a)$: Libera memoria del cubo de datos $a$. |
| | Referencias | Press *et al.*, [Press, 1998]. |
| V.4.2 | mrqmin | Realiza una iteración del método de Levenberg-Marquardt. |
| | Variables de entrada | $x[]$: vector de puntos a evaluar.<br>$y[]$: vectorización de las observaciones $Datos[][]$.<br>$sig[]$: desviación estándar del error esperado. Para nuestro caso, $sig[] = 1$.<br>$ndata = (2w+1)(2w+1)$: número total de datos.<br>$ia[]$: control de parámetros a minimizar.<br>$mfit$: número de parámetros del modelo.<br>$alpha[][] = \frac{1}{2}\mathbf{H}$. matriz del Hessiano.<br>$funcs =$ **corner**: función de evaluación $M'_L(x, y, \vec{P})$.<br>$alamda = \lambda$: Si $\lambda \leq 0$ se inicializan las variables de este método. Si $\lambda = 0$ se extrae el valor final de $\chi^2$ para los $\vec{P}$ parámetros optimizados y su matriz de covarianza correspondiente. |

*continúa en la siguiente página*

Tabla V.4: **libmisnr.so** *(continuación)*

| | | |
|---|---|---|
| | | $a[] = \vec{P}$: parámetros del sistemas optimizados. |
| | | $covar[][]$: matriz de covariancia de $\vec{P}$. |
| | Variables de salida | $chisq = \chi^2$: Valor de $Q$. |
| | | mrqmin regresa un valor de 0 si existe algun error. De otra forma regresa el valor de $-1$. |
| | Definición | int mrqmin( double x[], double y[], double sig[], int ndata, double a[], int ia[], int ma, double **covar, double **alpha, double *chisq, void (*funcs)(double, double [], double *, double [], int), double *alamda); |
| | Referencias | Secciones IV.3.3.1, IV.3.3.2, IV.3.3.3 |
| V.4.3 | mrqcof | Cálculo de:. $\beta_k \equiv -\frac{1}{2}\frac{\partial F(\vec{P})}{\partial \vec{P}_k}$ $\alpha_{kl} \equiv \frac{1}{2}\frac{\partial^2 F(\vec{P})}{\partial \vec{P}_k \partial \vec{P}_l}$ |
| | Variables de entrada | $x[], y[], sig[], ndata, a[], ia[], ma, funcs$: variables definidas en la rutina **mrqmin**, ver Tabla V.4.2. |
| | | $chisq = \chi^2$: Valor de $Q$. |
| | Variables de salida | $alpha[][] = \alpha_{kl}$ |
| | | $beta[] = \beta_k$ |
| | Definición | void mrqcof( double x[], double y[], double sig[], int ndata, double a[], int ia[], int ma, double **alpha, double beta[], double *chisq, void (*funcs)( double, double [], double *, double [], int)); |
| | Referencias | Secciones IV.3.3.1, IV.3.3.2, IV.3.3.3 |

### V.3.5 Librería libDown_hill_Ł.so

La librería **libDown_hill_L.so** se encarga calcular los mejores $\vec{P}$ parámetros de $M'_L(x, y, \vec{P})$ que se ajusten mejor a la región observada. La determinación de $\vec{P}$ se realiza con el método propuesto por Nelder y Mead. Como se mencionó en la sección IV.4.1.1 este método está basado en la rutina **amoeba** y **amotry** propuestas por Press *et al.*

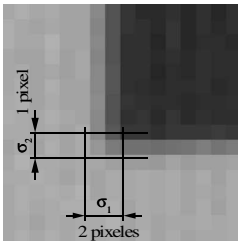



Figura 59: Rutinas que componen **libmisnr.so**.

[Press, 1998]. Las cuales son modificadas de la forma siguiente:

1. Las rutinas **amoeba** y **amotry** suponen que la función de evaluación o aptitud *funk* es de la forma $y = f(x : \vec{a})$. Donde $\vec{a}$ es un vector $n$-dimensional que se quiere minimizar. Sin embargo, nuestra función de aptitud es $Q$. $Q$ depende tanto de un vector de parámetros $\vec{P}$, así como de dos variables independiente $(x, y)$ dadas por el sistema coordenado del modelo. Es evidente que no es útil la forma de definir la función de aptitud en **amoeba**. Más aún $Q$ representa, un método de optimización local por él mismo, ver IV.3.3. Por tanto se módifico la definición de la función *funk* por el método de Levenberg-Marquardt calculado a partir de la rutina **Criterio_chi2_L**.

2. Se agregó la matriz de datos $Datos[][]$, el vector de control de parámetros $iP_m[]$ y el tamaño de la ventana $W$, en los argumentos de las rutinas **amoeba** y **amotry**. Dichas variables son necesarias para el manejo de **Criterio_chi2_L**.

3. Se adicionó la rutina **Limites_L** para asegurar que los movimientos propuestos por el *simplex*, correspondan a valores dentro del rango permitido en los $\vec{P}$ parámetros.

4. Finalmete, las rutinas **Modelo_L_amoeba** y **Modelo_L_amotry** corresponden a la modificación de las rutinas **amoeba** y **amotry** respectivamente.

La Tabla V.5 y la Figura 60 muestra las rutinas correspondientes a **libDown_hill_Ł.so**.

Tabla V.5: Rutinas asociadas a **libDown_hill_L.so**

| | | |
|---|---|---|
| V.5.1 | Down_hill_L | Genera el simplex inicial del método *Downhill simplex*. Cada vértice del simplex es una muestra uniforme de los posibles valores de $\vec{P}$. Las cotas de $\vec{P}$ están definidas por el usuario. Cada uno de los vértices iniciales es sometido a **Criterio_chi2_L** a fín de iniciar el proceso de optimización. |
| | Variables de entrada | $Datos[][]$: región en análisis.<br>$W$: tamaño de la ventana.<br>$signo\_x = s_{U_1}$<br>$signo\_y = s_{U_2}$<br>$N\_par$: número de parámetros a determinar.<br>$P\_max[]$: cota superior de $\vec{P}$.<br>$P\_min[]$: cota inferior de $\vec{P}$.<br>$ftol$: tolerancia.<br>$NMAX$: número máximo de pasos permitidos.<br>$P\_m = \sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B$ |
| | Variables de salida | Entrega el número total de movimientos requeridos en la determinación de $\vec{P}$. |
| | Definición | int Down_hill_L( double **Datos, int W, double signo_x, double signo_y, double *P_m, int N_par, double *P_max, double *P_min, double ftol, int NMAX); |
| | Referencias | Secciones IV.4.1.1, B.2 |
| V.5.2 | Modelo_L_amoeba | Minimización multimensional de la función de aptitud $Q$ a partir de la estrategia prpuesta por Nelder y Mead. |

*continúa en la siguiente página*

Tabla V.5: **libDown_hill_L.so** *(continuación)*

| | | |
|---|---|---|
| | Variables de entrada | $Datos[][]$: región en análisis.<br>$y[]$: evaluación de función de aptitud $Q$.<br>$ndim$: número de parámetros a minimizar.<br>$ftol$: tolerancia<br>$funk =$ **Criterio_ch2_L**: función objetivo $\chi^2$ calculada por el método de Levenberg-Marquadt.<br>$iP\_m[]$: vector de control de parámetros, ver Tabla V.3.2.<br>$W$: tamaño de la ventana.<br>$NMAX$: número máximo de pasos permitidos. |
| | Variables de salida | $p[][]$: vértices del simpex. El primer reglón de la matriz $p[1][]$ corresponde al mejor valor de $\vec{P}$.<br>$nfunk$: número total de movimientos del simplex. |
| | Definición | void Modelo_L_amoeba( double **p, double y[], int ndim, double ftol, double (*funk)(double **, double *,int *, int, int ), int *nfunk, double **Datos, int *iP_m, int W, int NMAX); |
| | Referencias | Secciones IV.4.1.1, B.2 |
| V.5.3 | Modelo_L_amotry | Cálculo de un nuevo vértice acorde a los movimientos propuestos del simplex. Si el nuevo vértice propuesto es mejor que todos los demás, entonces el mejor vértice del simplex (antes del movimiento) será reemplazado por el nuevo. |
| | Variables de entrada | $y[], ndim, funk, fac, Datos[][], i\_Pm[], W$: variables definidas en V.5.2.<br>$psum[]$: vector de peso $\frac{1}{n}\sum_{i=1,i\neq w}^{n+1} p_{(i,k)} \quad k = 1,\ldots,n$<br>$ihi$: índice del mejor vértice del simplex. |
| | Variables de salida | $p[][]$: vértices del simpex.<br>Evaluación de la función objetivo en precisión doble. |

*continúa en la siguiente página*

Tabla V.5: **libDown_hill_L.so** *(continuación)*

| | |
|---|---|
| Definición | double Modelo_L_amotry( double **p, double y[], double *psum, int ndim, double (*funk) (double **, double *, int *, int, int), int ihi, double fac, double **Datos, int *iP_m, int W); |
| Referencias | Secciones IV.4.1.1, B.2 |

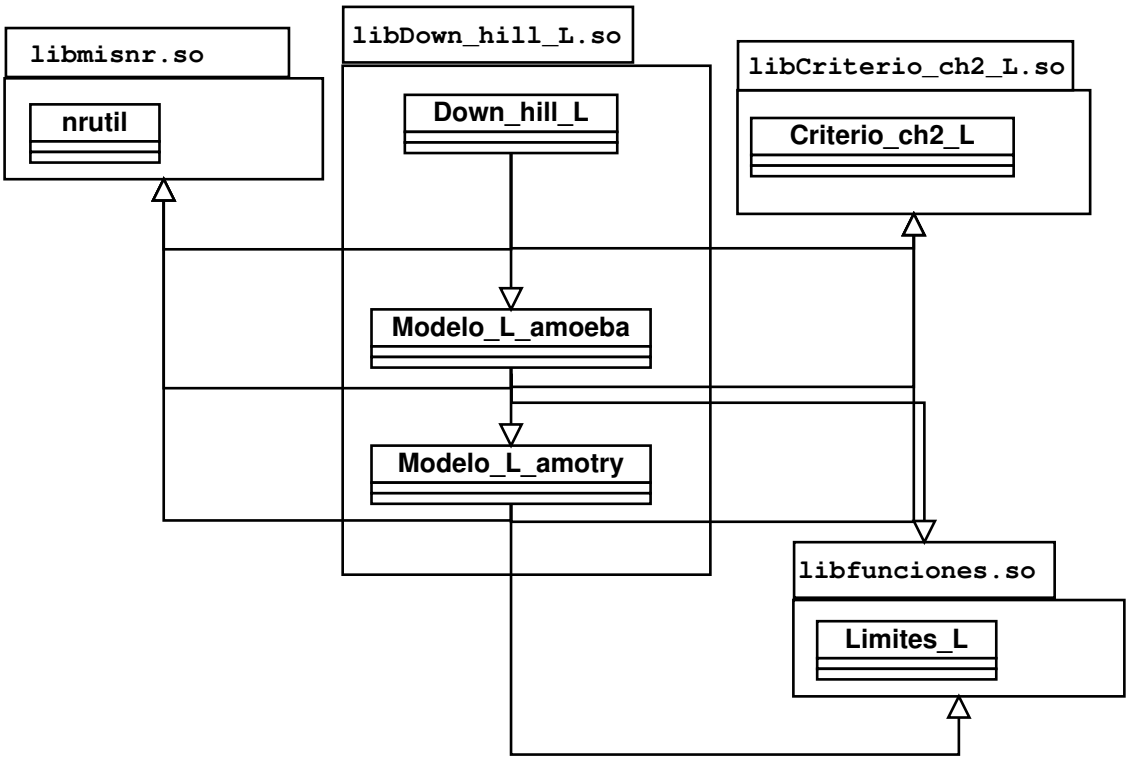



Figura 60: Rutinas que componen **libDown_hill_L.so**.

### V.3.6 Librería libModelo_L_Curvatura.so

La librería **libModelo_L_Curvatura.so** es la responsable del cálculo de punto exacto de una esquina-L $(x_e, y_e)$ que representa la minimina distancia euclidina $D_{min}$ entre el punto de intersección de las rectas de borde $r_1$ y $r_2$ y la curva de contorno central de

nuestro modelo definida por $M_L(x, y, \vec{P}) = 0.5$. La Tabla V.6 y la Figura 61 muestran las rutinas de **libModelo_L_Curvatura.so**.

El método de optimización propuesto por Nelder y Mead se utiliza para la extración de $(x_e, y_e)$. Al igual que **libDown_hill_L.so** las rutinas **amoeba** y **amotry** se modificaron de la forma siguiente:

1. La función $funk$ = **distancia**. La rutina **distancia**, ver Tabla V.1.8, calcula la distancia entre el punto de intersección $(x_0, y_0)$ de las rectas de borde y el punto $(x_s, y_s)$.

2. Se incluyeron las rutinas **Intercepta**, Tabla V.1.10, y **Manejo_error_2**, Tabla V.1.11, en el código de **amoeba** y **amotry**. Recuerde que resulta difícil resolver la ecuación ímplicita $M_L(x, y, \vec{P}) = 0.5$. Nosotros obtenemos numéricamente las coordenadas $(x_s, y_s)$ que satisfacen a $M_L(x, y, \vec{P}) = 0.5$ a través del algoritmo de *curva de controno* (rutina **Intercepta**). La rutina **Manejo_error_2** se incluye debido a que los movimientos propuestos por el simplex, no siempre generan una recta $r_s$ en la dirección de apertura de la esquina. Para mayor detalle puede consultarse la sección IV.4.1.1.

3. El vector parámetros del modelo $\vec{P}$ y el punto de $(x_0, y_0)$ son incluidos como argumento en **amoeba** y **amotry**.

4. Finalmente, las rutinas **mi_amoeba** y **mi_amotry** corresponden a la modificación de las rutinas **amoeba** y **amotry** respectivamente.

Tabla V.6: Rutinas asociadas **libModelo_L_Curvatura.so**

| | | |
|---|---|---|
| V.6.1 | Modelo_L_Curvatura | Genera el simplex inicial compuesto de 3 vértices para el cálculo de $D_{min}$. El primer vértice corresponde al punto de intersección de la recta bisectriz $r_{bis}$ y la curva $M_L(x, y, \vec{P}) = 0.5$. El segundo y el tercer vértice corresponde al punto de intersección de $r_1$ y $r_2$ con $M_L(x, y, \vec{P}) = 0.5$ respectivamente. |

*continúa en la siguiente página*

Tabla V.6: **libModelo_L_Curvatura.so** *(continuación)*

| | | |
|---|---|---|
| | Variables de entrada | $Param\_m = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B, s_{U_1}, s_{U_2})$<br>$N\_par$: número de parámetros del modelo.<br>$ftol$: Tolerancia. |
| | Variables de salida | $P\_xy = (x_e, y_e)$: Punto de ubicación exacta de una esquina-L.<br>Entrega el número total de movimientos requeridos en la determinación de $(x_e, y_e)$. |
| | Definición | int Modelo_L_Curvatura( double Param_m[], int N_par, double ftol, double *P_xy); |
| | Referencias | Secciones III.4.4, IV.4.1.1, B.2 |
| V.6.2 | mi_amoeba | Minimización multimensional de la función de aptitud $D_{min}$ a partir de la estrategia propuesta por Nelder y Mead. |
| | Variables de entrada | $y[]$: evaluación de función de aptitud $Q$.<br>$ndim = 2$: número de parámetros a minimizar.<br>$ftol$: tolerancia<br>$funk =$ **Intercepta**: función objetivo $D_{min}$.<br>$k\_p = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B, s_{U_1}, s_{U_2})$<br>$P\_xy[] = (x_0, y_0)$: Punto de intersección de las rectas de borde $r_1$ y $r_2$. |
| | Variables de salida | $p[][]$: vértices del simplex. El primer renglón de la matriz $p[1][]$ corresponde a la coordenada $(x_e, y_e)$.<br>$nfunk$: número total de movimientos del simplex. |
| | Definición | void mi_amoeba( double **p, double y[], int ndim, double ftol, double (*funk)(double [], double []), int *nfunk, double k_p[], double P_xy[]); |
| | Referencias | Secciones III.4.4, B.2 |

*continúa en la siguiente página*

Tabla V.6: **libModelo_L_Curvatura.so** *(continuación)*

| | | |
|---|---|---|
| V.6.3 | mi_amotry | Cálculo de un nuevo vértice acorde a los movimientos propuestos del simplex. Si el nuevo vértice propuesto es mejor que todos los demás, entonces el mejor vértice del simplex (antes del movimiento) será reemplazado por el nuevo. |
| | Variables de entrada | $y[], ndim, funk, fac, k\_p[], P\_xy[]$: variables definidas en V.6.2.<br>$psum[]$: vector de peso, ver Tabla V.5.3.<br>$ihi$: índice del mejor vértice del simplex. |
| | Variables de salida | $p[][]$: vértices del simpex.<br>Evaluación de la función objetivo en precisión doble. |
| | Definición | double mi_amotry( double **p, double y[], double psum[], int ndim, double (*funk)(double [], double[]), int ihi, double fac, double k_p[], double P_xy[]) |
| | Referencias | Secciones III.4.4, B.2 |

### V.3.7 Librería libRecocido_Simulado_L.so

La librería **libRecocido_Simulado_L.so** se encarga de calcular el mejor conjunto de parámetros $\vec{P}$ de $M'_L(x, y, \vec{P})$ que se ajusten a una ventana de la imagen, a través del método de *Recocido Simulado*, ver sección IV.4.2. La librería **libRecocido_Simulado_L.so** se basa en las rutinas **amoebsa** y **amotsa** propuesta por Press *et al.* [Press, 1998]. De manera similar que **libDown_hill_L.so** estas rutinas se modificaron de la forma siguiente:

1. La función $funk$ = **Criterio_ch2_L**. La rutina **Criterio_ch2_L** representa el cálculo de nuestra función objetivo $Q$ a partir del método de Levenberg-Marquardt ver Tabla V.3.2.

2. Se agregó la matriz de datos $Datos[][]$, el vector de control de parámetros $iP_m[]$ y el tamaño de la ventana $W$, en los argumentos de las rutinas **amoebsa** y **amotsa**. Dichas variables son necesarias para el manejo de **Criterio_chi2_L**.

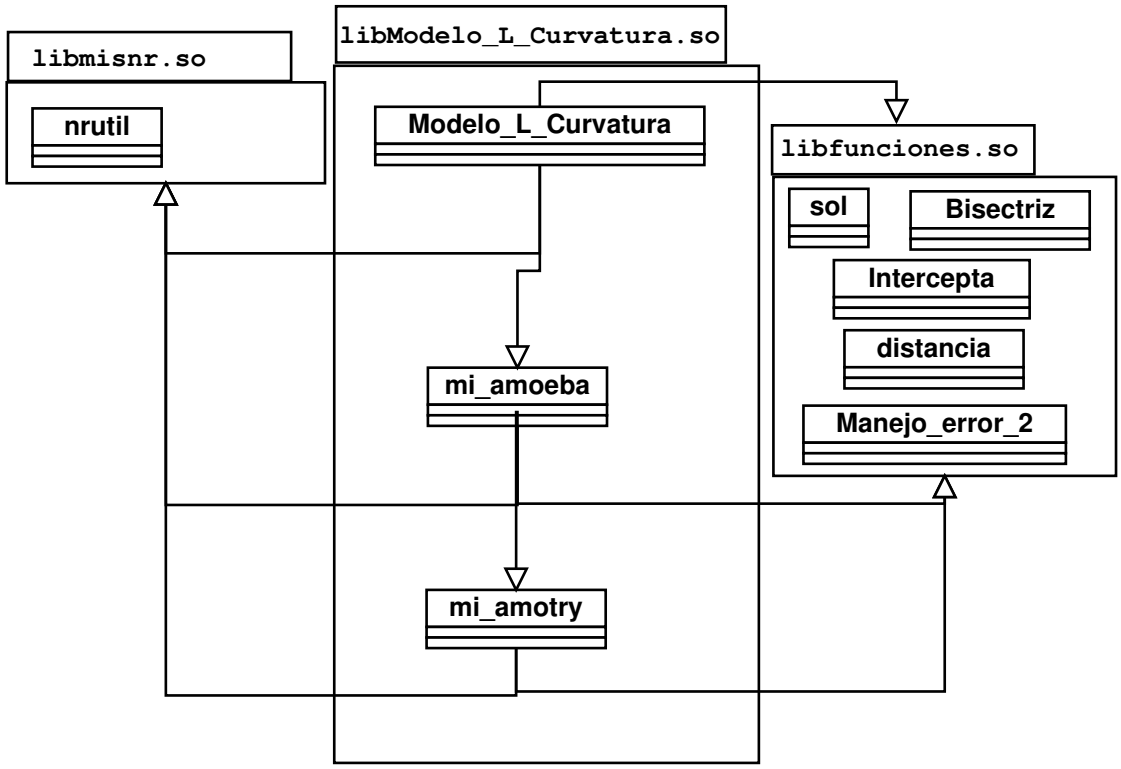



Figura 61: Rutinas que componen **libModelo_L_Curvatura.so**.

3. Las rutinas **Modelo_L_amoebsa** y **Modelo_L_amotsa** corresponden a la modificación de las rutinas **amoebsa** y **amotsa** respectivamente.

La Tabla V.7 y la Figura 62 muestran las rutinas correspondiente a **libRecocido_Simulado_L.so**.

Tabla V.7: Rutinas asociadas **libRecocido_Simulado_L.so**

| | | |
|---|---|---|
| V.7.1 | Recocido_Simulado_L | Genera el simplex inicial del método *Recocido Simulado*. Este simplex toma una muestra *aleatoria* dentro del rango permitido de $\vec{P}$. El esquema de descenso de temperatura utilizado esta dado por: $T_{nueva} = 0.8(T_{anterior})$. El esquema del estado de equilibrio esta dado por los $I$ movimientos geométricos que puede dar el simplex. |

*continúa en la siguiente página*

Tabla V.7: **libRecocido_Simulado_L.so** *(continuación)*

| | | |
|---|---|---|
| | Variables de entrada | $Datos[][]$: región en análisis.<br>$W$: tamaño de la ventana.<br>$signo\_x = s_{U_1}$<br>$signo\_y = s_{U_2}$<br>$N\_par$: número de parámetros a determinar.<br>$ftol$: Tolerancia.<br>$iteracion = I$: número de pasos permitidos en el estado de equilibrio.<br>$temperatura = T$: temperatura inicial.<br>$NMAX$: número máximo de pasos del algoritmo. |
| | Variables de salida | $P\_m = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$<br>Entrega el número total de movimientos requeridos en la determinación de $\vec{P}$. |
| | Definición | int Recocido_Simulado_L( double **Datos, int W, double signo_x, double signo_y, double *P_m, int N_par, double ftol, int iteracion, double temperatura, int NMAX); |
| | Referencias | Secciones IV.4.2.1, B.2 |
| V.7.2 | Recocido_Simulado_L_iso | Esta rutina es similar a V.7.1 con la modificación siguiente: El simplex inicial toma una muestra *uniforme* dentro del rango permitido de $\vec{P}$, definido explícitamente por el usuario. |
| | Variables de entrada | $Datos[][], W, signo\_x, signo\_y, N\_par, ftol, iteracion,$ $temperatura, NMAX$: definidos en la Tabla V.7.1.<br>$P\_max[]$: Cota superior de $\vec{P}$.<br>$P\_min[]$: Cota inferior de $\vec{P}$. |
| | Variables de salida | $P\_m = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$<br>Entrega el número total de movimientos requeridos en la determinación de $\vec{P}$. |

*continúa en la siguiente página*

Tabla V.7: **libRecocido_Simulado_L.so** *(continuación)*

| | | |
|---|---|---|
| | Definición | int Recocido_Simulado_L_iso (double **Datos, int W, double signo_x, double signo_y, double *P_m, int N_par, double *P_max, double *P_min, double ftol, int iteracion, double temperatura, int NMAX); |
| | Referencias | Secciones IV.4.2.1, B.2 |
| V.7.3 | Modelo_L_amoebsa | Esquema de equilibrio, controlado por los movimientos del simplex. Las perturbaciones de cada vértice en el estado de equilibrio simulan un movimiento "Browniano" equivalente al equilibrio térmico propuesto por Metropolis. |
| | Variables de entrada | $Datos[][]$: región en análisis.<br>$y[]$: evaluación de función de aptitud $Q$.<br>$ndim$: número de parámetros a minimizar.<br>$ftol$: tolerancia<br>$funk =$ **Criterio_ch2_L**: función objetivo $\chi^2$ calculada por el método de Levenberg-Marquadt.<br>$iP\_m[]$: vector de control de parámetros, ver Tabla V.3.2.<br>$W$: tamaño de la ventana.<br>$temptr = T_{nueva}$: temperatura en la cual se busca el equilibrio térmico.<br>$iter$: número de movimientos necesarios para encontrar un equilibrio térmico del simplex, donde $iter \leq I$. |
| | Variables de salida | $p[][]$: Vértices del simplex.<br>$yb$: Valor de $Q$ para los mejores $\vec{P}$.<br>$pb[] = \vec{P}$: mejores valores de los parámetros a una temperatura dada. |

*continúa en la siguiente página*

Tabla V.7: **libRecocido_Simulado_L.so** *(continuación)*

| | | |
|---|---|---|
| | Definición | void Modelo_L_amebsa( double **p, double y[], int ndim, double pb[], double *yb, double ftol, double (*funk)( double **, double *,int *, int, int ), int *iter, double temptr, double **Datos, int *iP_m, int W); |
| | Referencias | Sección IV.4.2.1, B.2 |
| V.7.4 | Modelo_L_amotsa | Cálculo de un nuevo vértice acorde a los movimientos propuestos del simplex. Si el nuevo vértice propuesto es mejor que todos los demás, entonces el mejor vértice del simplex (antes del movimiento) será reemplazado por el nuevo. |
| | Variables de entrada | $y[], ndim, funk, fac, Datos[][], i\_Pm[], W$: variables definidas en V.7.3.<br>$psum[]$: vector de peso, ver Tabla V.5.3.<br>$ihi$: índice del mejor vértice del simplex.<br>$yhi$: valor de $Q$ para el mejor vértice del simplex. |
| | Variables de salida | $p[][], pb[], yb$: variables definidas en la Tabla V.7.3<br>Evaluación de la función objetivo en precisión doble. |
| | Definición | double Modelo_L_amotsa( double **p, double y[], double psum[], int ndim, double pb[], double *yb, double (*funk) (double **, double *,int *, int, int ), int ihi, double *yhi, double fac, double **Datos, int *iP_m, int W); |
| | Referencias | Sección IV.4.2.1, B.2 |

### V.3.8 Librería libEvolutivo_L.so

La librería **libEvolutivo_L.so** se encarga de calcular el mejor conjunto de parámetros $\vec{P}$ de $M'_L(x, y, \vec{P})$ que se ajusten a una ventana de una imagen digital, a través de la estrategia *Evolutiva*, ver sección IV.4.3. La Tabla V.8 y la Figura 63 muestran las rutinas que conformanan a esta librería. **libEvolutivo_L.so** define una estructura de los individuos de una población como sigue:

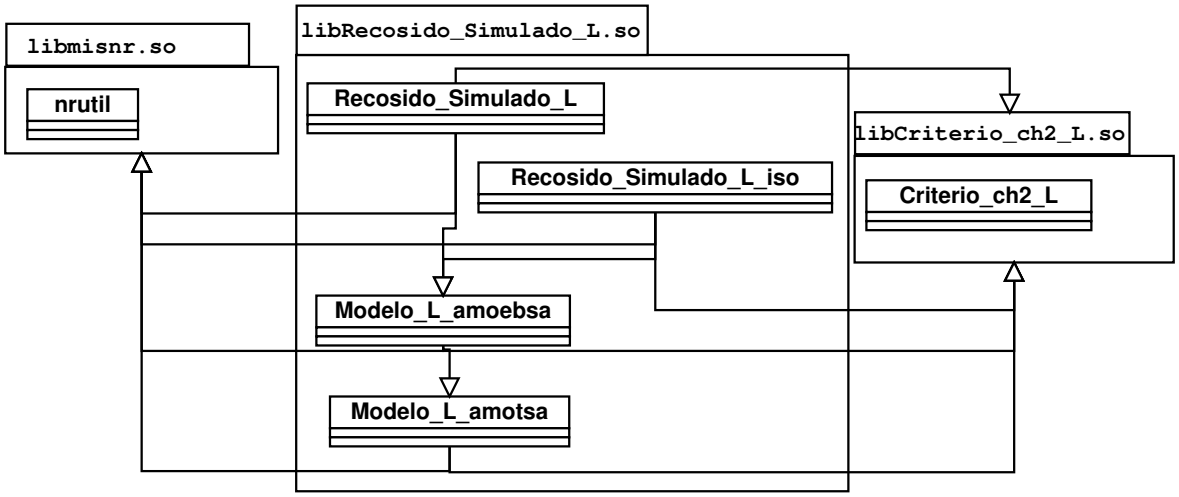



Figura 62: Rutinas que componen **libRecocido_Simulado_L.so**.

```
struct   population   {
         double       *value;
         double       fitness;
     };
```

donde los primeros 8 elementos del vector $value[] = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B, s_{U_1}, s_{U_2})$ deben ser ajustados. $fitness$ contiene el valor de $\chi^2$ de cada uno de los individuos. Por tanto para cada individuo de una población, existe una estructura *population*. En términos de programación, se define un vector del tipo *population* del tamaño de la población. Finalmente, los operadores evolutivos trabajan sobre esta estructura durante todo el proceso evolutivo.

Tabla V.8: Rutinas asociadas **libEvolutivo_L.so**

| | | |
|---|---|---|
| V.8.1 | Evolutivo_L | Se encarga de la evolución generacional de todos los individuos. Esta rutina termina cuando el valor $\chi^2$ es menor a cierta tolerancia $Q \leq ftol^2$ o cuando se excede el número máximo de generaciones o cuando la Ecuación IV.39 en menor a $ftol$. |

*continúa en la siguiente página*

Tabla V.8: **libEvolutivo_L.so** *(continuación)*

| | | |
|---|---|---|
| | Variables de entrada | $Datos[][]$: región en análisis.<br>$W$: tamaño de la ventana.<br>$signo\_x = s_{U_1}$<br>$signo\_y = s_{U_2}$<br>$N\_par$: número de parámetros a determinar.<br>$P\_max[]$: cota superior de $\vec{P}$.<br>$P\_min[]$: cota inferior de $\vec{P}$.<br>$ftol$: tolerancia.<br>$POPULATION\_SIZE$: número de individuos en una población.<br>$PCROSS$: porcentaje de apareamiento, expresada en el rango de $[0.0, 1.0]$.<br>$PMUT$: porcentaje de mutación, expresada en el rango de $[0.0, 1.0]$.<br>$PCONV$: porcentaje de convergencia de los individuos, expresada en el rango de $[0.0, 1.0]$.<br>$MAX\_GEN$: número máximo de generaciones permitidas.<br>$P\_m = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$ |
| | Variables de salida | Entrega el número total de generaciones requeridas en la determinación de $\vec{P}$. |
| | Definición | int Evolutivo_L(double **Datos, int W, double signo_x, double signo_y, double *P_m, int N_par, double *P_max, double *P_min, double ftol, int POPULATION_SIZE, double PCROSS, double PMUT, int MAX_GEN, double PCONV); |
| | Referencias | Secciones IV.4.3.2, B.2 |
| V.8.2 | initialize_population | Genera una población inicial distribuida iniformemente entre $P\_max[]$ y $P\_min[]$. |

*continúa en la siguiente página*

Tabla V.8: **libEvolutivo_L.so** *(continuación)*

| | | |
|---|---|---|
| | | $POPULATION\_SIZE$, $P\_max$[] y $P\_min$[] definidas en V.8.1 |
| | Variables de entrada | $sgx = s_{U_1}$<br>$sgy = s_{U_2}$<br>*variables*: número de parámetros a determinar. |
| | Variables de salida | *pool*[]: vector de individuos del tipo *population*. |
| | Definición | void initialize_population (struct population *pool, int variables, int POPULATION_SIZE, double sgx, double sgy, double *P_max, double *P_min); |
| | Referencias | Secciones IV.4.3.2, B.2 |
| V.8.3 | evaluate | Cálculo de $\chi^2$ para un individuo. |
| | Variables de entrada | $Datos$[][], $W$ y $N\_par$ definidas en V.8.1<br>*individuo*: un individuo de la población del tipo *population*.<br>$iP\_m$[]: vector de control de algoritmo de Levenberg-Marquard, ver Tabla V.4.2 |
| | Variables de salida | $fitness$: valor de $\chi^2$. |
| | Definición | double evaluate (struct population individuo, double **Datos, int * iP_m, int W, int N_par); |
| | Referencias | Secciones IV.3.3.3, IV.4.3.2, B.2 |
| V.8.4 | seleccion_ruleta | Algoritmo de selección por el método de la *ruleta* y *torneo*. |
| | Variables de entrada | $POPULATION\_SIZE$ y *pool*[] definidas en V.8.2<br>*suma_ap*: aptutud total de la población.<br>*qap*[]: suma de la aptitud acumulada.<br>*inicia*: si $inicia = 0$, inicia el proceso de selección, entonces se calcula *qap*[] para esa generación. De otra forma, no es necesario recalcular *qap*[], en cada uno de los giros de la rueda de la ruleta. |

*continúa en la siguiente página*

Tabla V.8: **libEvolutivo_L.so** *(continuación)*

| | | |
|---|---|---|
| | Variables de salida | regresa el índice $i$ del individuo seleccionado. Esto es el individuo $pool[i]$ es seleccionado para mutación o cruzamiento. |
| | Definición | int seleccion_ruleta (int inicia, int POPULATION_SIZE, double suma_ap, struct population *pool, double *qap); |
| | Referencias | Secciones IV.4.3.2, B.2 |
| V.8.5 | homografie | Cáculo de la transformación afin sobre los operadores genéticos, para un parámetro de un par de individuos de una población. |
| | Variables de entrada | $POPULATION\_SIZE$, $PCROSS$, $PMUT$, $MAX\_GEN$, $P\_max[]$ y $P\_min[]$ definidas en V.8.1<br>$pool\_temp$: población del tipo *population*, antes de aplicar los operadores genéticos.<br>$parent1$: índice del primer padre seleccionado para mutación o cruzamiento.<br>$parent2$: índice del segundo padre sleccionado.<br>$child1$: índice del primer hijo generado.<br>$child2$: índice del segundo hijo generado.<br>$variable$: índice del alele sobre el cual se llevara a cabo la tranformación afin. |
| | Variables de salida | $pool[]$: población nueva del tipo *populations*. |
| | Definición | void homographie (int parent1, int parent2, int child1, int child2, int variable, double PMUT, double PCROSS, int generations, int MAX_GEN, struct population *pool, struct population *pool_temp, double P_max[], double P_min[]); |
| | Referencias | Secciones IV.4.3.2, B.2 |

*continúa en la siguiente página*

Tabla V.8: **libEvolutivo_L.so** *(continuación)*

| | | |
|---|---|---|
| V.8.6 | flip | Genera una bandera de estado (cierto o falso), a partir de la lógica siguiente: se genera un número aleatorio $r$ uniformemente distribuido entre $[0, 1]$; si $r$ es mayor que cierta probabilidad *prob* entonces regresa el valor 1, de otra forma regresa 0. |
| | Variables de entrada | *prob*: probabilidad a evaluar. |
| | Variables de salida | Cierto "1" ó falso "0". |
| | Definición | int flip (double prob); |
| | Referencias | Secciones IV.4.3.2, B.2 |

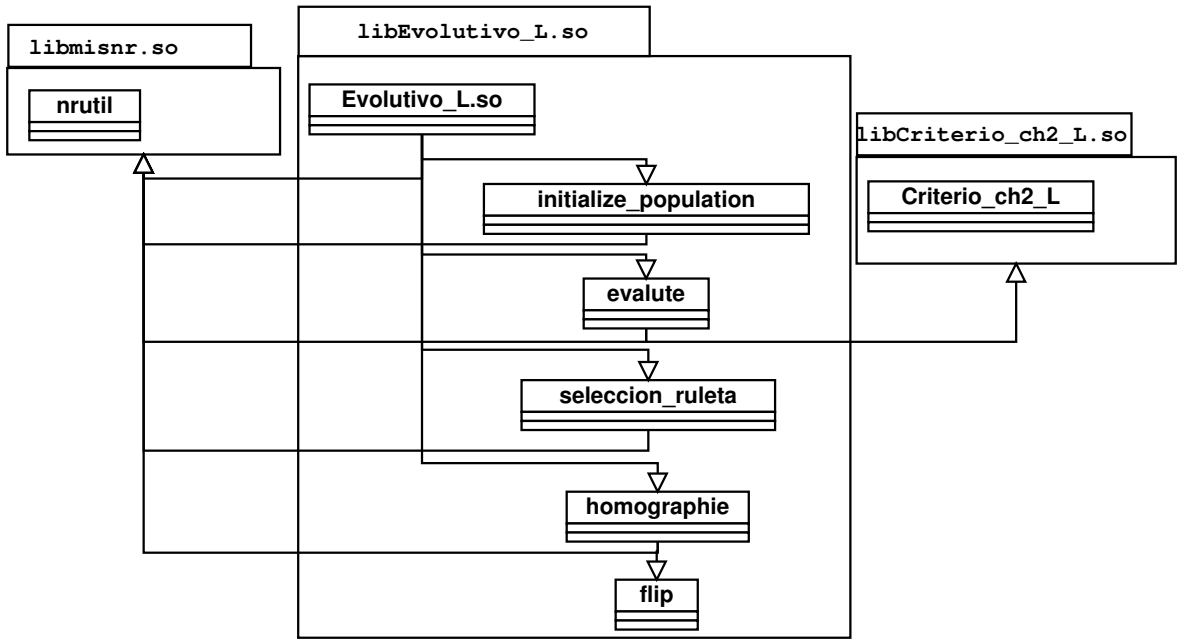



Figura 63: Rutinas que componen **libEvolutivo_L.so**.

## V.4 Programas de Explotación

Los *programas de explotación* son responsables de la manipulación de la imagen digital, la adquisición de una escena en el mundo real y de la presentación de resultados al

usuario. Dichos programas se comunican con SIDEE a través del llamado a un método de optimización o a un conjunto de rutinas que componen a SIDEE. Dependiendo del método de optimización o la rutina que se desee emplear, el programa de explotación deberá ser ligado con la librería que contenga a dicha rutina. Las figuras 56 a 63 muestran una guía rápida del contenido de cada una de las librerías de SIDEE y su interdependencia.

Existen dos tipos de programas de explotación: 1) Los programas de *adquisición y desplegado* en pantalla de imágenes y 2) los programas de *cálculo masivo*. Los primeros, permiten visualizar una imagen o escena del mundo real en pantalla. Los programas de adquisición y desplegado permiten también, la selección de las regiones donde se quiera aplicar nuestro detector preciso de esquina-L por parte del usuario, así como el método de optimización deseado. Los segundos, aplican algún método de optimización a una región previamente definida por el usuario, sin desplegar la imagen en pantalla. El objetivo de los programas de cálculo masivo, es la obtención de diferentes estadísticas del comportamiento de $M'_L(x, y, \vec{P})$ para distintos tipos de estructura.

Los programas de explotación que se codificaron son los siguientes:

1. *Programas de adquisición y desplegado.* Son programas de adquisición de imágenes y ubicación exacta del punto de esquina $(x_e, y_e)$. Dichos programas contienen todos los métodos de optimización presentados en el presente desarrollo. Además, controlan la adquisición y despliegue de imágenes de una escena real a través del sistema estereo, ver Figura 64.

   (a) **capturaDosCamaras**, ver sección V.4.1.

   (b) **capturaDosTest**, ver sección V.4.2.

2. *Programas de cálculo masivo.* Son programas que obtienen diferentes estadísticas del comportamiento de nuestro detector preciso de esquinas múltiples para distintos tipos de regiones, sin desplegar la imagen en pantalla. La intención principal de estos programas es la facilidad de poder ejecutarse en "lotes[1]" para la obtención de diversas estadísticas. Las estadística son de diverso tipo, vease por ejemplo las figuras 36, 37 y 47. Sin embargo, los resultados de cada ejecución de los programas

[1]conocidos en inglés como "batch process"

de cálculo masivo, son almacenados en un archivo de texto con el formato por renglón siguiente:

Tabla V.9: Formato de archivo de resultados

| | |
|---|---|
| Campo 1: | valor de la coordenada $u_0$. |
| Campo 2: | valor de la coordenada $v_0$. |
| Campo 3: | tamaño de la región $W$. |
| Campo 4: | disponible. |
| Campo 5: | disponible. |
| Campo 6: | valor de $\sigma_1$. |
| Campo 7: | valor de $\mu_1$. |
| Campo 8: | valor de $\vartheta_1$. |
| Campo 9: | valor de $\sigma_2$. |
| Campo 10: | valor de $\mu_2$. |
| Campo 11: | valor de $\vartheta_2$. |
| Campo 12: | valor de $A$. |
| Campo 13: | valor de $B$. |
| Campo 14: | valor de $s_{U_1}$. |
| Campo 15: | valor de $s_{U_2}$. |
| Campo 16: | total de pasos del método de optimización en la obtención de $\vec{P}$. |
| Campo 17: | total de pasos en la obtención de $(x_e, y_e)$. |
| Campo 18: | valor de $x_e$. |
| Campo 19: | valor de $y_e$. |
| Campo 20: | valor de $u_e$. |
| Campo 21: | valor de $v_e$. |
| Campo 22: | segundos empleados para la obtención de $\vec{P}$. |
| Campo 23: | segundos empleados para la obtención de $(x_e, y_e)$. |
| Campo 24: | nombre del archivo de la imagen. |

Este archivio es la base para mostrar diferentes resultados. El archivo de resultados tiene la extensión "**.res**". Otra característica común que tienen los programas de cálculo masivo, es el empleo de las librerías de control de imágenes proporcionadas por *Image Understanding Environment* (IUE) [Lerner, 1999]. IUE es un paquete de

visión por computadora soportado por *Amerinex Applied Imaging*. Dicho paquete puede ser obtenido en la dirección *www.aai.com/AAI/IUE*. Adicionalmente, los programas de cálculo masivo se codificaron empleando el estilo de los comandos del sistema operativo UNIX. Esto es, su ejecución requiere de múltiples "opciones" en un orden específico, vea por ejemplo, el programa **Sensibiliad** en la sección V.4.3. Finalmete, los programas de cálculo masivo son los siguientes:

(a) **Sensibiliad**.

(b) **Modelo_L_DownHill**.

(c) **Modelo_L_DownHill_ruido**.

(d) **Modelo_L_DownHill_gauss**.

(e) **Modelo_L_Recocido**.

(f) **Modelo_L_Recocido_iso_ruido**.

(g) **Modelo_L_Evolutivo**.

La compilación de cada uno de los programas de explotación, programas de adquisición y desplegado de imágenes y los programas de cálculo masivo, puede ser consultada en el Apéndice B.

### V.4.1 Programa capturaDosCamaras

El programa **capturaDosCamaras** permite que el usuario observe en la pantalla de la computadora, una escena del mundo real. Dicha escena es capturada a través de un sistema de adquisición que consta de lo siguiente:

1. Dos cámaras digitales marca Pulnix TM-9101.

2. Dos tarjetas de adquisición de imagen Marca MUTECH MV-1500.

3. Una computadora personal marca DELL.

4. Una montura que permite colocar ambas cámaras sobre un mismo eje. Así mismo, la montura permite que las cámaras esten separadas una distanacia $\lambda$.

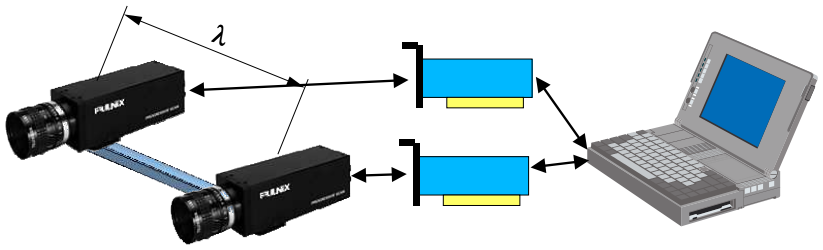



Figura 64: Composición de la *cabeza estereoscópica*.

A este sistema de adquisición lo llamaremos *cabeza estereoscópica*, ver Figura 64. Para mayor referencia técnica de cada uno de los elementos que componen la cabeza estereoscópica consulte el Apéndice A. El programa **capturaDosCamaras** es un sistema integral que controla la adquisición de cabeza estereoscópica y permite al usuario lo siguiente:

- Visualizar las imágenes de la cabeza estereoscópica.
- Congelar las imágenes cuando el usuario lo decida.
- Seleccionar la región ó regiones donde se buscará la ubicación exacta de la esquina.
- Seleccionar el método de optimización multidimensional (DownHill Simplex, Recocido Simulado y Evolutivo) que se aplique a las regiones previamente seleccionadas.
- Reporta en pantalla la ubicación exacta de los puntos de esquina, de cada una de las regiones.

#### V.4.1.1 Arquitectura

**capturaDosCamaras** emplea las rutinas de control de la tajeta digitalizadora en la adquisición de las imágenes. Así mismo, despliega la imagen adquirida haciendo uso del sistema XWindows. Finalmente, el control de las regiones y los métodos de optimización son manipulados por "botones" generados también con Xwindos. En forma específica, **capturaDosCamaras** hace uso de los sistemas siguientes:

1. Las librerías de adquisición de la tarjeta digitalizadora MuTech, las cuales permiten el control de la cámara Pulnix y la transferencia de datos de la memoria de la tarjeta a la memoria RAM. Estas librerías son proporcionadas por el fabricante. En

particular, las librerías deben ser incluidas como módulos dentro del "kernel" del sistema, las cuales están compiladas para trabajar con el sistema operativo RedHat 5.2 y el kernel-2.0, ver referencia [MV-Vision, 1998]. Debido a que el fabricante no proporciona el código fuente de las librerías y los módulos, existe la restricción de operar con el sistema operativo RedHat 5.2. En el futuro, es recomendable adquirir una versión más actualizada de las mismas.

2. Las librerías de X-based Image Processing Tools and Enviroment (XITE) que proporcionan la funcionalidad del manejo de la imagen a través del ambiente XWindows. Para mayor referencias consulte [Böe, 1998a], [Böe, 1998b], [Böe, 1998c]. XITE es un paquete gratuito que puede ser obtenido en la dirección Internet *www.ifi.uio.no/~blab/Sotware/xite/*.

3. Los métodos de optimización multidimensional para la ubicación exacta del punto de esquina proporcionado por SIDEE.

La sintáxis de **capturaDosCamaras** es la siguiente:

**capturaDosCamara** *Configuración_de_Cámaras Configuración_de_Métodos*

donde:

*Configuración_de_Cámaras*: es un archivo de texto que especifica las características técnicas de la cámara que será controlada por la tarjeta Mutech. La sintáxis de este archivo está dada por el proveedor. El archivo de configuración de la cámara que se utilizó en la captura de las imágenes se localiza en disco compacto perteneciente a este trabajo, ver Apéndice B. La descripción completa de cada campo puede ser consultada en [MV-Vision, 1998].

*Configuración_de_Métodos*: es un archivo de texto que contiene la descripción de cada método de optimización que se emplea. La sintáxis es la siguiente:

*NombreMetodo* **START**
#
*NombreMetodo NombreVariable* [ **R**|**I** , $M$ ]**:** $V_k \quad k = 1, \cdots, M$
...
#
*NombreMetodo* **END**

donde:

*NombreMetodo*: es un nombre arbitrario que identifica todas las ocurrencias de un bloque llamado *NombreMetodo*.

**START**: es el inicio de un bloque llamado *NombreMetodo*. **START** debe ser incluido literalmente después de *NombreMetodo*.

**END**: es el fín de un bloque llamado *NombreMetodo*. **END** debe ser incluido literalmente después de *NombreMetodo*. La descripción de un método está contenida entre las literales **START** y **END**.

#: es un renglón de comentarios. # deberá ser escrito como primer caracter de renglón nuevo.

*NombreVariable*: es el nombre de algún vector ó variable que pertenezca al método *NombreMetodo*.

**R**: identifica a *NombreVariable* como una variable del tipo real.

**I**: identifica a *NombreVariable* como una variable del tipo entera.

$M$: es el número de elementos que contiene el vector *NombreVariable*. Si *NombreVariable* no es un vector, entonces $M = 1$. En todos los casos $M \geq 1$.

[ , ]: caracteres que delimitan la descripción de una variable. Las literales **R** e **I** son mutuamente excluyentes.

$V_k$: es el valor de cada uno de los elementos del vector *NombreVariable*.

Debido a que cada método contiene diferentes tipos de variables para su inicialización, se contruyó un archivo que permitiera modificar los valores en cada uno de los experimentos, en forma simple y sin necesidad de recompilar **capturaDosCamaras**. En la Tabla V.10 se muestra un ejemplo de este archivo. Obsérvese que se requieren variables generales para operar este programa. Tal es el caso del tamaño

de la subventana en análisis y el número de subventanas que se quiere trabajar, para las dos imágenes capturadas por la cabeza estereo. Por otro lado, el método de *Recocido Simulado* requiere de la temperatuta de inicio y el número máximo de pasos en el estado de equilibrio. El método de Nelder y Mead no requieren de estos parámetros. En su lugar, el método de *Downhill Simplex* requiere de otros tipos de parámetros en su inicialización, como es el caso de los vectores de cotas de $\vec{P}$ para la creación del simplex inicial. Es por ello, que el archivo de texto *Configuración_de_Métodos* es una herramienta versátil para manipular estas variables. Finalmente, la verificación de la sintáxis del archivo de configuración de métodos se lleva a cabo por la librería **libConfigMetodos.so**, ver página 182.

Tabla V.10: Archivo de configuración de métodos

```
# Archivo Metodos.ini
# Configuracion General. Para la apicacion de Cualquier
# Metodo. Son DOS las variables Indispensables:
# Inicio Configuracion General
ConfigGeneral START
ConfigGeneral TamanoSubvetanas [I,1]   : 17
ConfigGeneral NumeroSubventanas [I,1] : 4
ConfigGeneral END


# ************************************
# Inicio Down Hill
DownHill START
#      Cotas  del Simplex Maximo y Minimo.
#                       Sigma1 Mu1 Theta1(g) Sigma2 Mu2 Theta2(g)  A   B
DownHill Maximo [R,8] :   5.0  5.0  45.0      5.0   5.   45.0     250 255
DownHill Minimo [R,8] :   1e-05 -5.0 -45.0   1e-05 -5.0  -45.0    0    0
#
DownHill END
```

*continúa en la siguiente página*

Tabla V.10: configuración métodos (continuación)

```
#************************************
#Inicio Recocido Simulado
RecocidoSimulado START
#     Por pruebas la Temperatura de Inicio es 1
RecocidoSimulado Temperatura [I,1] : 1
#    Por pruebas el numero de pasos del estado de Equilibrio es 7
RecocidoSimulado PasosEquilibrio [I,1] : 7
RecocidoSimulado END
```

Una véz planteada la función y la sintáxis de **capturaDosCamaras** explicaremos como trabaja este algoritmo. **capturaDosCamaras** tiene la secuencia de pasos siguiente:

1. Requerimientos para ejecutar **capturaDosCamaras**.
   (a) Sistema Operativo RedHat 5.2.
   (b) Ambiente gráfico de ventanas Xwindos con $2^{16}$ colores activos.
   (c) Módulo **mv1x00kp.0**, proporcionado por el fabricante, instalado. Este módulo permite la comunicación de la tarjeta MuTech con el sistema operativo.
   (d) Librería de control de la tarjeta MuTech **libmv15.a** instalada. Esta librería es proporcionada también por el fabricante.
   (e) Librerías de XITE instaladas.
   (f) Archivos *Configuración_de_Cámaras* y *Configuración_de_Métodos* preparados.
   (g) Cámara izquierda y cámara derecha encendidas (cabeza estereoscópica).
2. Inicializa el sistema Xwindows, las cámaras y las variables de los métodos.
3. Presenta al usuario tres secciones, ver Figura 65: 1) Despliegue de la ventana de visualización de cámara derecha, 2) Despliegue de la ventana de visualización de cámara izquierda, 3) Ventana de control de procesos.

4. Las imágenes izquierda y derecha son adquiridas y presentadas en pantalla cuando el dispositivo apuntador de la computadora (ratón), se encuentra ubicado dentro de la región correspondiente. El proceso de transferencia cámara → memoria de la tarjeta de adquisición → desplegado en pantalla, se activa, si el ráton "apunta" a una ventana de despliegue de imagen. Por tanto, sólo se puede visualizar una imagen a la vez, la que corresponda a la cámara izquierda o derecha.

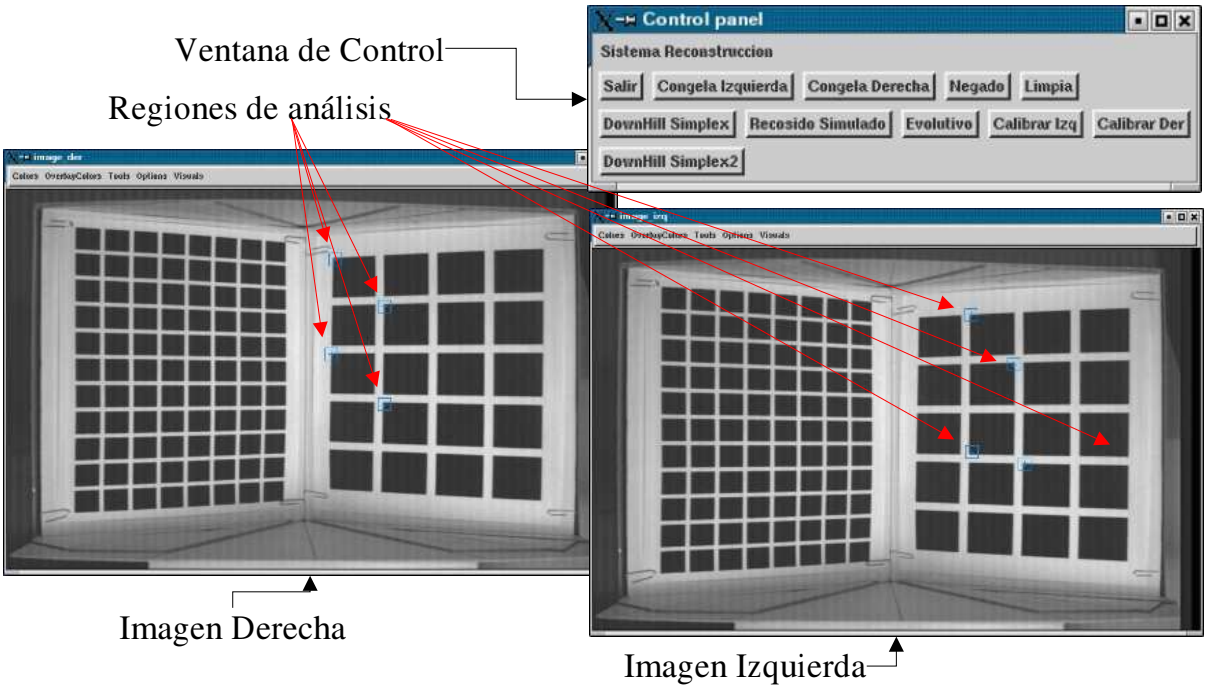



Figura 65: Panel de imágenes y control del programa **capturaDosCamaras**.

5. El panel de control está compuesto por "botones" que reprentan las funciones siguientes:

   - "Congela Izquierda": desactiva el proceso de adquisición de la cámara izquierda.
   - "Congela Derecha": desactiva el proceso de adquisición de la cámara derecha. Cuando "Congela Izquierda" y "Congela Derecha" son presionados, el usuario está en posibilidad de seleccionar las regiones que desee análizar. El número de regiones está dada en el archivo de configuración de métodos, así como el tamaño de las mismas. En la Figura 65 se muestran 4 regiones por imagen,

con un tamaño de $17 \times 17$ pixeles. Ambas variables se muestran en la Tabla V.10.

- "Limpia": se elimina cualquier región seleccionada por el usuario y se activa el proceso de adquisición nuevamente.
- "DownHill Simplex": ubicación exacta del punto de esquina a través del método *Downhill Simplex*, de las regiones seleccionadas previamente por el usuario.
- "Recocido Simulado": ubicación exacta del punto de esquina a través del método *Recocido Simulado*, de las regiones seleccionadas previamente por el usuario.
- "Evolutivo": ubicación exacta del punto de esquina a través del método *Evolutivo*, de las regiones seleccionadas previamente por el usuario.
- "Negado": invierte los valores de intensidad en tonos de gris de las imágenes.
- "Calibra Izq": calibración de la cámara izquierda.
- "Calibra Der": calibración de la cámara derecha.
- "Salir": finaliza el programa.

6. Los métodos de optimización son ejecutados si todas las regiones en análisis han sido seleccionadas. Por otro lado, cuando un método está en ejecución no se puede seleccionar alguna otra función.

7. Los resultados son presentados en la pantalla.

La arquitectura de **capturaDosCamara** es mostrada en en la Figura 66 y las rutinas que la componen se especifican en la Tabla V.11.

Tabla V.11: Rutinas asociadas al sistema **capturaDosCamaras**

| | | |
|---|---|---|
| | | **Rutinas de Control de la tarjeta Mutech** |
| V.11.1 | MV1500_inicia | Inicializa la tarjeta digitalizadora MuTech MV1500, acorde al archivo de inicialización de la cámara. |

*continúa en la siguiente página*

Tabla V.11: **capturaDosCamaras** *(continuación)*

| | | |
|---|---|---|
| | Variables de entrada | *camaraFile*: nombre del archivo de inicialización de la cámara.<br>*cualTarjeta*: número entero que identifica las tarjetas de adquisición. "0" tarjeta izquierda, "1" tarjeta derecha. |
| | Variables de salida | *pixel_mode*: pixel a color o en tonos de gris. La cámara digital PULNIX acepta únicamente tonos de gris.<br>*pixel_bits*: número de bits por pixel.<br>*pixel_bytes*: número de bytes por pixel.<br>*xsizeCam*: número de pixeles horizontales de la imagen.<br>*ysizeCam*: número de pixeles verticales de la imagen.<br>*elFrame*: apuntador a la región de memoria perteneciente a la tarjeta de adquisición, donde se encuentra la imagen capturada. |
| | Definición | int MV1500_inicia(const char *camaraFile, int cualTarjeta, int *pixel_mode, int *pixel_bits, int *pixel_bytes, int *xsizeCam, int *ysizeCam, MV1Frame *elFrame); |
| | Referencias | Manuales de la Tarjeta MuTech MV1500 [MV-Vision, 1998]. Sección B.3 |
| V.11.2 | Aparta_mem | Memoria RAM que contiene la imagen capturada por la tarjeta MV1500, en forma de vector de caracteres. |
| | Variables de entrada | $p =$: número de bytes por pixel.<br>$x$: número de pixeles horizontales de la imagen.<br>$y$: número de pixeles verticales de la imagen. |
| | Variables de salida | Dirección de la memoria RAM |
| | Definición | char *Aparta_mem(int p, int x, int y); |
| | Referencias | Sección B.3 |

*continúa en la siguiente página*

Tabla V.11: **capturaDosCamaras** *(continuación)*

| | | |
|---|---|---|
| V.11.3 | MV1500_capturaBuffer | Transferencia de la imagen cámara → memoria tarjeta MV1500 → memoria RAM. |
| | Variables de entrada | $cualTarjeta, elFrame, xsizeCam, ysizeCam$: definidas en V.11.1. |
| | Variables de salida | $master\_buffer$: imagen en memoria RAM. Si la transferencia es exitosa regresa el valor cero. |
| | Definición | int MV1500_capturaBuffer( int cualTarjeta, MV1Frame *elFrame, int xsizeCam, int ysizeCam, char *master_buffer); |
| | Referencias | Manuales de la Tarjeta MuTech MV1500 [MV-Vision, 1998]. Sección B.3 |
| V.11.4 | MV1500_cierraCamara | Cierra las cámaras PULNIX y las tarjetas MV1500. |
| | Definición | void MV1500_cierraCamara(void); |
| | Referencias | Manuales de la Tarjeta MuTech MV1500 [MV-Vision, 1998]. Sección B.3 |
| | | **Manipulación de memoria a través de XITE** |
| V.11.5 | Copia_Buffer8_banda8 | Transporta la imagen en la memoria RAM a una imagen en el formato exclusivo de XITE (Blab Image File Format BIFF). |
| | Variables de entrada | $Buf$: imagen en memoria RAM. $xs$: número de pixeles horizantales de la imagen. $ys$: número de pixeles verticales de la imagen. |
| | Variables de salida | $Ban$: imagen en formato BIFF. |
| | Definición | int Copia_Buffer8_banda8 ( char *Buf, IBAND Ban, int xs, int ys); |
| | Referencias | Manuales de XITE [Böe, 1998a], [Böe, 1998b],[Böe, 1998c]. Sección B.3 |

*continúa en la siguiente página*

Tabla V.11: **capturaDosCamaras** *(continuación)*

| | | |
|---|---|---|
| V.11.6 | Copia_Buffer8_Datos | Transporta la región seleccionada de la imagen en formato BIFF, a una matriz de datos. |
| | Variables de entrada | *Ban*: imagen en formato BIFF.<br>$x0$: posición central de la region seleccionada por el usuario sobre el eje $x$.<br>$y0$: posición central de la región seleccionada por el usuarios sobre el eje $y$.<br>$W$: tamaño de la ventana. |
| | Variables de salida | *Datos*[][]: matriz que contiene los valores de intensidad en tonos de gris de la región a análizar. La matriz *Datos*[][] contiene valores del tipo real. |
| | Definición | int Copia_Banda8_Datos(IBAND Ban, double **Datos, int x0, int y0, int W); |
| | Referencias | Manuales de XITE [Böe, 1998a], [Böe, 1998b],[Böe, 1998c]. Sección B.3 |
| | | **Panel de control** |
| V.11.7 | congela_der | Manipula el estado de la bandera global *Congela_der* que actúa sobre la cámara derecha. *Congela_der* tiene dos estados: "0" habilita el proceso de adquisición, "1" deshabilita el proceso de adquisición. En cada instante que se presiona el botón de congelación de la cámara derecha la bandera pasa al estado siguiente. |
| | Variables globales | *Congela_der* |
| | Definición | void congela_der (void); |
| | Referencias | Figura 65. Sección B.3 |

*continúa en la siguiente página*

Tabla V.11: **capturaDosCamaras** *(continuación)*

| | | |
|---|---|---|
| V.11.8 | congela_izq | Manipula el estado de la bandera global $Congela\_izq$ que actúa sobre la cámara izquierda. $Congela\_izq$ tiene dos estados: "0" habilita el proceso de adquisición, "1" deshabilita el proceso de adquisición. En cada instante que se presiona el botón de congelación de la cámara izquierda, la bandera pasa al estado siguiente. |
| | Variables globales | $Congela\_izq$ |
| | Definición | void congela_izq (void); |
| | Referencias | Figura 65. Sección B.3 |
| V.11.9 | botonazo_der | Adquisición de las coordenadas de cada región, seleccionadas con el ratón por parte del usuario. La selección se lleva a cabo, presionando el botón central o derecho del ratón sobre la imagen derecha. Así mismo, lleva un registro del número de regiones seleccionadas. Esta rutina se activa siempre que el número de regiones seleccionadas, sea menor que al número de ventanas en el archivo de configuración de los métodos. Finalmente, se "dibuja" un cuadrado alrededor del punto seleccionado de longitud $W$. *contador_der*: número de ventanas seleccionas. Se incrementa en uno cada vez que se presiona el botón del ratón. |
| | Variables globales | $Puntos[k][]$: matriz de coordenadas $(x, y)$ de cada punto. En particular $Puntos[k][1] = x$ y $Puntos[k][2] = y$, para $k = 1, \cdots, Np$ puntos sobre la imagen. $Np$ es el número total de ventanas. |
| | Definición | static void botonazo_der( ImageCallback button) |
| | Referencias | Figura 65. Sección B.3 |
| V.11.10 | botonazo_izq | Simliar a V.11.9, aplicado a la imagen izquierda. |

*continúa en la siguiente página*

Tabla V.11: **capturaDosCamaras** *(continuación)*

| | | |
|---|---|---|
| | Variables globales | *contador_izq*: número de ventanas seleccionas. Se incrementa en uno cada vez que se presiona el botón del ratón.<br>$Puntos[k][]$: matriz de coordenadas $(x, y)$ de cada punto. En particular $Puntos[k][1] = x$ y $Puntos[k][2] = y$, para $k = Np+1, \cdots, 2Np$ puntos sobre la imagen. $Np$ es el número total de ventanas. Notese, que la matriz $Puntos[][]$ contiene los puntos de la imagen derecha e izquierda. La distinción se efectúa por la inspección del subíndice $k$. |
| | Definición | static void botonazo_izq( ImageCallback button) |
| | Referencias | Figura 65. Sección B.3 |
| V.11.11 | Down_hill2 | Obtención del punto exacto de esquina-L a partir del método *Downhill Simplex*, para todas las regiones seleccionadas por el usuario. Los parámetros de inicialización son extraídos del archivo de configuración de métodos. |
| | Variables globales | $Datos[k][][]$: es un cubo de datos que contiene los valores de intensidad de todas las regiones, alrededor de los puntos $Puntos[k][]$ seleccionados por el usuarios. Donde $k = 1, \cdots, Np$ para las regiones de la imagen derecha y $k = Np+1, \cdots, 2Np$ las correspondientes a la imagen izquierda. $Np$ es el número total de puntos por imagen.<br>$Punto\_exacto[k][]$: coordenadas de la ubicación exacta de esquina-L en el sistema coordenado del modelo para la *k-ésima* estructura. |

*continúa en la siguiente página*

Tabla V.11: **capturaDosCamaras** *(continuación)*

| | | |
|---|---|---|
| | Variables globales | $Punto[k][]$: coordenadas de la ubicación exacta de esquina-L en el sistema coordenado de la imagen para la *k-ésima* estructura.<br>$Signo\_x[k][] = s_{U_1}$: signo del eje $x$ en la dirección de apertura de la *k-ésima* estructura.<br>$Signo\_y[k][] = s_{U_2}$: signo del eje $y$ en la dirección de apertura de la *k-ésima* estructura.<br>$Param[k][]$: $\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B$ parámetros del modelo $M_L(x, y, \vec{P})$ ajustados a la *k-ésima* estructura. |
| | Definición | void Down_hill2() |
| | Referencias | Tablas V.5, V.6, V.12. Sección B.3 |
| V.11.12 | RecocidoSimulado | Obtención del punto exacto de esquina-L a partir del método *Recocido Simulado*, para todas las regiones seleccionadas por el usuario. Los parámetros de inicialización son extraídos del archivo de configuración de métodos. En particular, la generación del simplex inicial del esquema de equilibrio es aleatorio. |
| | Variables globales | $Datos[k][][], Punto\_exacto[k][], Punto[k][], Signo\_x[k][],$ $Signo\_y[k][], Param[k][]$ están definidos en V.11.11 |
| | Definición | void RecocidoSimulado() |
| | Referencias | Tablas V.6, V.7, V.12. Sección B.3 |
| V.11.13 | RecocidoSimulado_iso | Similar a V.11.12, en donde la generación del simplex inicial del esquema de equilibrio, es una muestra uniforme entre las cotas superior e inferior de los parámetros del modelo. |
| | Variables globales | $Datos[k][][], Punto\_exacto[k][], Punto[k][], Signo\_x[k][],$ $Signo\_y[k][], Param[k][]$ están definidos en V.11.11 |
| | Definición | void RecocidoSimulado_iso() |
| | Referencias | Tablas V.6, V.7, V.12. Sección B.3 |

*continúa en la siguiente página*

Tabla V.11: **capturaDosCamaras** *(continuación)*

| | | |
|---|---|---|
| V.11.14 | Evolutivo | Obtención del punto exacto de esquina-L a partir del método *Evolutivo*, para todas las regiones seleccionadas por el usuario. Los parámetros de inicialización son extraídos del archivo de configuración de métodos. La población inicial es muestreada uniformemente. |
| | Variables globales | $Datos[k][][]$, $Punto\_exacto[k][]$, $Punto[k][]$, $Signo\_x[k][]$, $Signo\_y[k][]$, $Param[k][]$ están definidos en V.11.11 |
| | Definición | void Evolutivo() |
| | Referencias | Tablas V.6, V.8, Sección B.3 |
| V.11.15 | Limpia | Elimina las regiones seleccionadas por el usuario y vuelve a desplegar las imágenes (derecha e izquierada) sin dichas marcas. |
| | Variables globales | $Congela\_der = 0$<br>$Congela\_izq = 0$<br>$contador\_der = 0$<br>$contador\_der = 0$ |
| | Definición | void Limpia() |
| | Referencias | Sección B.3 |
| V.11.16 | ximage_work | Funcionalidad de las instancias (creación y manipulación del ambiente XWindos) y procedimientos (rutinas anteriores) acorde a las librerías de XITE. |
| | Definición | static void ximage_work( Widget wid, XtPointer client_data, XtPointer call_data) |
| | Referencias | Manuales de XITE [Böe, 1998a], [Böe, 1998b],[Böe, 1998c]. Sección B.3 |

### V.4.1.2 Librería libConfigMetodos.so

La librería **libConfigMetodos.so** es la responsable de la sintáxis del archivo de configuración de los métodos. Las rutinas que componen esta librería se muestran en la

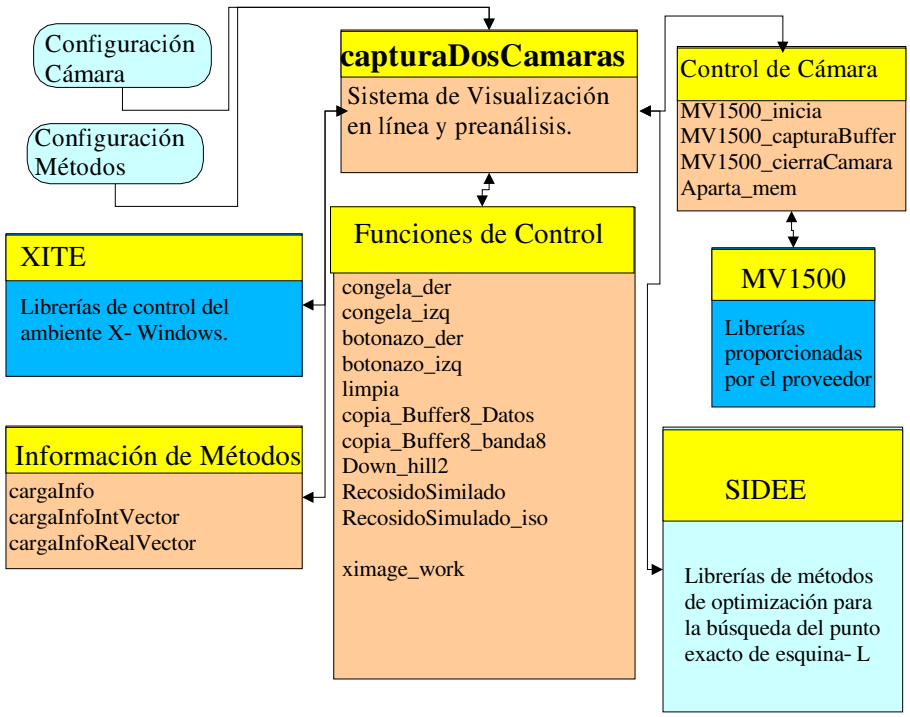



Figura 66: Arquitectura de **capturaDosCamaras**.

Tabla V.12. Una vez que se ha revisado la sintáxis, **libConfigMetodos.so** genera una estructura definida como sigue:

```
struct   infoMetodos   {
         char          metod[30];
         cha           var[30];
         char          tipo[2];
         int           element;
         char          resline[100];
    };
```

donde:

*metod*[30]: nombre del método.

*var*[30]: nombre de la variable que corresponde al método.

*tipo*: contiene una **R** para valores reales o una **I** si la variables es entera.

*element*: número de elementos de la variable.

*reslin*[100]: valores de cada elemento de la variable.

La estructura **infoMetodos** contiene un renglón válido del archivo de configuración de los métodos. Un renglón válido es aquel que obedece a la sintáxis antes propuesta y no es un comentario. Un método se define completamente con un vector que contiene tantos elementos del tipo **infoMetodos** como renglones válidos existan entre las literales **START** y **END** en el archivo de configuración. Finalmente, los valores de las variables son extraídos de este vector, para ser operados por **capturaDosCamaras**. La arquitectura de **libConfigMetodos.so** se muestra en la Figura 67.

Tabla V.12: Rutinas asociadas a **libConfigMetodos.so**

| | | |
|---|---|---|
| V.12.1 | saltaCaracter | Adelanta el apuntador *pos* tantas posiciones como caracteres *Car* consecutivos existan en el renglón del archivo de configuración. Generalmente se emplea para eliminar espacios en blanco en cada línea. |
| | Variables de entrada | *Linea*[]: renglón completo del archivo de configuración.<br>*pos*: posición actual del apuntador dentro de *Linea*[].<br>*Car*[]: caracter a evaluar. |
| | Variables de salida | Posición final del apuntador, una vez saltados los caracteres consecutivos *Car*. |
| | Definición | int saltaCaracter( char Car[2], char Linea[], int *pos) |
| | Referencias | Seción B.3 |
| V.12.2 | nextReadCampo | Sintáxis de cada renglón del archivo de configuración de métodos. Lee una ocurrencia de cada renglón y regresa a dicha ocurrencia sin caracteres especiales. Los caracteres especiales son:<br>" " : barra espaciadora ó espacio en blanco.<br>"[" : paréntesis cuadrado abierto.<br>"]" : paréntesis cuadrado cerrado.<br>"," : coma.<br>":" : dos puntos. |

*continúa en la siguiente página*

Tabla V.12: **libConfigMetodos.so** *(continuación)*

| | | |
|---|---|---|
| | Variables de entrada | *Linea*[]: renglón completo del archivo de configuración. *campos*: revisa la sintáxis del campo que se encuentra analizando. Los casos son: 1 = *metod*, 2 = *var*, 3 = *tipo*, 4 = *element* y 5 = *resline*. Estos valores corresponden a la estructura **infoMetodos**. *pos*: posición actual del apuntador dentro de *Linea*[]. Cuando termina esta rutina *pos* es la posición final despúes de leer la ocurrencia. |
| | Variables de salida | *elcampo*: valor de la ocurrencia sin caracteres especiales. |
| | Definición | char *nextReadCampo( char Linea[], int campos, int *pos, char *elcampo) |
| | Referencias | Seción B.3 |
| V.12.3 | cargaInfo | Carga la información completa de un método en el vector $infoM[]$. $infoM[]$ es un vector con la estructura **infoMetodos**. Revisa que la información del método esté contenida entre las literales **START** y **END**. |
| | Variables de entrada | *Arch*: archivo de configuración de métodos. *cadena*: nombre del método. |
| | Variables de salida | $infoM[]$: vector de información del método. Regresa el número de elementos del vector $infoM[]$. |
| | Definición | int cargaInfo(FILE *Arch, char *cadena, struct infoMetodos *infoM); |
| | Referencias | Seción B.3 |
| V.12.4 | cargaInfoIntVector | Extracción del valor de una variable entera definida en el vector $infoM[]$. |
| | Variables de entrada | $laInfo$: elemento del vector $infoM[]$, que contiene una variable del tipo entera. |
| | Variables de salida | *intvector*: valor tipo entero. |
| | Definición | int cargaInfoIntvector(struct infoMetodos laInfo, int *intvector); |

*continúa en la siguiente página*

Tabla V.12: **libConfigMetodos.so** *(continuación)*

| | | |
|---|---|---|
| | Referencias | Seción B.3 |
| V.12.5 | cargaInfoRealVector | Extracción del valor de una variable tipo real definida en el vector $infoM[]$. |
| | Variables de entrada | $laInfo$: elemento del vector $infoM[]$, que contiene una variable del tipo real. |
| | Variables de salida | *realvector*: valor tipo entero. |
| | Definición | int cargaInfoRealvector(struct infoMetodos laInfo, double *realvector); |
| | Referencias | Seción B.3 |

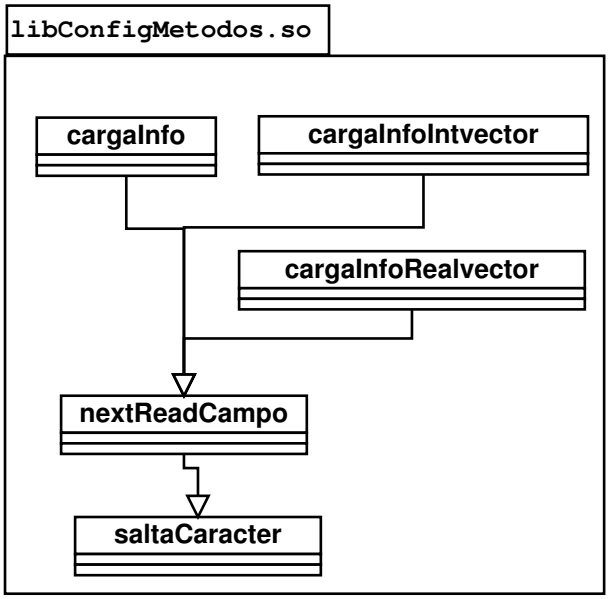



Figura 67: Rutinas que componen **libConfigMetodos.so**.

## V.4.2 Programa capturaDosTest

El programa **capturaDosTest** se generó con la intención de poder ejecutar a **capturaDosCamaras** en plataformas computacionales distintas. Esto es, **capturaDosTest**

se desarrollo con las características siguientes: 1) no se requiere que la tarjeta digitilizadora MuTech MV1500 y sus librerías estén instaladas, 2) **capturaDosTest** no adquiere imágenes, en su lugar despliega dos archivos de imágenes en formato BIFF, las cuales corresponden a la imagen derecha e izquierda, 3) no se requiere tener instalado el sistema operativo RedHat 5.2, 4) se necesita que las librerías de XITE esten compiladas en el sistema operativo donde se ejecutará **capturaDosTest**. Fuera de las características antes citadas, **capturaDosTest** y **capturaDosCamaras** tienen la misma funcionalidad. La sintáxis de **capturaDosTest** es la siguiente:

**capturaDosTest** *Image_Derecha Image_Izquierda Configuración_de_Métodos*
donde:

*Image_Derecha*: es un archivo de imagen en formato BIFF, que será desplegada en la ventana de visualización de la cámara derecha, ver Figura 65.

*Image_Izquierda*: es un archivo de imagen en formato BIFF, que será desplegada en la ventana de visualización de la cámara izquierda, ver Figura 65.

*Configuración_de_Métodos*: es el archivo de configuración de los métodos, ver Tabla V.10.

### V.4.3 Programa Sensibilidad

Genera una tabla del comportamiento de la ubicación exacta del punto de esquina $(x_e, y_e)$ para diferentes variaciones de los parámetros del modelo. **Sensibilidad** se generó como un programa que acepta una variedad de "opciones" empleando el estilo de los comandos del sistema operativo UNIX. La sintáxis es la siguiente y debe escribirse en el estricto orden que se muestra:

**Sensibilidad** -m *Archivo_de_salida* -w *Tamano_Ventana* -v *Nombre_Parametro*
-r *Cota_inferior* -R *Cota_superior* -i *Incremento* -o *Archivo_de_Salida*
-c *Parametros_del_Modelo*

donde:

- -m *Archivo_de_salida*: es el prefijo del nombre del archivo de resultados. El nombre del archivo de resultados es *Archivo_de_salida*.**res**.
- -w *Tamano_Ventana*: es el tamaño de la ventana.

- -v *Nombre_Parametro*: es el nombre del parámetro que se desee variar en un rango dado por las opciones "-r" y "-R". Los valores válidos que puede tomar *Nombre_Parametro* son los siguientes:

  **sigma1**: muestreo de $\sigma_1$.

  **sigma2**: muestreo de $\sigma_2$.

  **sigma12**: muestreo simultáneo de $\sigma_1$ y $\sigma_2$, donde $\sigma_1 = \sigma_2$.

  **mu1**: muestreo de $\mu_1$.

  **mu2**: muestreo de $\mu_2$.

  **theta1**: muestreo de $\vartheta_1$.

  **theta2**: muestreo de $\vartheta_2$.

  **A**: muestreo de $A$.

  **B**: muestreo de $B$.

- -r *Cota_inferior*: valor de la cota inferior del parámetro que se desee muestrear. Los ángulos $\vartheta_1$ y $\vartheta_2$ están expresados en grados.

- -R *Cota_superior*: valor de la cota superior del parámetro que se desee muestrear. Los ángulos $\vartheta_1$ y $\vartheta_2$ están expresados en grados.

- -i *Incremento*: tamaño de cada muestra.

- -o *Archivo_de_Salida*: nombre del archivo que contiene la matriz $M'_L(x, y, \vec{P})$.

- -c *Parametros_del_Modelo*: nombre del archivo que contiene los $\vec{P}$ parámetros inciales de $M'_L(x, y, \vec{P})$. Donde $\vec{P} = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B, s_{U_1}, s_{U_2})$. Este archivo debe contener una línea, con los valores de los paramétros escritos en el estricto orden de $\vec{P}$.

El programa **Sensibilidad** funciona como sigue:

1. Asigna *Cota_inferior* a $\vec{P}(i)$. Donde $i$ es el *i-ésimo* elemento de $\vec{P}$. $i$ depende de la opción "-v".

2. Evalúa el modelo $M'_L(x, y, \vec{P})$ en los $\vec{P}$ parámetros iniciales diferentes a la asignación expresada en el paso 1. $M'_L(x, y, \vec{P})$ es evaluada en una ventana de longitud definida por la opción -w.

3. Localiza el punto exacto de esquina $(x_e, y_e)$, a partir de nuestro criterio de ubicación exacta de esquina $Q$.

4. Almacena los valores de $\vec{P}$ y $(x_e, y_e)$ en el archivo de resultados, ver Tabla V.9.

5. Almacena la matriz $M'_L(x, y, \vec{P})$ en el archivo de salida.

6. Incrementa el valor del parámetro $\vec{P}(i)$ acorde a valor de la muestra dada por la opción "-i". Esto es, $\vec{P}(i) = P(i) + Incremento$. Si el parámetro de muestro es menor a *Cota_superior* entonces se ejecuta el paso 2, de otra forma termina el programa.

El programa **Sensibilidad** se utilizó para obtener los datos de la Figura 36 y 37. En dichas figuras se generó una esquina centrada en el origen y los valores iniciales de $\vec{P}$ mostrados en la Tabla V.13. Los datos mostrados en dichas figura, son el resultado de la ejecución de **Sensibilidad** variando $\vartheta_1$ en el rango $[-89^o, 89^o]$. Por ejemplo, para obtener dichas figuras, **Sensibilidad** se ejecutó como sigue:

**Sensibilidad** -m angulo -w 21 -v theta1 -r -89 -R 89 -i 1.0 -o Angulos -c Parametros.txt

Tabla V.13: Valores inicales de $\vec{P}$ utilizados en la obtención las figuras 36 y 37. Cada una de los renglones de esta tabla corresponden a distintos archivos de valores iniciales de $\vec{P}$.

| $\sigma_1$ | $\mu_1$ | $\vartheta_1$ | $\sigma_2$ | $\mu_2$ | $\vartheta_2$ | $A$ | $B$ | $s_{U_1}$ | $s_{U_2}$ |
|---|---|---|---|---|---|---|---|---|---|
| 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 1.0 |
| 1.313229 | 0.0 | 0.0 | 1.313229 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 1.0 |
| 1.5 | 0.0 | 0.0 | 1.5 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 1.0 |

### V.4.4 Modelo_L_DownHill

Detección del punto exacto de esquina $(u_e, v_e)$ a través del método de minimización multidimensional *Down Hill Simplex*. La sintáxis es la siguiente:

**Modelo_L_DownHill** -f *Nomber_Imagen* -x *CoordenadaX* -y *CoordenadaY*
-w *Tamano_Ventana* -o *Salida_Maple* -C *Archivo_Cotas_del_Simplex*
donde:

- -f *Nomber_Imagen*: es el nombre del archivo de imagen en tonos de gris. La imagen debe tener el formato "Portable Gray Map" (PGM). PGM es un formato soportado por las librerías de IUE. *Nomber_Imagen* es también el prefijo del nombre del archivo de resultados.
- -x *CoordenadaX*: coordenada inicial $u_0$.
- -y *CoordenadaY*: coordenada inicial $v_0$.
- -w *Tamano_Ventana*: tamaño de la región en análisis, expresada en pixeles.
- -o *Salida_Maple*: Archivo que contiene la matriz de los valores de intensidad de la región original. El archivo tiene la extensión **.maple**, por tanto el nombre completo es *Salida_Maple***.maple**. Esta matriz se utiliza para sobreponer la región de la imagen original en estudio y $M'_L(x, y, \vec{P})$ después del ajuste, apoyandonos con el paquete de matemática simbólica MAPLE, vea por ejemplo la Figura 6.f.
- -C *Archivo_Cotas_del_Simplex*: archivo que contiene la cota superior e inferior de los $\vec{P}$ parámetros iniciales del simplex. El primer renglón deberá contener la cota superior y el segundo renglón la cota inferior. Los parámetros deben ser escritos en el orden siguiente: $\sigma_1\ \mu_1\ \vartheta_1\ \sigma_2\ \mu_2\ \vartheta_2\ A\ B$.

El funcionamiento de **Modelo_L_DownHill** obedece al lógica expresada en el algoritmo general de búsqueda mostrado en la Figura 55.

### V.4.5 Modelo_L_DownHill_ruido

El programa **Modelo_L_DownHill_ruido** se generó para observar el comportamiento del punto exacto de esquina $(u_e, v_e)$, a través del método de optimización propuesto por Nelder y Mead, en presencia de un ruido Gaussiano. Este programa es similar a **Modelo_L_DownHill** con las modificaciones adicionales siguientes:

1. Se genera un ruido Gaussiano con una varianza unitaria y una media cero, a partir de una distribución aleatoria uniforme. Para ello, se emplea la rutina **gasdev**

propuesta por Press *et. al.* [Press, 1998]. El valor propuesto por **gasdev** $\rho_i$ es escalado por un factor $\lambda$, proporcionado por el usuario.

2. A cada uno de los pixeles de la imagen original $I(u,v)$ se les adiciona un ruido Gaussiano $\rho_i$. $\rho_i$ se calcula para cada pixel, tal que la imagen con ruido $I_{ruido}(u,v)$ se expresa como:

$$I_{ruido}(u,v) = I(u,v) + \lambda\rho_i \tag{V.1}$$

3. $I_{ruido}(u,v)$ es normalizada en el rango de valores de intensidad de $[0,255]$ y se guarda $I_{ruido}(u,v)$ en otro archivo de imagen. Este paso se requiere, debido a que las perturbaciones provocadas por el ruido Gaussiano escalado pueden provocar que $I_{ruido}(u,v)$ contenga valores fuera del rango permitido de escala de gris.

4. Para medir las perturbaciones provocadas por el ruido Gaussiano escalado, se calcula la *relación señal a ruido*. La relación señal a ruido es calculada sólo en la región que se desea analizar. Las matrices $\mathbf{D_{ruido}}$ y $\mathbf{D}$ denotan a la región de interés de $I_{ruido}(u,v)$ y $I(u,v)$ respectivamente. La cantidad del error producido por el ruido Gaussiano escalado se expresa por $\mathbf{E} = \mathbf{D_{ruido}} - \mathbf{D}$. A partir de estas consideraciones, la relación señal a ruido expresada en decibeles $(_{DB})$, se calcula como sigue:

$$\begin{aligned}
SR_{DB} &= 10log\left(\frac{S^2}{S^2_{ruido}}\right) \\
S^2 &= \frac{1}{n^2-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(\mathbf{D}(i,j)-\overline{\mathbf{D}})^2 \\
S^2_{ruido} &= \frac{1}{n^2-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(\mathbf{E}(i,j)-\overline{\mathbf{E}})^2 \\
\overline{\mathbf{D}} &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{D}(\mathbf{i},\mathbf{j}) \\
\overline{\mathbf{E}} &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{E}(i,j)
\end{aligned} \tag{V.2}$$

donde: $SR_{DB}$ es el valor de la relación señal a ruido, en decibeles, formada por el cociente de la varianza de la imagen perfecta $S^2$ y la varianza del error $S^2_{ruido}$.

5. El valor $SR_{DB}$ es almacenado en el archivo de resultados en el "Campo5" y el factor de escalamiento $\lambda$ en el "Campo4", ver Tabla V.9.

6. Finalmente, $(x_e, y_e)$ se localiza a través de la estructura almacenada en la matriz $\mathbf{D_{ruido}}$.

La sintáxis de este programa es la siguiente:

**Modelo_L_DownHill_ruido** -f *Nomber_Imagen* -x *CoordenadaX* -y *CoordenadaY* -w *Tamano_Ventana* -s *factor_gauss* -n *Numero_realizaciones* -o *Imagen_conRuido* -g *Numero_pasos* -C *Archivo_Cotas_del_Simplex*

donde:

- Las opciones "-f", "-x", "-y", "-w" y "-C" son similares al programa **Modelo_L_DownHill**.
- -s *factor_gauss*: factor de escalamiento del ruido Gaussiano $\lambda$.
- -n *Numero_realizaciones*: es el número de veces que se calcula el punto exacto de esquina, para un valor de $\lambda$ dado.
- -o *Imagen_conRuido*: tiene dos funciones: 1) es el nombre del archivo de la imagen con ruido Gaussiano en formato PGM, 2) es el nombre del archivo *Imagen_conRuido***.maple** que contiene los valores de intensidad de la región en estudio, para ser empleada por Maple, ver sección V.4.4.
- -g *Numero_pasos*: número máximo de pasos que puede dar el *simplex*.

### V.4.6 Modelo_L_DownHill_gauss

El programa **Modelo_L_DownHill_gauss** convoluciona la imagen original a través de un filtro Gaussino bidimensional $G(i,j)$. A partir de la imagen convolucionada aplica la estrategia de Nelder y Mead para localizar el punto exacto de esquina-L $(u_e, v_e)$. $G(i,j)$ es un filtro que suaviza el piso y la cresta de la esquina-L. Por tanto, los bordes de la esquina-L estan mejor perfilados. Sin embargo, $G(x,y)$ provoca un dezplazamiento del punto exacto por donde pasa la esquina [Rohr, 1992]. **Modelo_L_DownHill_gauss** se generó para observar el comportamiento de $M'_L(x, y, \vec{P})$ aplicando este tipo de filtro. $G(i,j)$ se define como:

$$G(i,j) = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{i^2+j^2}{2\sigma^2}\right)} \quad i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, n \tag{V.3}$$

donde $n$ es el tamaño del núcleo[2] de convolución y $\sigma$ es el factor de suavizado. $\sigma$ no tiene relación con los factores de difuminado $\sigma_1$ y $\sigma_2$ de $M'_L(x, y, \vec{P})$. La sintáxis de **Modelo_L_DownHill_gauss** es la siguiente:

**Modelo_L_DownHill_gauss** -f *Nomber_Imagen* -x *CoordenadaX* -y *CoordenadaY* -w *Tamano_Ventana* -s *sigma_Kernel_gauss* -o *Imagen_convolucionada* -k *Tamano_del_kernel* -C *Archivo_Cotas_del_Simplex*

donde:

- las opciones -f, -x, -y, -w y -C son similares al programa **Modelo_L_DownHill_ruido**, ver sección V.4.5.
- -s *sigma_Kernel_gauss*: equivale a $\sigma$ de filtro Gaussiano, ver Ecuación (V.3).
- -k *Tamano_del_kernel*: es el tamaño $n$ del filtro Gaussiano.

### V.4.7 Modelo_L_Recocido

Detección del punto exacto de esquina $(u_e, v_e)$ a través del método de optimización multidimensional *Recocido Simulado*. El programa **Modelo_L_Recocido** detecta el punto $(u_e, v_e)$ a partir de un *simplex* inicial aleatorio. Esto es, hace un llamado a la rutina **Recocido_Simulado_L**, ver Tabla V.7.1. Este programa se empleó en la generación de las figuras 46, 47 y 48. El funcionamiento de **Modelo_L_Recocido** obedece a lógica expresada en el algoritmo general de búsqueda mostrada en la Figura 55. La sintáxis es la siguiente:

**Modelo_L_Recocido** -f *Nomber_Imagen* -x *CoordenadaX* -y *CoordenadaY* -w *Tamano_Ventana* -T *Temperatura_de_Inicio* -I *Numero_de_Pasos_Iniciales*

donde:

- las opciones -f, -x, -y, -w son similares al programa **Modelo_L_DownHill_ruido**, ver sección V.4.5.
- -T *Temperatura_de_Inicio*: temperatura inicial del esquema de enfriamiento del *simplex*.
- -I *Numero_de_Pasos_Iniciales*: número máximo de pasos en el estado de equilibrio.

[2] conocido en inglés como *kernel*

### V.4.8 Modelo_L_Recocido_iso_ruido

Detección del punto exacto de esquina $(u_e, v_e)$ a través del método de optimización multidimensional *Recocido Simulado*, a través de un *simplex* inicial uniformemente distribuido, entre las cotas superior e inferior de los $\vec{P}$ parámetros. Esto es, hace un llamado a la rutina **Recocido_Simulado_L_iso** de la librería **libRecocido_Simulado_L.so**, ver Tabla V.7.2. **Modelo_L_Recocido_iso_ruido** agrega un ruido Gaussiano a cada una de las observaciones, de manera similar que **Modelo_L_DownHill_iso_ruido**, ver sección V.4.5. La sintáxis en la siguiente:

**Modelo_L_Recocido_iso_ruido** -f *Nomber_Imagen* -x *CoordenadaX* -y *CoordenadaY* -w *Tamano_Ventana* -s *factor_gauss* -n *numero_muestra* -T *Temperatura_de_Inicio* -I *Numero_de_Pasos_Iniciales* -g *Numero_pasos* -C *Archivo_de_Cotas*

donde:

- Las opciones -f, -x, -y, -w, -s, -n, -g, -C son similares al programa **Modelo_L_DownHill_ruido**, ver sección V.4.5.
- Las opciones -T, -I son similares al programa **Modelo_L_Recocido**, ver sección V.4.7.

### V.4.9 Modelo_L_Evolutivo

Detección del punto exacto de esquina $(u_e, v_e)$ a través de la estrategia *Evolutiva*. El programa **Modelo_L_Evolutivo** calcula $(u_e, v_e)$ a partir de una población uniformemente distribuida entre las cotas superior e inferior de los $\vec{P}$ parámetros. Así mismo, este programa genera un ruido Gaussiano aleatorio sobre todas las observaciones de manera similar que **Modelo_L_DownHill_ruido**, ver sección V.4.5. El funcionamiento de **Modelo_L_Evolutivo** obedece a la lógica expresada en el algoritmo general de búsqueda mostrada en la Figura 55. La sintáxis es la siguiente:

**Modelo_L_Evolutivo** -f *Nomber_Imagen* -x *CoordenadaX* -y *CoordenadaY* -w *Tamano_Ventana* -s *factor_gauss* -n *numero_muestra* -p *poblacion* -a *porcentaje_apareamiento* -m *porcentaje_mutacion* -g *generaciones* -c *porcentaje_convergencia* -C *Archivo_de_Cotas*

donde:

- Las opciones -f, -x, -y, -w, -s, -n, -C son similares al programa **Modelo_L_DownHill_ruido**, ver sección V.4.5.
- -p *poblacion*: número de individuos de la población.
- -a *porcentaje_apareamiento*: porcentaje de apareamiento o cruzamiento. Este porcentaje se expresa en el rango de $[0, 1]$.
- -m *porcentaje_mutacion*: porcentaje de mutación. Este porcentaje se expresa en el rango de $[0, 1]$.
- -g *generaciones*: número máximo de genereaciones.
- -c *porcentaje_convergencia*: porcentaje de convergencia de los individuos de una población. Esta variable determina la terminación del proceso evolutivo.

## V.5 Conclusiones

En este capítulo se ha planteado la infraestructura computacional necesaria para localizar el punto exacto por donde pasa una esquina-L en una imagen digital. Este proceso se realiza a través de nuestro modelo de esquina $M'_L(x, y, \vec{P})$ y tres estrategias de optimización distintas: *Down Hill Simplex*, *Recocido Simulado* y *Cómputo Evolutivo*.

La infraestructura se ha organizado en dos grandes secciones: 1) un conjunto de librerías dinámicas compartidas que contiene la descripción completa y evaluación de nuestro modelo análitico de esquina y a los algoritmos de optimizacón descritos previamente y 2) un conjunto de programas que hacen uso de estas librerías, los cuales se encargan de presentar al usuario una imagen digital adquirida previamente o en tiempo real, por un sistema de adquisición de imágenes, o bién, efectuar cálculos masivos sobre dichas imágenes. Esta forma de organización permite:

1. Aislar los métodos de optimización, la evaluación del modelo y sus derivadas, y a nuestro criterio de ubicación precisa de esquina-L, de la complejidad del manejo y adquisición de una imagen digital. Por ejemplo: 1) el manejo de distintos formatos de una imagen digital, 2) los mecanísmos de desplegado de una imagen en el ambiente X-Windows, 3) el manejo de las tarjetas de adquisición en la obtención de escenas del mundo real.

2. Segmentar las funciones o rutinas tal que puedan ser usadas desde diferentes programas (código reusable).

3. Agreagar, rutinas o funciones que permitan nuevas habilidades de SIDEE.

Finalmente, en la dos tablas siguientes se muestra un resumen cuantitativo del trabajo realizado en el desarrollo de esta Tesis.

Tabla V.14: Resume cuantitativo de ***SIDEE***.

| SIDEE | | |
|---|---|---|
| Librería | No. Rutinas | No. Lineas |
| **libfunciones.so** | 17 | 467 |
| **libderivadas_L.so** | 8 | 219 |
| **libCriterio_ch2_L.so** | 2 | 129 |
| **libmisnr.so** | [3](9) | — |
| **libDown_hill_L.so** | 3 | 272 |
| **libModelo_L_Curvatura.so** | 3 | 274 |
| **libDown_Recocido_Simulado_L.so** | 4 | 388 |
| **libEvolutivo_L.so** | 4 | 594 |
| Total | 41 | 2342 |

[3]No desarrolladas en el presente trabajo

Tabla V.15: Resume cuantitativo de los *Programas de Explotación*.

| Programas de Explotación | | |
|---|---|---|
| Programa | No. Rutinas | No. Lineas |
| **capturaDosCamaras**<br>**capturaDosTest** | 16 | 623 |
| **libdConfigMetodos.so** | 5 | 224 |
| **Sensibilidad** | — | 261 |
| **Modelo_L_DownHill** | — | 302 |
| **Modelo_L_DownHill_ruido** | — | 637 |
| **Modelo_L_DownHill_gauss** | — | 368 |
| **Modelo_L_Recocido** | — | 279 |
| **Modelo_L_Recocido_iso_ruido** | — | 599 |
| **Modelo_L_Evolutivo** | — | 633 |
| Total | 21 | 3926 |

# Capítulo VI

# Experimentación y Conclusiones

## VI.1 Introducción

En los capítulos anteriores se plantearon las ecuaciones que definen a nuestro nuevo modelo paramétrico de esquinas múltiples, el cual simula las variaciones de intensidad en tonos de gris de una región perteneciente a una imagen digital. Una esquina-L es modelada completamente por la expresión $M'_L(x, y, \vec{P})$, Ecuación (III.9). $M'_L(x, y, \vec{P})$ es una función paramétrica que debe ser ajustada a una región de la imagen, por lo tanto, nuestro problema se transforma en un *problema de optimización*. Esto es, debemos encontrar el mejor conjunto de valores $\vec{P} = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$ que se ajusten a una región de la imagen $I(u, v)$. En este sentido, se describieron tres métodos de optimización multidimensional para modelos no lineales. Estos métodos son: 1) la estrategia *Nelder y Mead*, 2) la estrategia *Recocido Simulado* y 3) la estrategia *Evolutiva*. Así mismo, se definió a $Q$, Ecuación (IV.9), como nuestro estimador de máxima probabilidad. Este estimador garantiza que los $\vec{P}$ parámetros encontrados una vez aplicados los métodos de optimización, sean los que representen mejor las variaciones de intensidad de $I(u, v)$. $Q$ representa nuestro criterio de optimización o función de aptitud.

Así mismo, se ha planteado la arquitectura del sistema a través de la cual obtenemos la ubicación del punto exacto de esquina-L a un nivel de **subpixel**. Este sistema está basado en un conjunto de librerías que aglutinan las características físicas y geométricas de nuestro modelo y los métodos de optimización antes citados, el cual fue llamado *Sistema Integral para la Detección Exacta de Esquinas* (SIDEE). Adicionalmente, se

codificaron un conjunto de programas que hacen uso de las caractéricas de SIDEE, que llamamos *Programas de Explotación*.

En este capítulo, se presentan las imágenes reales y sintéticas utilizadas para probar a nuestro modelo de esquina-L y las ruitinas codificadas en SIDEE. Para efectos de probar la robustez de nuestro detector, se aplicó ruido aleatorio a las imágenes sintéticas. Adicionalmente, se efectúa una comparación entre los métodos de optimización multidimensional *Down Hill Simplex*, *Recocido Simulado* y *Evolutivo*, a fin de determinar cual de ellos obedece mejor a nuestras necesidades. Finalmente se presentarán las conclusiones generales del trabajo.

Este capítulo está organizado en las secciones siguientes:

*Experimentación.* Se presentan las tablas y los gráficos de las imágenes empleadas en la prueba de SIDEE y nuestro detector de esquinas múltiples. Así mismo, se muestra la eficiencia de los tres métodos de optimización multidimensional, en términos del número de pasos requeridos y precisión, en la obtención del punto exacto de una esquina-L. Finalmente, se agrega un ruido aleatorio Gaussiano a una imagen sintética y se observa la robustez de estas estrategias de optimización.

*Conclusiones.* Se presentan las conclusiones generales del trabajo, así como los trabajos futuros que pueden generase del mismo.

## VI.2 Experimentación

Una vez que se ha discutido un modelo análitico nuevo de esquinas y un conjunto de técnicas de optimización multidimensional, es de nuestro interés conocer:

1. Como se comportan los $\vec{P} = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$ parámetros de nuestro modelo analítico, en relación con la ubicación del punto exacto de esquina. Recuerde, que la ubicación de $(x_e, y_e)$ está dada por nuestro criterio de ubicación de esquinas-L, ver sección III.4.

2. Si los métodos de optimización codificados son consistentes.

3. Cual es el rendimiento de los tres métodos de optimización.

4. Cual es la precesición de dichos métodos.

5. Cual es la robutez de los mismos.

Para ello, se plantean una serie de experimentos que apoyen las conclusiones que mostraremos en las subsecuentes secciones.

## VI.2.1 Comportamiento de los Parámetros de $M'_L(x, y, \vec{P})$

En esta sección se discutirá cual es el comportamiento de la ubicación exacta de esquina $(x_e, y_e)$, con respecto a cada uno de los $\vec{P}$ parámetros de nuestro modelo $M'_L(x, y, \vec{P})$. Por otro lado, las gráficas que se muestran fueron obtenidas con el programa **Sensibilidad**, ver sección V.4.3.

Los parámetros del modelo son: 1) el punto central de las rectas de borde $r_1$ y $r_2$, dado por $\mu_1$ y $\mu_2$, 2) el ángulo de apertura de la esquina, dado por $\vartheta_1$ y $\vartheta_2$. 3) El factor de difuminado en la dirección de los dos ejes principales del plano del sensor, dados por $\sigma_1$ y a $\sigma_2$, y 4) la amplitud y el piso de la esquina, dada por $A$ y $B$ respectivamente. El primer conjunto de parámetros, $\mu_1$ y $\mu_2$, no tiene efectos en la ubicación de $(x_e, y_e)$, debido a que depende completamente de la posición de la estructura, la esquina-L, en la región que se desee analizar. Esto es, en la región cuadrada de la imagen, que es definida por el usuario, debe existir una esquina-L, en caso contrario es imposible determinar el punto exacto de la estructura. El segundo conjunto de parámetros, $\vartheta_1$ y $\vartheta_2$, se discutió ampliamente en la sección III.5.5. En dicha sección, se concluyó que $(x_e, y_e)$ tiene un comportamiento no líneal con respecto al ángulo de apertura de la esquina, ver Figuras 36 y 37. Finalmente, el comportamiento de nuestro modelo quedará completo, analizando el desplazamiento del punto de esquina con respecto a los factores de difuminado y la amplitud de la esquina-L. Los cuales presentaremos a continuación.

### VI.2.1.1 Difuminado

Karl Rohr, reporta que el factor de difuminado tiene una relación líneal con respecto a la ubicación exacta del punto de esquina, ver sección III.5.5. Para observar cual es el comportamiento de $M'_L(x, y, \vec{P})$ se generó la estructura representada por la Figura 68.d. En dicha estructura se tomaron muestras distintas de valores de difuminado. Las condiciones iniciales del experimento son:

- Esquina-L con un ángulo de apertura de $90^o$, formada por $\vartheta_1 = \vartheta_2 = 0$.
- Un factor de difuminado inicial $\sigma_1 = \sigma_2 = 1.0$.
- La esquina-L está centrada en el origen del sistema coordenado del modelo, dado por $\mu_1 = \mu_2 = 0$.
- El piso de la esquina es $B = 50$.
- La amplitud de la estructura es $A = 100$.
- El tamaño de la región es de $51 \times 51$ pixeles.
- Los factores de difuminadado son muestreados en el rango $0.1 \leq \sigma_1, \sigma_2 \leq 25$ en incrementos de 0.1.

En la Figura 68, observamos lo siguiente.

1. Cuando $\sigma_1 = \sigma_2$ el desplazamiento de $(x_e, y_e)$ se comporta en forma líneal, ver Figura 68.c. Esta misma observación fue descrita por Rorh. Es importante enfatizar, que Rohr maneja un valor de difuminado $\sigma$, lo cual es equivalente a igular $\sigma_1 = \sigma_2$ en nuestro modelo de esquina-L. Por lo tanto, nuestro modelo es consistente con el desarrollo propuesto por Rohr.

2. Cuando $\sigma_1$ varía en el rango descrito en las condiciones iniciales y $\sigma_2 = 1$ permanece constante (curvas marcadas con $\sigma_1$ en la Figura 68.a y 68.b) se observa que el desplazamiento de $(x_e, y_e)$ *no es líneal.* Este mismo fenómeno ocurre cuando $\sigma_2$ varía y $\sigma_1$ permanece constante, ver curvas marcadas con $\sigma_2$ en la Figura 68.a y 68.b. Recuerde, que los CCD's rectangulares son los causantes de que $\sigma_1 \neq \sigma_2$. Es por ello, que en nuestro detector de esquina-L incluimos dos diferentes factores de difuminado para modelar la distorsión causada por los elementos fontosensibles con geometría rectangular. Por lo tanto, $M'_L(x, y, \vec{P})$ es un modelo superior, a los descritos anteriormente en la literatura, en términos de los desplazammientos de la esquina causados por $\sigma_1$ y $\sigma_2$.

3. Cuando una imagen no esta distorsionada, $\sigma_1 \to 0$ y $\sigma_2 \to 0$, el punto de esquina $(x_e, y_e) \to (x_0, y_0)$, vea los gáficos 68.a, 68.b y 68.c. $(x_0, y_0)$ es el punto de intersección de las rectas de borde $r_1$ y $r_2$. Este fenómeno puede ocurrir en imágenes sintéticas. Por lo tanto, nuestro modelo también es consistente con esta obervación.

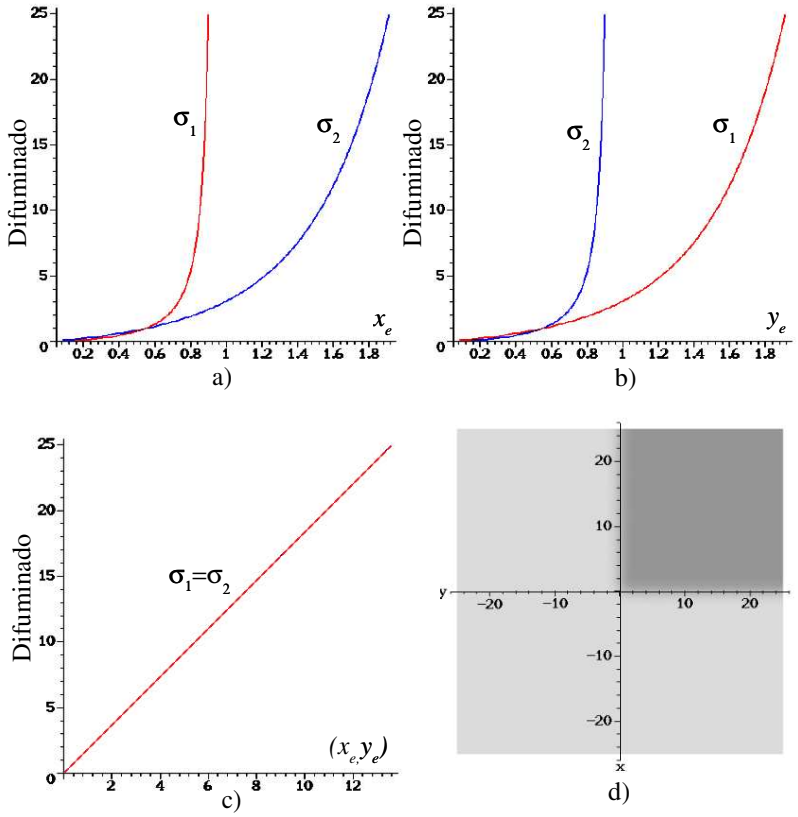



Figura 68: Comportamineto de $(x_e, y_e)$ con respecto a los factores de difuminado. a) Desplazamiento de $x_e$ cuando $\sigma_1$ y $\sigma_2$ varían. b) Desplazamiento de $y_e$ cuando $\sigma_1$ y $\sigma_2$ varían. c) Desplazamiento de $(x_e, y_e)$ cuando se varían simultáneamente $\sigma_1 = \sigma_2$. d) Estructura inicial de prueba.

#### VI.2.1.2 Amplitud

Para observar el comportamiento de $(x_e, y_e)$ con respecto a la amplitud de una esquina-L, se generaron las estructuras mostradas en la Figura 69 con las condiciones iniciales siguientes:

- Esquina-L con una ángulo de apertura de $90^o$, formada por $\vartheta_1 = \vartheta_2 = 0$.
- Un factor de difuminado inicial $\sigma_1 = \sigma_2 = 1.0$.
- La esquina-L está centrada en el origen del sistema coordenado del modelo, dado

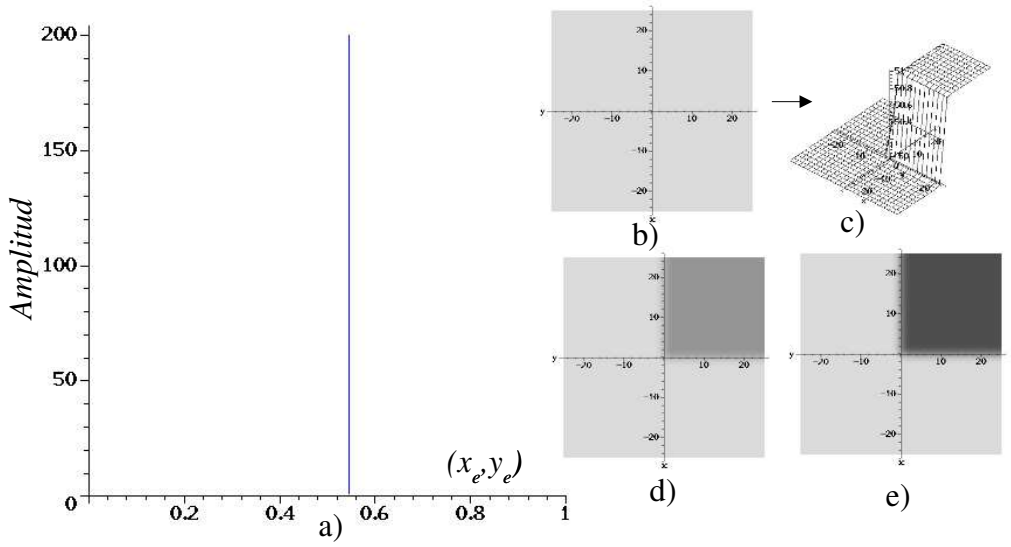



Figura 69: Comportamineto de $(x_e, y_e)$ con respecto a la amplitud de la esquina-L. a) Desplazamiento de $(x_e, y_e)$ cuando $A$ se varía. b) Estructura para $A = 1$. c) Modelo tridimnesional para $A = 1$. d) Estructura para $A = 100$. d) Estructura para $A = 200$.

por $\mu_1 = \mu_2 = 0$.

- El piso de la esquina es $B = 50$.
- La amplitud de la estructura es $A = 100$.
- El tamaño de la región es de $51 \times 51$ pixeles.
- Se tomaron muestras de la amplitud de la esquina en el rango de $1 \leq A \leq 200$ con incrementos de 1.0.

En la Figura 69.a observamos que la ubicación de $(x_e, y_e)$ no es afectada por la amplitud de la esquina, vea las estructuras mostradas en los gráficos b), d) y e) de la Figura 69. Esta misma observación fue presentada por Rohr. Por lo tanto, nuestro modelo de esquina $M'_L(x, y, \vec{P})$ es consistente. Finalmente, nuestro detector es capaz de detectar esquinas cuando el diferencial de valores de intensidad entre el piso y el techo de la esquina es de 1 tono de gris, ver Figuras 69.a y 69.b.

### VI.2.2 Prueba del Modelo con Imágenes Sintéticas

El detector $M'_L(x, y, \vec{P})$ se aplicó a un conjunto de esquinas con diferente ángulo de apertura generadas en forma sintética, ver Figura 70. El objetivo de esta prueba, es mostrar la confiabilidad de los métodos de optimización multidimensional pertenecientes a SIDEE. Las condiciones iniciales son las siguientes:

- Tamaño de la ventana $31 \times 31$ pixeles.
- El origen del sistema coordenado del modelo $(x, y)$ de las estructuras (a), (b), (c), (d), (e) y (f) están ubicadas en el pixel $(u_0, v_0)$ del sistema coordenado de la imagen que se lista a continuación. Cada una de estas estructuras representa distintas propiedades geométricas de una esquina-L, las cuales son:

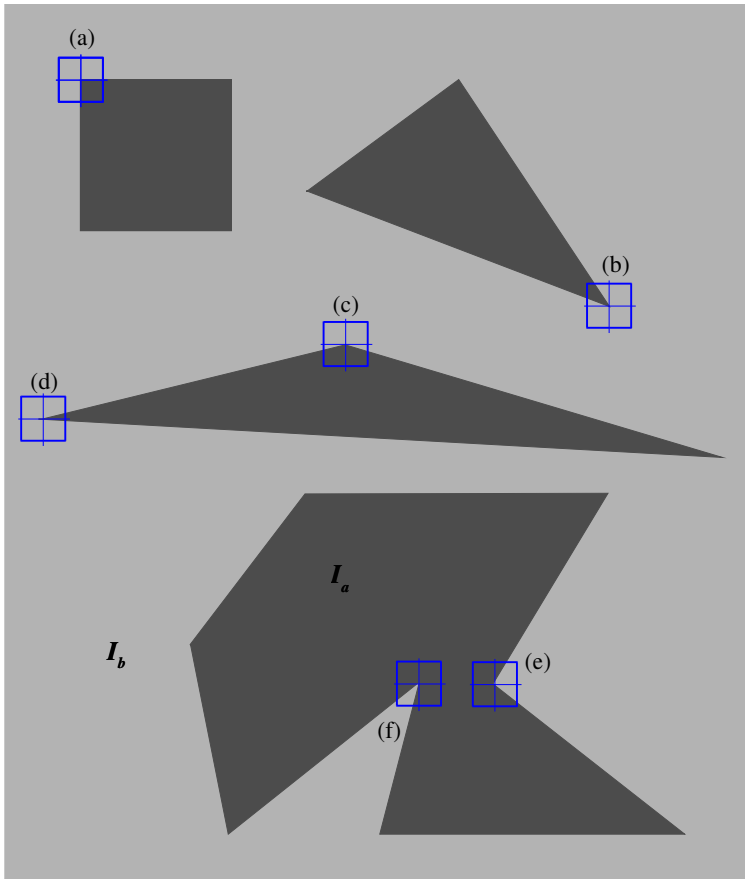



Figura 70: Imagen sintética de prueba.

(a) $(u_0, v_0) = (89, 89)$. Esquina con un ángulo recto, ubicada en el cuadrante $(x^+, y^-)$.

(b) $(u_0, v_0) = (706, 353)$. Esquina con un ángulo agudo, ubicada en el cuadrante $(x^-, y^+)$.

(c) $(u_0, v_0) = (398, 397)$. Esquina con un ángulo obtuso, ubicada en el cuadrante $(x^-, y^-)$.

(d) $(u_0, v_0) = (44, 484)$. Esquina con un ángulo agudo, ubicada en el cuadrante $(x^+, y^+)$.

(e) $(u_0, v_0) = (570, 795)$. Esquina con un ángulo obtuso, ubicada en el cuadrante $(x^+, y^+)$, donde el fondo de la estructura es diferente a (a), (b), (c) y (d).

(f) $(u_0, v_0) = (482, 795)$. Esquina con un ángulo agudo, ubicada en el cuadrante $(x^-, y^-)$, donde el fondo de la estructura es diferente a (a), (b), (c) y (d).

- Los poliédros mostrados en la figura de prueba, tienen un valor de intensidad $I_a = 76$ y el fondo de la imagen tiene un valor de intensidad $I_b = 179$.
- Los valores iniciales de los parámetros se muestran en la Tabla VI.1.

Tabla VI.1: Cotas de inicio de $\vec{P}$, utilizados en la obtención de las coordenadas de las esquina (a) $\rightarrow$ (f) de la Figura 70, para los 3 métodos de optimización.

| Estructura | Cotas | $\sigma_1$ | $\mu_1$ | $\vartheta_1$ | $\sigma_2$ | $\mu_2$ | $\vartheta_2$ | $A$ | $B$ |
|---|---|---|---|---|---|---|---|---|---|
| (a) | Máximo | 7.0 | 8.0 | 10.0 | 7.0 | 8.0 | 10.0 | 255 | 255 |
| | Mínimo | 1.0e-20 | -8.0 | -10.0 | 1.0e-20 | -8.0 | -10.0 | 0.0 | 0.0 |
| (b) $\rightarrow$ (f) | Máximo | 7.0 | 8.0 | 90.0 | 7.0 | 8.0 | 45.0 | 255 | 255 |
| | Mínimo | 1.0e-20 | -8.0 | -90.0 | 1.0e-20 | -8.0 | -90.0 | 0.0 | 0.0 |
| Parámetros Adicioneles | | | | | | | | | |
| Recocido Simulado | Temperatura inicial = 1 | | | | | | | | |
| | Pasos en el estado de equilibrio = 20 | | | | | | | | |
| Evolutivo | Número de individuos = 10 | | | | | | | | |
| | Porcentaje de apareamiento=70% | | | | | | | | |
| | Porcentaje de mutación=5% | | | | | | | | |
| | Porcentaje de convergencia = 80% | | | | | | | | |

La imagen sintética de prueba, Figura 70, fue generada con el editor de gráficas proporcionado por StartOffice. A partir de StartOffice, la imagen se transformó en el formato PGM (Portable Gray Map). El formato PGM es soportado por las librerías del paquete IUE. Finalmente, se ejecutaron los programas **Modelo_L_DownHill_ruido**, **Modelo_L_Recocido_iso_ruido** y **Modelo_L_Evolutivo**, ver secciones V.4.4, V.4.8, V.4.9 respectivamente, para generar las Tabla VI.2. Esta tabla muestra la ubicación del punto de esquina $(u_e, v_e)$ y los parámetros de nuestro modelo, para cada una de las estructuras mostradas en la Figura 70.

Tabla VI.2: Ubicación del punto exacto de esquina-L $(u_e, v_e)$ y valores de los $\vec{P}$ parámetros una vez aplicados los métodos de optimización descritos en SIDEE, para las estructuras (a) $\rightarrow$ (f) de la Figura 70.

*Down Hill simplex*

| Esquina | Pixel de inicio | | Punto de esquina | | Apertura | $s_{U_1}$ | $s_{U_2}$ | $\chi^2$ |
|---|---|---|---|---|---|---|---|---|
| | $u_0$ | $v_0$ | $u_e$ | $v_e$ | $\alpha^o$ | | | |
| (a) | 89 | 89 | 87.404142 | 87.709506 | 89.76333 | +1 | -1 | 1.13215e-15 |
| (b) | 706 | 353 | 707.028574 | 353.831573 | 35.7994 | -1 | +1 | 5.44007e-16 |
| (c) | 398 | 397 | 396.825889 | 397.466375 | 149.9239 | -1 | -1 | 1.31891e-09 |
| (d) | 44 | 484 | 40.985292 | 484.451324 | 19.1055 | +1 | +1 | 4.22482e+04 |
| (e) | 570 | 795 | 570.573303 | 792.934090 | 97.4398 | +1 | +1 | 1.22339e-11 |
| (f) | 482 | 795 | 483.081960 | 792.272648 | 37.2723 | -1 | -1 | 6.41950e-23 |

| | $\sigma_1$ | $\sigma_2$ | $\mu_1$ | $\mu_2$ | $\vartheta_1^o$ | $\vartheta_2^o$ | $A$ | $B$ |
|---|---|---|---|---|---|---|---|---|
| (a) | 6.04357e-02 | 5.20620e-02 | -1.60603e+00 | 1.33941e+00 | -8.74457e-01 | 6.37787e-01 | 103.00 | 76.00 |
| (b) | 2.34760e-02 | 9.25585e-03 | 4.76497e-01 | -4.44535e-01 | -3.35800e+01 | -2.06206e+01 | 103.00 | 76.00 |
| (c) | 2.34221e-03 | 1.72416e-02 | -2.75132e+00 | -1.82255e-01 | -7.35272e+01 | 1.36033e+01 | 103.00 | 76.00 |
| (d) | 6.29735e-02 | 8.53333e-03 | -1.25144e+00 | -7.01737e-01 | 7.56428e+01 | -4.74830e+00 | 102.54 | 76.46 |
| (e) | 1.15833e-02 | 8.79562e-04 | -6.67335e-01 | 2.51901e+00 | 3.09704e+01 | -3.84093e+01 | 103.00 | 76.00 |
| (f) | 1.33997e-02 | 4.80644e-03 | 3.97708e-01 | 1.86938e+00 | 1.41271e+01 | 3.86009e+01 | 103.00 | 76.00 |

*continúa en la siguiente página*

Tabla VI.2: Ubicación del punto exacto *(continuación)*

*Recocido Simulado*

| Esquina | Pixel de inicio | | Punto de esquina | | Apertura | $s_{U_1}$ | $s_{U_2}$ | $\chi^2$ |
|---|---|---|---|---|---|---|---|---|
| | $u_0$ | $v_0$ | $u_e$ | $v_e$ | $\alpha^o$ | | | |
| (a) | 89 | 89 | 87.526970 | 87.437709 | 90.01414 | +1 | -1 | 1.94364e-10 |
| (b) | 706 | 353 | 707.017039 | 353.829675 | 35.7614 | -1 | +1 | 3.56136e-13 |
| (c) | 398 | 397 | 396.797404 | 397.427744 | 149.4558 | -1 | -1 | 9.75333e-13 |
| (d) | 44 | 484 | 40.077467 | 484.514532 | 16.977 | +1 | +1 | 4.22483e+04 |
| (e) | 570 | 795 | 570.647257 | 793.111032 | 97.7988 | +1 | +1 | 6.15267e-01 |
| (f) | 482 | 795 | 483.046760 | 792.255049 | 37.3776 | -1 | -1 | 1.54081e-13 |

| | $\sigma_1$ | $\sigma_2$ | $\mu_1$ | $\mu_2$ | $\vartheta_1^o$ | $\vartheta_2^o$ | $A$ | $B$ |
|---|---|---|---|---|---|---|---|---|
| (a) | 8.69092e-02 | 7.67896e-02 | -1.47251e+00 | 1.56314e+00 | -1.88830e-02 | 3.30323e-02 | 103.00 | 76.00 |
| (b) | 5.01805e-03 | 7.87605e-03 | 4.71431e-01 | -4.47754e-01 | -3.35296e+01 | -2.07090e+01 | 103.00 | 76.00 |
| (c) | 2.11139e-02 | 2.12681e-02 | -2.62621e+00 | -1.21410e-01 | -7.34947e+01 | 1.40389e+01 | 103.00 | 76.00 |
| (d) | 3.16852e-07 | 3.69060e-04 | -1.76589e+00 | -7.58444e-01 | 7.65812e+01 | -3.55820e+00 | 102.56 | 76.46 |
| (e) | 9.50347e-10 | 7.77463e-10 | -4.33799e-01 | 2.38709e+00 | 2.97825e+01 | -3.75813e+01 | 102.98 | 76.03 |
| (f) | 1.37867e-02 | 3.61182e-03 | 3.66855e-01 | 1.91193e+00 | 1.39431e+01 | 3.86793e+01 | 103.00 | 76.00 |

*Evolutivo*

| Esquina | Pixel de inicio | | Punto de esquina | | Apertura | $s_{U_1}$ | $s_{U_2}$ | $\chi^2$ |
|---|---|---|---|---|---|---|---|---|
| | $u_0$ | $v_0$ | $u_e$ | $v_e$ | $\alpha^o$ | | | |
| (a) | 89 | 89 | 87.895717 | 87.644934 | 89.3094 | +1 | -1 | 9.40007e-06 |
| (b) | 706 | 353 | 706.807636 | 353.755519 | 36.2416 | -1 | +1 | 1.24802e-06 |
| (c) | 398 | 397 | 396.943119 | 397.390921 | 148.7181 | -1 | -1 | 5.69696e-05 |
| (d) | 44 | 484 | 41.009408 | 484.473710 | 18.53073 | +1 | +1 | 4.22483e+04 |
| (e) | 570 | 795 | 570.411461 | 792.991919 | 95.7904 | +1 | +1 | 4.53057e-04 |
| (f) | 482 | 795 | 483.170778 | 792.099214 | 37.0405 | -1 | -1 | 1.17246e-06 |

| | $\sigma_1$ | $\sigma_2$ | $\mu_1$ | $\mu_2$ | $\vartheta_1^o$ | $\vartheta_2^o$ | $A$ | $B$ |
|---|---|---|---|---|---|---|---|---|
| (a) | 1.24043e-02 | 1.41591e-02 | -1.11322e+00 | 1.34753e+00 | 6.73623e-02 | -7.57916e-01 | 103.00 | 76.00 |
| (b) | 4.32836e-03 | 9.65062e-03 | 3.20867e-01 | -4.52084e-01 | -3.30288e+01 | -2.07296e+01 | 103.00 | 76.00 |
| (c) | 9.80709e-03 | 8.63958e-03 | -2.33377e+00 | -1.17649e-01 | -7.30867e+01 | 1.43686e+01 | 103.00 | 76.00 |
| (d) | 3.31949e-02 | 2.23349e-03 | -1.10955e+00 | -7.08228e-01 | 7.58993e+01 | -4.43003e+00 | 102.59 | 76.45 |
| (e) | 1.22747e-03 | 1.74839e-03 | -8.23944e-01 | 2.32158e+00 | 3.15841e+01 | -3.73745e+01 | 103.00 | 76.00 |
| (f) | 1.13199e-02 | 5.92920e-03 | 4.34052e-01 | 1.97009e+00 | 1.43014e+01 | 3.86581e+01 | 103.00 | 76.00 |

Las Tablas VI.2, muestra la confiabilidad de los algoritmos de optimización multidimensional codificados en las librerías de SIDEE. Estas tablas, listan los valores de los $\vec{P}$ parámetros de $M'_L(x, y, \vec{P})$ para las estructuras (a) $\rightarrow$ (f) después del ajuste. Así mismo, se reporta el ángulo de apertura de la esquina-L $\alpha$, calculado mediante la Ecuación

(III.19) y el valor de $\chi^2$ para las estructuras en estudio. En dichas tablas, se sometió a nuestras librerías a estructuras con diferente geometría. Esto es; esquinas ubicadas en diferentes cuadrantes del sistema coordenado del modelo, vea los recuadros mostrados en los gráficos (a) $\rightarrow$ (c) de la Figura 70; esquinas agudas, gráficos (b), (d) y (f); esquina obtusa, gráficos (c) y (e); esquina recta, gráfico (a); y esquinas con diferente estructura en piso y en la cresta de la esquina, compare por ejemplo el gráfico (f) donde el piso de la esquina correspone a la cresta de la esquina del gráfico (b). Para todos estos casos los métodos de optimización y nuestro modelo analítico $M'_L(x, y, \vec{P})$ detectaron exitosamente el punto exacto de esquina $(u_e, v_e)$. Notese, como el valor de $\chi^2 \rightarrow 0$ para las estructuras (a), (b), (c), (e) y (f), lo cual significa que la suma de error cuadrático entre nuestro modelo y los valores de intensidad de la imagen digital tienden a cero. Por lo tanto, nuestros algoritmos son consistentes en relación a su codificación expresada por SIDEE.

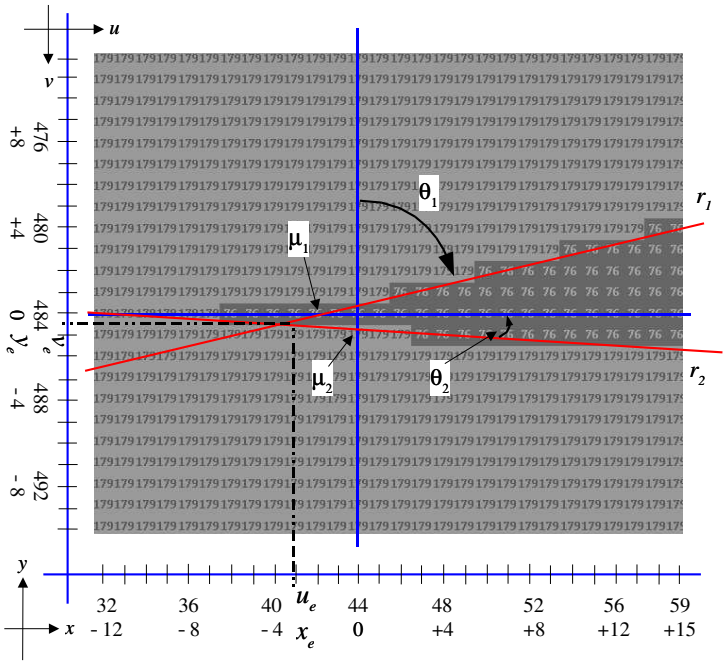



Figura 71: Ubicación de los $\vec{P}$ parámetros de la estructura 70.d.

Finalmente, en la Figura 71 se presenta la ubicación de los $\vec{P}$ parámetros de $M'_L(x, y, \vec{P})$ para la estructura (d) de la imagen de prueba. Dicha figura muestra los valores de intensidad de la estructura en estudio, el sistema coordenado del modelo $(x, y)$ y el sistema

coordenado de la imagen $(u, v)$. La descripción del modelo $M'_L(x, y, \vec{P})$ empleando los valores obtenidos a través de la estrategia Nelder y Mead es la siguiente:

1. Los signos de cada una de la *Funciones Unitarias de Borde* de nuestro modelo, se calculan a partir de la rutina **Cual_cuadrante**, ver Tabla V.1.15. Los valores reportados son: $s_{U_1} = +1$ y $s_{U_2} = +1$. Obsérvese, como la esquina está orientada en el cuadrante $(x^+, y^+)$ en la Figura 71.

2. El piso de la esquina es $B = 76.46$ y la amplitud es $A = 102.54$. Obsérvese que el valor original del piso de la esquina es $I_b = 179$ y la cresta $I_a = 76$. La amplitud teórica de la estructura es $|I_a - I_b| = 103$. Dicho valor se aproxima al reportado por los métodos de optimización después del ajuste. Por otro lado, nuestro modelo supone que el piso de la esquina siempre tiene menor energía que la cresta de la esquina. Esto no ocurre en la imagen original, donde el piso de la esquina tiene más energía que la cresta de la misma $(I_b > I_a)$, por lo tanto es necesario invertir estos valores de intensidad. La rutina **Si_background_blanco**, ver Tabla V.1.17., se encarga de dicha inversión de valores de intensidad a partir de $I_{invertida}(u, v) = 255 - I_{original}(u, v)$. Aplicando esta última ecuación el piso de la esquina es teoricamente $I_b = 76$. Dicho valor se aproxima también al reportado por los métodos de optimización después del ajuste. Obsérvese que las otras estructuras (a), (b), (c) y (f) los valores de $A$ y $B$ corresponden exactamente al valor teórico.

3. Aplicando las ecuaciones (III.2) y (III.5), las rectas de borde $r_1$ y $r_2$, son, ver Figura 71:

$$\begin{aligned} r_1 &= x - y\tan(75.6428^o) - (-1.25144) \\ r_2 &= y - x\tan(-4.7483^o) - (-0.701737) \end{aligned}$$

4. Aplicando las ecuaciones (III.6), (III.9) y (III.10), el modelo resultante es:

$$\begin{aligned} M'_L(x, y, \vec{P}) &= \left(+\tfrac{1}{2}\phi\left(\tfrac{\sqrt{2}(x - y\tan(75.6428^o)+1.25144)}{2(6.29735e-02)}\right) + \tfrac{1}{2}\right) \\ &\quad \left(+\tfrac{1}{2}\phi\left(\tfrac{\sqrt{2}(y - x\tan(-4.7483^o)+0.701737))}{2(8.5333e-03)}\right) + \tfrac{1}{2}\right)(102.54) + 76.46 \end{aligned}$$

5. A partir de estos valores, se calcula el punto exacto de esquina $(u_e, v_e)$ a través de nuestro criterio de ubicación exacta de esquina, dado por la rutina **Modelo_L_Curvatura**.

6. Finalmente, obeservese que el valor de $\chi^2 = 4.2248e + 04$ para la estructura (d), no tiende a cero como en las otras estructuras, ver Tabla VI.2. Esto se debe a la composición de la esquina (d). Notese, como existe una "línea" de pixeles antes de observarse como se abre la esquina, ubicada entre los pixles (38,484) y (45,484). Dicha "línea" provoca que $M'_L(x, y, \vec{P})$ no pueda modelar completamente a todos estos pixeles y por tanto la suma cuadrática de los errores no tienda a cero. En particular, el valor de $\chi^2$ en imágenes reales nunca tiende a cero, es por ello que nuestro criterio de "paro" en los algoritmos de optimización, está en relación con una fracción de la distancia entre el mejor y el peor vector de parámetros de dichos algoritmos, ver Ecuación (IV.39).

### VI.2.3 Prueba del Modelo con Imágenes Reales

El detector preciso de esquinas múltiples $M'_L(x, y, \vec{P})$ se aplicó a un conjunto de imágenes reales. El objetivo de esta sección es mostrar el funcionamiento de SIDEE y nuestro modelo de esquina-L, sobre una región de una imagen real. La Figura 72, esquematiza el proceso de detección del punto exacto de esquina $(u_e, y_e)$. Las condiciones iniciales de la prueba son:

1. El gráfico 72.a, es una malla de calibración adquirida con una cámara digital Pulnix 9701, con una resolución de 768x484 pixeles y un tamaño del sensor de $11.6\mu m \times 13.6\mu m$.

2. El tamaño de la ventana es de $17 \times 17$ pixeles alrededor del pixel $(u_0 = 411, v_0 = 198)$, ver gráfico 72.b.

3. Las cotas para generar el simplex o población inicial corresponden a la estructura (a) de la Tabla VI.1.

4. Los parámetros de control de los algoritmos son:

   - *Down Hill Simplex.* Ninguno.
   - *Recocido Simulado.* Temperatura de inicio $T = 1$. Longitud del estado de equilibrio $I = 20$.
   - *Evolutivo.* Porcentaje de apareamiento $pa = 0.80$. Porcentaje de mutación $pm = 0.05$. Porcentaje de convergencia $pc = 0.8$.

- Para todos los casos el número máximo de iteraciones o generaciones es de 4500.

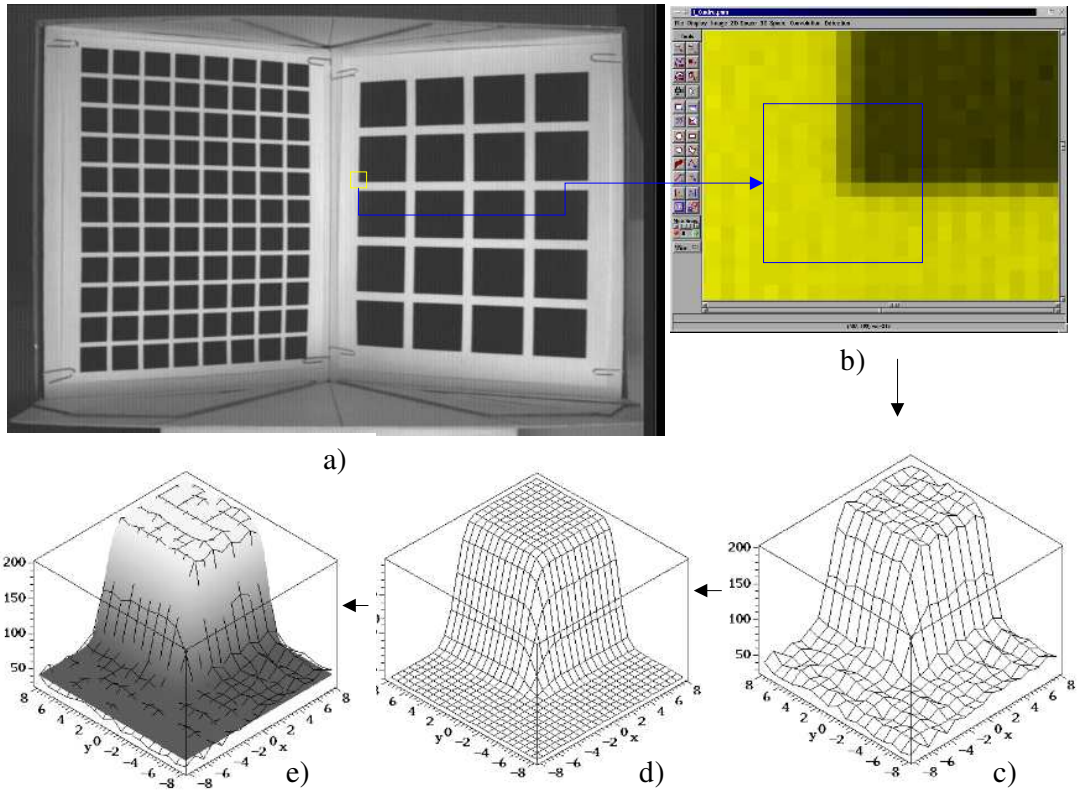



Figura 72: Detección del punto exacto de esquina $(u_e, y_e)$ de una imagen real. a) malla de calibración. b) amplificación de la imagen en la vecindad del pixel $(411, 198)$. c) malla tridimensional de la superficie real. d) malla tridimensional del modelo $M'_L(x, y, \vec{P})$, después del ajuste. e) superposición de c) y d).

A partir de estas condiciones experimentales, se genera la Tabla VI.3, en la cual se muestran los valores de los $\vec{P}$ parámetros de nuestro detector de esquina para las tres estrategias de optimización. El gráfico 72.d, es una malla tridimensional de $M'_L(x, y, \vec{P})$, para los valores de $\vec{P}$ reportados en la Tabla VI.3 por el método *Evolutivo*. Obsérvese que el valor de $\chi^2$ es prácticamente el mismo para las estrategias en estudio. El gráfico 72.e, muestra la calidad del ajuste de nuestros algoritmos de optimización. Adicionalmente,

muestra también la consistencia del código implementado en SIDEE aplicado a imágenes reales.

Tabla VI.3: Ubicación del punto exacto de esquina-L $(u_e, v_e)$ y los valores de los $\vec{P}$ parámetros una vez aplicados los métodos de optimización, para la estructura de la Figura 72.

| Método | Pixel de inicio | | Punto de esquina | | $s_{U_1}$ | $s_{U_2}$ | $\chi^2$ |
|---|---|---|---|---|---|---|---|
| | $u_0$ | $v_0$ | $u_e$ | $v_e$ | | | |
| Down Hill (D) | 411 | 198 | 410.433980 | 198.596549 | +1 | +1 | +8.48973e+03 |
| Recocido Simulado (R) | 411 | 198 | 410.434020 | 198.596534 | +1 | +1 | +8.48973e+03 |
| Evolutivo (E) | 411 | 198 | 410.438597 | 198.598558 | +1 | +1 | +8.48974e+03 |

| | $\sigma_1$ | $\sigma_2$ | $\mu_1$ | $\mu_2$ | $\vartheta_1^o$ | $\vartheta_2^o$ | $A$ | $B$ |
|---|---|---|---|---|---|---|---|---|
| (D) | 8.75785e-01 | 7.04228e-01 | -9.68571e-01 | -1.03475e+00 | 1.00912e+00 | 1.43804e-01 | 153.61 | 41.48 |
| (R) | 8.76027e-01 | 7.04226e-01 | -9.68557e-01 | -1.03481e+00 | 1.00847e+00 | 1.43922e-01 | 153.61 | 41.48 |
| (E) | 8.75922e-01 | 7.04278e-01 | -9.67822e-01 | -1.03464e+00 | 9.97871e-01 | 1.41724e-01 | 153.61 | 41.49 |

En la parte introductoria de este trabajo y en el desarrollo de nuestro modelo analítico de esquinas múltiples, capítulos I y III respectivamente, planteamos que $M'_L(x, y, \vec{P})$ tiene la capacidad de modelar factores de difuminado $\sigma_1$ y $\sigma_2$ en la dirección de los dos ejes principales del plano del sensor. Si la geometría del sensor es rectanguar, afirmamos que $\sigma_1 \neq \sigma_2$. Además, de la geometría de $M'_L(x, y, \vec{P})$, observamos que el factor de difuminado produce una curvatura en el punto de unión de las rectas de borde $r_1$ y $r_2$. Esta curvatura corresponde al ecuación implícita $M_L(x, y, \vec{P}) = 0.5$. Si $\sigma_1 \neq \sigma_2$, entonces $M_L(x, y, \vec{P}) = 0.5$ es una curva no simétrica a la recta $r_{bis}$ que bisecta al ángulo de apertura de la esquina-L $\alpha$. Por lo tanto, concluimos que el punto exacto de esquina $(u_e, v_e)$ debiera ser el punto que represente la mínima distancia euclidiana entre $M_L(x, y, \vec{P}) = 0.5$ y el punto de itersección de $r_1$ y $r_2$. En la Figura 73, cada uno de estos conceptos son graficadas, de la siguiente forma:

1. El sistema coordenado del modelo se encuentra centrado en el pixel (411,198). A partir de ese punto el sistema coordenado de la esquina se ubica en rango $[-w, +w]$, donde $w$ es la mitad del tamaño de la ventana definida por el usuario.

2. El punto exacto de esquina se ubica en la coordenada $(x_e, y_e)$, en el sistema coordenado del modelo y en la coordenada $(u_e, v_e)$, en el sistema coordenado de la imagen.

Por tanto $(x_e, y_e)$ y $(u_e, v_e)$ son equivalentes.

3. Obsérvese que el punto de intersección $(x_0, y_0)$, de las rectas de borde $r_1$ y $r_2$ no corresponde al punto exacto por donde pasa la esquina. Los factores de difuminado $\sigma_1$ y $\sigma_2$ provocan un dezplazamiento del punto exacto de esquina, ver sección III.5. Para el caso de la Figura 73, $(x_0 = -0.9868, y_0 = -1.0372)$.

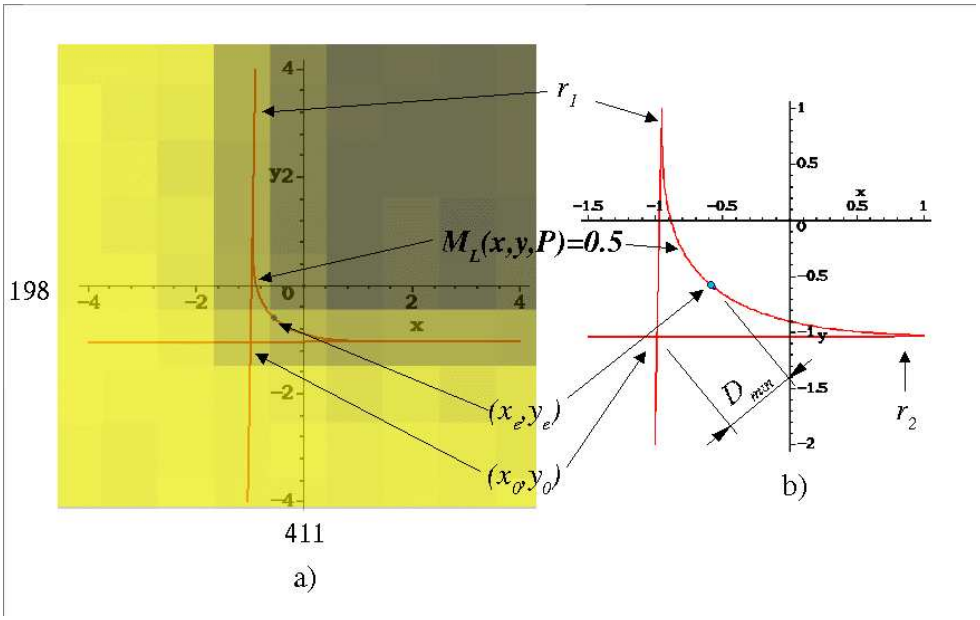



Figura 73: Geometría de $M'_L(x, y, \vec{P})$, para una esquina-L de una imagen real. a) superposición de la estructura real en la vecindad del pixel (411,198) y la geometría de nuestro detector. d) amplificación de la zona donde se ubica el punto exacto de esquina $(u_e, v_e)$.

4. Dado que $\sigma_1 \neq \sigma_2$, ver Tabla VI.5, la curva de contorno central del modelo no es simétrica. El la figura 73.b, se observa que $M_L(x, y, \vec{P}) = 0.5$ se extiende más sobre el eje $x$ que sobre otro eje.

5. El punto exacto de esquina $(x_e = -0.5660, y_e = -0.5965)$ corresponde, a la mínima distancia $D_{min}$ entre $(x_0, y_0)$ y $M_L(x, y, \vec{P}) = 0.5$. Para mostrar este efecto, calculamos el punto de intersección $(x_{bis} = 0.5224, y_{bis} = 0.6093)$ de la recta bisectriz $r_{bis}$, Ecuación (III.21), y $M_L(x, y, \vec{P}) = 0.5$. Después, calculamos la distancia euclidiana $D_{bis} = 0.6314$ entre $(x_{bis}, y_{bis})$ y $(x_0, y_0)$. Seguido, obtenemos $D_{min} = 0.60933$. Es evidente que $D_{min} < D_{bis}$, lo que muestra la confiabilidad de nuestros algoritmos.

6. Finalmente, observamos que la relación existe entre la geometría del sensor $\frac{11.6\mu m}{13.6\mu m} = 0.85$ es aproximadamente igual a la relación existente entre $\sigma_1$ y $\sigma_2$, $\frac{0.704228}{0.875785} = 0.80$.

En conclusión, se ha mostrado con imágenes reales la robustez de nuestro análisis en el desarrollo del detector preciso de esquinas múltiples y su implementación a través de las librerías codificadas en SIDEE.

### VI.2.4 Importancia en la Inicialización de los Parámetros

Las estrategias *Down Hill Simplex*, *Recocido Simulado* y *Evolutiva* tienen dos clases de parámetros de inicialización: los primeros se refieren a la forma de como generar el *simplex* o *población* inicial, los segundos se relacionan con los parámetros que cada uno de los métodos de optimización necesitan para su ejecución, por ejemplo, la temperatura inicial y número de pasos en el estado de equilibrio en *Recocido Simulado* y el porcentaje de apareamiento y mutación en la estrategia *Evolutiva*. El segundo tipo de parámetros de inicialización los llamamos *parámetros de control*. En esta sección se discutirá la importancia de la primera clase de parámetros, la generación del *simplex* o *población* inicial. Sin embargo, ambas clases de parámetros impactan en el desempeño de los algoritmos de optimización.

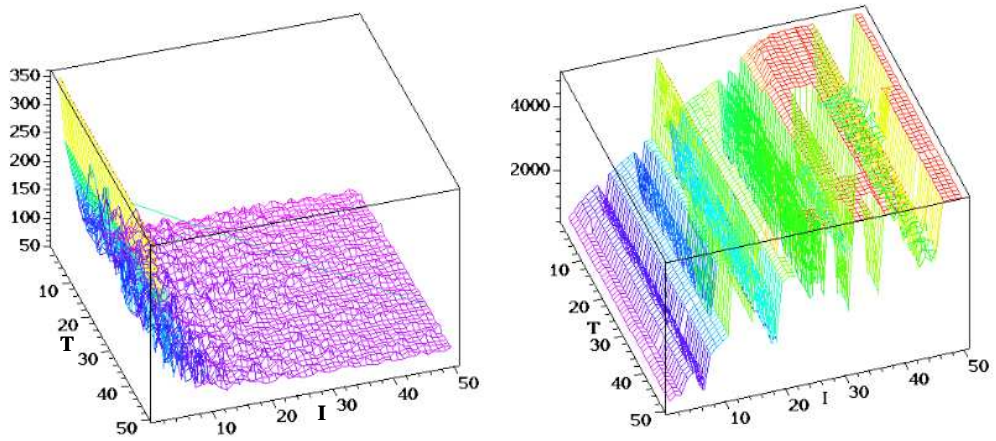
En el capítulo de *Optimización*, el método de *Recocido Simulado* se codificó atendiendo a dos criterios distintos para generar el simplex inicial, los cuales son:

1. La población inicial se genera en forma aleatoria, entre las cotas máximas y mínimas definidas por la *Función Unitaria de Borde* (FUB). Estas cotas son: los ángulos de las rectas de borde $-90^o < \vartheta_1, \vartheta_2 < 90^o$, los puntos centrales $-\frac{w}{2} < \mu_1, \mu_2 < +\frac{w}{2}$, los factores de difuminado $-\frac{w}{2} < \sigma_1, \sigma_2 < +\frac{w}{2}$ y la amplitud y piso de la esquina $0 \leq A, B \leq 255$. A partir de este criterio, se hace un análisis de los parámetros de control de la estrategia *Recocido Simulado* y se concluye que con algunos valores de estos parámetros el algoritmo es más eficiente, ver sección IV.4.2.1.

2. La población inicial es formada a través de un muestreo uniforme entre las cotas superior e inferior, proporcionadas por el usuario.

Primeramente, analizaremos cual es el comportamiento de la estrategia *Recocido Simulado*. Para ello, se genera la misma malla tridimensional que se presento en la Figura

46.b., utilizando una inicialización *uniforme* en lugar de una aleatoria. La Figura 74, muestra el comportamiento de los parámetros de control, temperatura inicial $T$ y número de pasos en el estado equilibrio $I$, en función del número total de pasos $N$ que el algoritmo necesita para detectar el punto de esquina-L. Para esta figura se utilizaron las mismas condiciones iniciales mostradas en la Figura 46. El gráfico a) es la malla tridimensional resultante de una inicialización uniforme, donde las cotas proporciadas por el usuario están dadas por los límtes de la FUB. El gráfico b) es la malla tridimensional resultante de una inicialización aleatoria.




a) b)

Figura 74: Comportamiento de los parámetros de control, temperatura inicial $T$ y número de pasos en el estado de equilibrio $I$, del algoritmo *Recocido Simulado*, de la estructura de la Figura 46.c. a) Malla tridimensional generada con un simplex *uniforme*. b) Malla tridimensinal generada con un simplex *aleatorio*.

En la Figura 74.a, se observa claramente que el número de pasos necesarios del algoritmo *Recocido Simulado* para la ubicación de $(u_e, v_e)$, se comporta en forma diametralmente opuesta al mostrado en la 74.b. Esto es: 1) para cualquier valor de $I$ y $T$ en una inicialización uniforme, el algoritmo siempre converge, observe como para $I = 50$, $N \approx 50$, en oposición, cuando la inicialización es aleatoria existen valores de $I$ en donde el algoritmo no converge, observe como para $I = 50$, $N \approx 5000$, 2) en una inicialización uniforme no se observan muchos extremos y zonas de linearidad como en una inicialización aleatoria, 3) en la inicializacón uniforme el peor rendimiento es $N \approx 350$, mien-

tras que en la inicilización aleatoria existen muchos casos para $N \approx 5000$, y 4) cuando $I > 20$ el agoritmo converge rápidamente en una inicialización uniforme, en oposición, cuando $I < 20$ se observa mejor rendimento en la inicialización aleatoria. Es por ello, que ***nosotros empleamos una inicialización que muestrea uniformemente el simplex o población inicial en los rangos definidos por el usuario***, en nuestros algoritmos de optimización.

En la sección IV.4.2.1, se trató de explicar el comportamiento errático de la Figura 74.b., en función de las heurísticas involucradas en el proceso de *Recocido Simulado* y no se consideró el efecto de la generación del simplex o población inicial. Intencionalmente no se modificó dicho análisis. Ahora, se muestra que el comportamiento errático del rendimiento del proceso *Recocido Simulado* en la Figura 74.b, se debe completamente a la forma de generar el simplex de inicio, esto es, una generación aleatoria en lugar de una generación uniforme.

Otra característica importante de nuestros algoritmos, es que las cotas del simplex o la población de inicio pueden ser controladas por el usuario. Si conocemos *a priori* cuales son los posibles límites que puede tomar la población inicial de la estructura en estudio, entonces podemos obtener un mejor rendimiento de los procesos de optimización, menor número de pasos, o bién, podemos obtener una mayor precisión en la detección de $(u_e, v_e)$.

En conclusión, un muestreo uniforme del simplex o población inicial sumado a un conocimiento previo de la estructura, redundan en un mejor desempeño del detector preciso de esquinas múltiples y los algoritmos codificados en SIDEE.

### VI.2.5 Eficiencia de los Algoritmos de Optimización

En esta sección se analiza el comportamiento de las estrategias *Down Hill Simplex*, *Recocido Simulado* y *Evolutiva*, en cuanto a la precisión alcanzada por cada uno de ellos en un determinado número de pasos. Para logar esto, se generó el siguiente experimento:

1. Se aplicaron los tres algoritmos de optimización a las estructuras (a), (b) y (c) de la imagen sintética de prueba, Figura 70.

2. Las cotas del simplex o población inicial de estas estructuras se muestran en la Tabla VI.1.

3. El tamaño de la ventana es de $17 \times 17$ pixeles, centradas en el pixel:

   (a) $(u_0, v_0) = (87, 87)$.

   (b) $(u_0, v_0) = (707, 353)$.

   (c) $(u_0, v_0) = (396, 397)$.

4. Los parámetros de control de los algoritmos son:

   - *Down Hill Simplex*. Ninguno.
   - *Recocido Simulado*. Temperatura de inicio $T = 1$. Longitud del estado de equilibrio $I = 20$.
   - *Evolutivo*. Porcentaje de apareamiento $pa = 0.80$. Porcentaje de mutación $pm = 0.05$. Porcentaje de convergencia $pc = 2$, a fin de que el algoritmo no se detenga por este concepto.

5. El simplex está compuesto por 9 vértices para los métodos *Down Hill Simplex* y *Recocido Simulado*. Este número es fijo y no puede ser modificado. En el caso de la estrategia *Evolutiva* el tamaño de población es 10. Por razones de diseño de la librería **Evolutivo_L**, el número de pobladores debe ser un número par. El valor de 10, en el número de individuos, fue seleccionado para poder comparar a los tres métodos de optimización en igualdad de circunstancias.

6. El simplex o población inicial es una muestra uniforme entre las cotas de $\vec{P}$ dadas en la Tabla VI.1.

7. El número de pasos permitidos para cada algoritmo se muestrea en el rango de $[25, 475]$, con incrementos de 25. Para el rango de $[500, 900]$ se toman incrementos de 100 en cada muestra. Finalmente, para el rango de $[1000, 4000]$ se tomaron incrementos de 500 pasos.

8. Se efectuaron 30 realizaciones por muestra.

9. La tolerancia del criterio de paro $ftol = 1 \times 10^{-80}$, Ecuación (IV.39), se utiliza para que los algoritmos no se detengan por este concepto.

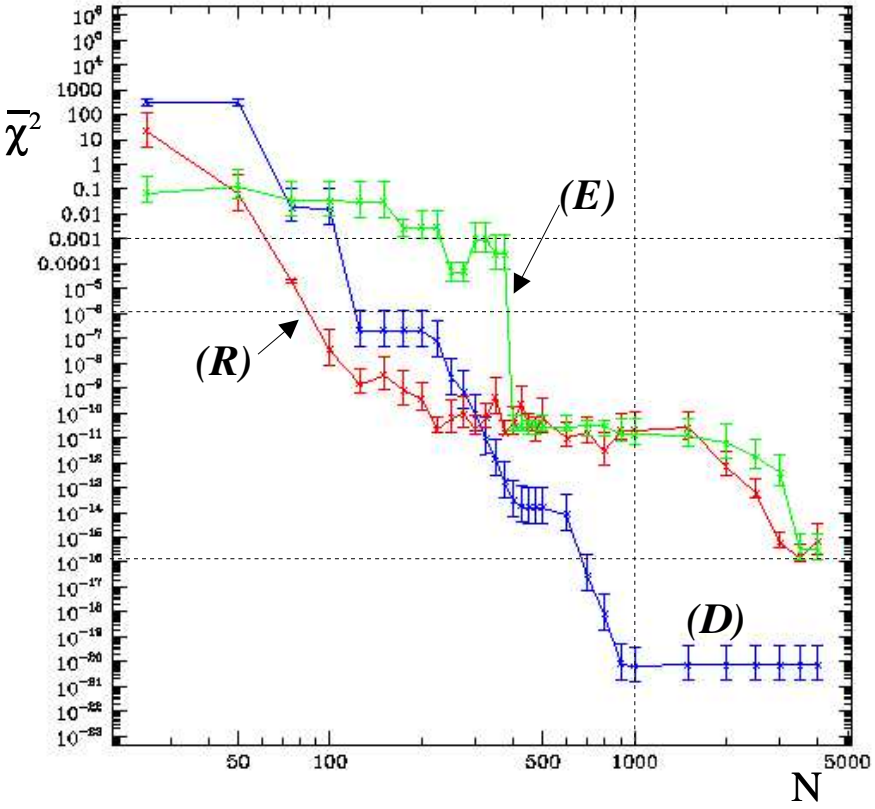



Figura 75: Eficiencia de las estrategias *Down Hill Simplex* ***(D)***, *Recocido Simulado* ***(R)*** y *Evolutiva* ***(E)*** para la estructura (**a**) de la imagen sintética de prueba, ver Figura 70.

A partir de estas consideraciones experimentales, se generaron las figuras 75, 76 y 77, de las estructuras (a), (b) y (c) de la imagen sintética de prueba. Cada una de estas figuras, representan el rendimiento o eficiencia de los tres métodos de optimización para una esquina recta, aguda u obtusa respectivamente, donde : 1) el eje horizontal representa el número de pasos o generaciones $N$ de cada uno de los algoritmos, graficados en forma logarítmica, 2) el eje vertical es el promedio de $\overline{\chi^2}$ resultante del mejor individuo o vértice del simplex en cada una de las 30 realizaciones para un valor fijo de $N$, graficada también en forma logarítmica. Adicionalmente, se grafica la desviación estándar $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\chi_i^2 - \overline{\chi^2})^2}$ para mostrar el "ancho" de la distribución de las 30 realizaciones para cada $N$. $s$ es representado por las barras $\frac{\top}{\perp}$ centradas en $\overline{\chi^2}$. Dado que la escala es

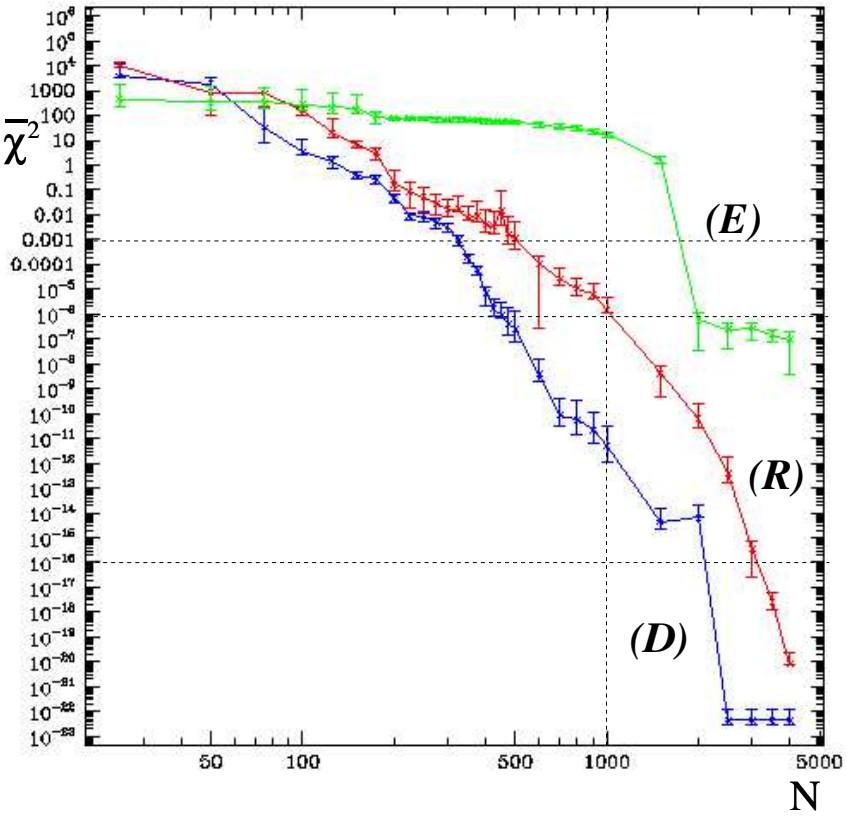



Figura 76: Eficiencia de las estrategias *Down Hill Simplex* ***(D)***, *Recocido Simulado* ***(R)*** y *Evolutiva* ***(E)*** para la estructura (**b**) de la imagen sintética de prueba, ver Figura 70.

logarítmica, el tamaño de cada barra se calcula mediante:

$$\begin{aligned} \top &= \log(\overline{\chi^2} + s) - \log(\overline{\chi^2}) \\ \perp &= \log(\overline{\chi^2}) - \log(|\overline{\chi^2} - s|) \end{aligned} \tag{VI.1}$$

El cálculo de $\perp$ provoca que cuando $s > \overline{\chi^2}$ las barras de $s$ no se observen de igual tamaño en estas gráficas.

El eje horizontal $N$ representa la *distancia* o *trabajo* de los métodos de optimización. Esto es, que tanto recorre un algoritmo para alcanzar cierto valor de $\chi^2$. El eje vertical $\chi^2$ es la *precisión* alcanzada por las tres estrategias de optimización. Finalmente, debe recordarse que el valor de $\chi^2$ es adimensional y su magnitud es unicamente una medida de

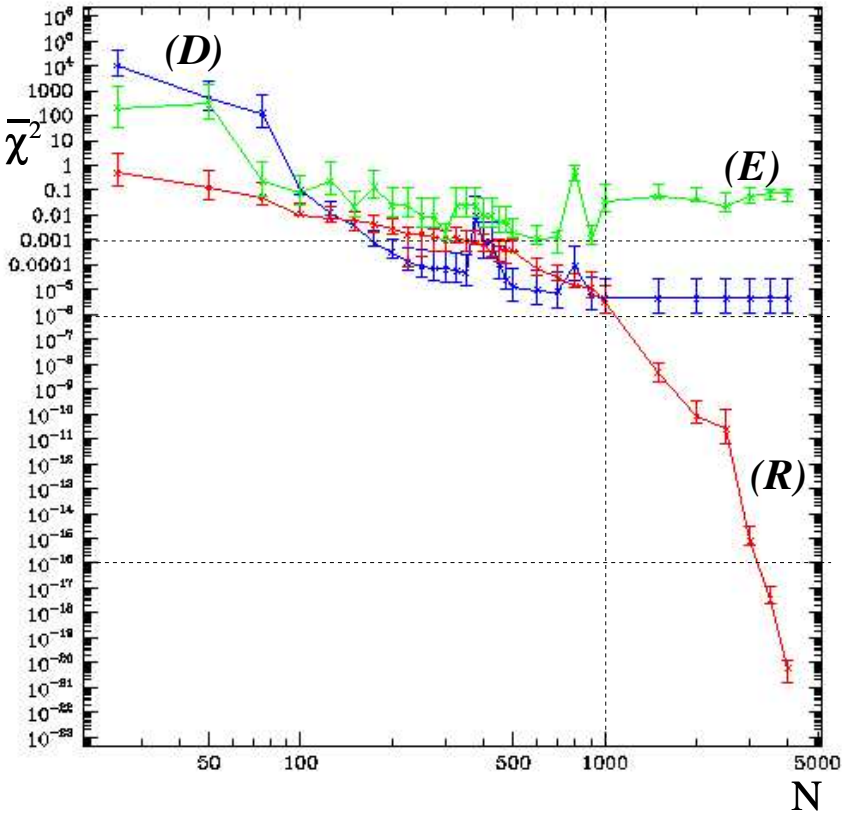



Figura 77: Eficiencia de las estrategias *Down Hill Simplex* ***(D)***, *Recocido Simulado* ***(R)*** y *Evolutiva* ***(E)*** para la estructura (**c**) de la imagen sintética de prueba, ver Figura 70.

la calidad del ajuste, tal que valores de magnitud mayor a los mostrados en estas figuras pueden ser considerables como aceptables en imágenes reales o estructuras sintéticas que presenten cierta patología, vea por ejemplo el valor de $\chi^2$ para la estructura (d) de la imagen sintética de prueba. Sin embargo, $\chi^2$ nos ayuda a visualizar el comportamiento de la precisión en imágenes sintéticas, y por consiguiente predecir su comportamiento en imágenes adquiridas con algún sistema de adquisición.

El término *eficiencia* es un compromiso entre el trabajo realizado o la rapidez de cada algoritmo y la precisión que se desea alcanzar. Para poder calificar a los algoritmos, debemos definir intuitivamente ciertos límites en cuanto al término precisión y rapidez.

Estos límites son:

1. Precisión:

   (a) *Pobre.* Una precisión pobre la definimos cuando $\chi^2 \geq 1 \times 10^{-3}$. Esto implica que se emplearan aproximadamente tres cifras significativas en los cálculos de punto flotante.

   (b) *Aceptable.* Una precisión aceptable la definimos cuando $1 \times 10^{-3} < \chi^2 \leq 1 \times 10^{-6}$. Esto implica que se emplearan aproximadamente seis cifras significativas en nuestros cálculos. En términos de precisión en operaciones con punto flotante, equivale aproximadamente a emplear números flotantes de "precisión sencilla".

   (c) *Superior.* Una precisión superior la definimos cuando $1 \times 10^{-6} < \chi^2 \leq 1 \times 10^{-16}$. Esto implica que se emplearan aproximadamente 16 cifras significativas en nuestros cálculos. En términos de precisión en operaciones con punto flotante, equivale a emplear números flotantes de "precisión doble".

   (d) *Indeterminada.* Una precisión indeterminada la definimos cuando $\chi^2 < 1 \times 10^{-16}$. Para marcar los límites anteriores, se determinaron las características de la aritmética de punto flotante del equipo en el cual se realizaron los experimentos, vea Tabla VI.4. Como se mencionó en el Cápitulo V, los algoritmos se codificaron basados en variables de precisión doble. Los valores mostrados en la Tabla VI.4 indican que el equipo donde se ejecutaron las pruebas cumple con el estándar IEEE-754 para variables de precisión doble, ver referencia [Press, 1998].

   El número más pequeño representado por la mantiza, ver VI.4.4, es un parámetro importante. Este valor indica cual es el menor número sin errores por redondeo[1] que la computadora puede manejar. Por lo tanto, valores inferiores a $1 \times 10^{-16}$, consideramos que están afectados por errores de redondeo y no corresponden exactamente a los valores proporcionados por el detector preciso de esquinas múltiples o los métodos de optimización codificados en SIDEE. Es por ello, que lo hemos marcado como un rango de precisión *indeterminada.*

[1] *rounfoff* en inglés

Tabla VI.4: Características de la representación del punto flotante utilizada en los algoritmos de optimización y el detector de esquinas múltiples.

| | Concepto | Precisión doble |
|---|---|---|
| VI.4.1 | Base | 2 |
| VI.4.2 | Número de dígitos en la mantiza | 53 |
| VI.4.3 | Número más pequeño representado en la mantiza | -52 |
| VI.4.4 | Número decimal más pequeño representado por la mantiza | $2.22 \times 10^{-16}$ |
| VI.4.5 | Exponente | 11 bits |
| VI.4.6 | Número mayor representado por el exponente | 1024 |
| VI.4.7 | Número menor representado por el exponente | -1022 |
| VI.4.8 | Número decimal mayor | $1.80 \times 10^{+308}$ |
| VI.4.9 | Número decimal menor | $2.23 \times 10^{-308}$ |

Sin embargo, muestran también que los algoritmos codificados pueden alcanzar valores superiores a precisión de la mantiza que estamos usando, indicando con ello la confiabilidad de nuestro código.

2. Rapidez: Para tener una idea de la rapidez de los algoritmos de optimización se calculó el tiempo que tarda cada uno de ellos en la generación de las figuras de esta sección. Dichos datos se almacenan en el archivo de resultados explicado en la Tabla V.9. Los tiempos unitarios son: 1) el algoritmo *Down Hill Simplex* requiere en promedio de 0.039 segundos en cada paso, 2) el algoritmo *Recocido Simulado* requiere en promedio de 0.035 segundos en cada paso y 3) el algoritmo *Evolutivo* requiere en promedio de 0.072 segundos en cada generación de 10 individuos, tal que para la generación de un nuevo individuo, aplicar los operadores de mutación y apareamiento, selección y evaluación, requiere de 0.0072 segundos. Es evidente que la estrategia *Evolutiva* es aproximadamente 1.9 veces más lenta que los otros dos algoritmos. Sin embargo, el proceso estocástico unitario para generar un individuo en el algoritmo *Evolutivo* es aproximadamente 5.1 veces más rápido que *Down Hill Simplex* y *Recocido Simulado*.

   (a) *Rápido.* Menos de 1000 pasos. Esto implica que se espera encontrar los

parámetros de una esquina en aproximadamente 37 segundos para *Recocido Simulado* y *Down Hill Simplex*.

(b) *Aceptable*. Entre 1000 pasos y 4000 pasos. Esto implica que se espera encontrar los parámetros de una esquina en aproximadamente 2.5 minutos.

Cada uno de los límites previamente definidos se marcan con líneas punteadas en la figuras 75, 76 y 77. Analizando estas figuras, observamos lo siguiente:

- *Down Hill Simplex*. Esta estrategia alcanza generalmente una precisión superior antes de las 1000 interaciones. En las figuras 75 y 76, su curva de precisión se observa generalmente por abajo de los otros dos métodos (mejor respuesta). Sin embargo, antes de los primeros 100 pasos, los movimientos geométricos definidos por la estrategia de Nelder y Mead, tiene una menor precisión que los otros dos métodos. Después de los 1000 pasos la precisión del algoritmo se estabiliza, por ejemplo en la Figura 75 la curva permanece constante en $\overline{\chi^2} \approx 7.29 \times 10^{-21}$ y en la Figura 77, $\overline{\chi^2} \approx 4.5 \times 10^{-6}$. Por otro lado, en la misma en la Figura 77, se puede observar la limitante de este algoritmo. Obsérvese, como después de aproximadamente 1000 pasos el algoritmo es incapaz de "escapar" a un extremo local. Esto se debe a la búsqueda geométrica del óptimo inherente en método *Down Hill Simplex*. En oposición, obsérvese que el algoritmo *Recocido Simulado* "escapa" a dicho extremo local, en otras palabras, existe y es computacionalmente posible encontrar un mejor conjunto de valores de $\vec{P}$ en $M'_L(x, y, \vec{P})$ para representar la estructura (c) de la imagen sintética de prueba. Finalmente, la estrategia Nelder y Mead no contiene parámetros de ajuste para lograr un mejor desempeño en su funcionamiento, los otros dos algoritmos si tienen esta habilidad, lo que implica que su respuesta es la misma en todos los casos.

- *Recocido Simulado*. Esta estrategia alcanza una precisión superior antes de las 1000 iteraciones. La curva de precisión de este método mejora gradualmente mientras se aumenta el número de pasos. Más aún, después de las 1000 iteraciones el esquema de equilibrio térmico exitado por un movimiento "Browniano" y dirigido por los movimientos geométricos del simplex, y el esquema de descenso de temperatura, responden de mejor manera a los extremos locales. Obsérvese como en las figuras 75, 76 y 77, cada muestra mejora generalmente con respecto a la otra. En partícular,

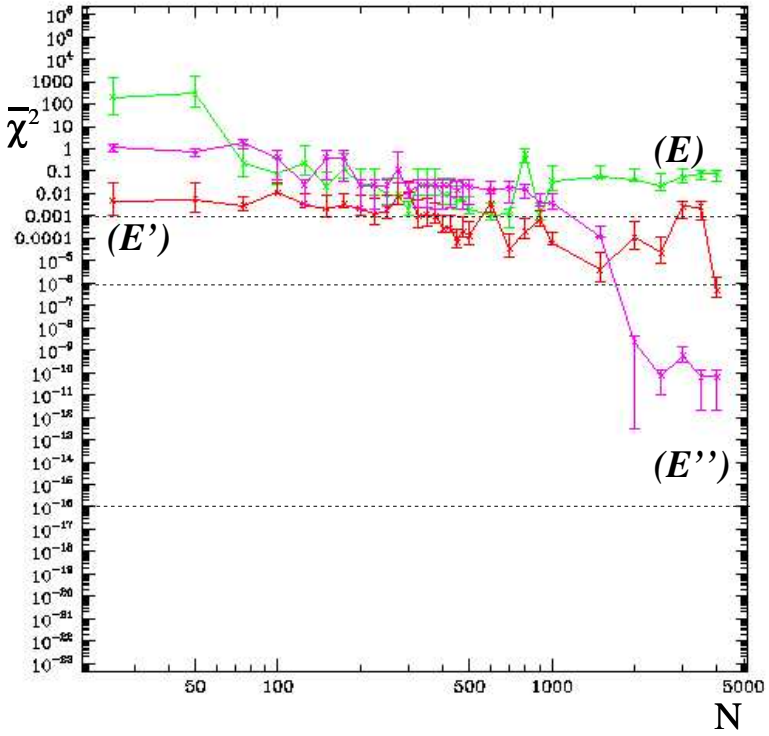



Figura 78: Comportamiento de la eficiencia de la estrategia *Evolutiva* para la estructura (**c**) de la imagen sintética de prueba. ***(E)*** curva de comportamiento para las condiciones iniciales experimentales. ***(E')*** curva de comportamiento cuando la población es de 30 individuos. ***(E")*** curva de comportamiento para 30 individuos, un 60% de apareamiento y un 45% de mutación.

en la Figura 77 el algoritmo *Recocido Simulado* es el único que mejora el valor de $\chi^2$ en la región donde los otros dos métodos se estabilizan.

- *Evolutivo.* Un algoritmo evolutivo es una técnica nueva que se basa en una heurística que contiene elementos estocásticos y determinísticos donde el proceso de selección determina la conducción de estos algoritmos. Para las dos estrategias anteriores, la conducción está dada por un proceso geométrico determinístico y para *Recocido Simulado* se incluye un proceso estocástico (movimiento Browniano). En el caso de nuestro algoritmo evolutivo la conducción está dada por un proceso es-

tocástico, selección por ruleta, y determínistico, selección por torneo. En la Figura 75 se observa como después de las 500 iteraciones los procesos *Recocido Simulado* y *Evolutivo* se comportan de forma similar logrando una precisión superior, lo que muestra la viabilidad de la aplicación de la tecnica evolutiva. Cuando el número de pasos es inferior a 100, el proceso *Evolutivo* responde generalmente mejor que los otros dos métodos. Sin embargo, en las gráficas 76 y 77 el algoritmo evolutivo es el que presenta la peor curva de precisión. Más aún, en la Figura 77, después de la 1000 iteraciones el algoritmo encuentra un extremo local y se ve imposibilitado de superarlo. Dicha observación, sugiere la existencia de un *supercromosoma* que domina el proceso evolutivo. Este es un problema común en este tipo de algoritmos [Gen, 1996]. Para resolver la creación de supercromosomas podemos tomar las acciones siguientes:

1. Aumentar la diversidad genética. Esto se logra aumentando, el número de pobladores en las generaciones.
2. Aumentar el porcentaje de mutación en los individuos.
3. Disminuir la presión sobre la selección. Esto es, utilizar únicamente la selección por ruleta o la selección por torneo. O bién, alguna otra estrategia de selección.

Las primeras dos acciones equivalen a ejecutar **Modelo_L_Evolutivo**, modificando los parámetros de control del algoritmo. La tercera, requiere de una recodificación de la librería **Evolutivo_L.so**. Para mostrar el efecto de las dos primeras acciones, se generó la Figura 78, en la cual, la curva (***E***) es el comportamiento del proceso *Evolutivo* en las condiciones experimentales inicales, la curva (***E'***) es el comportamiento de (***E***) para 30 individuos y la curva (***E"***) es el comportamiento de (***E'***) para $pa = 0.6$ y $pm = 0.45$. Obsérvese como (***E'***) responde mejor en los primeros 100 pasos, esto es, existe más diversidad genética. Por otro lado, (***E"***) responde mejor a la existencia de extremos locales o a la generación de *supercromosomas* en evolución del algoritmo. Sin embargo, es menos eficiente en las primeras 100 iteraciones. En ambos casos, la mejor precisión se obtiene después de las 1000 iteraciones.

En conclusión la estrategia *Recocido Simulado* tiene un mejor balance entre el trabajo realizado, la precisión alcanzada y la habilidad de escapar a extremos locales, en la

detección del punto exacto de esquina $(x_e, y_e)$ a través de nuestro detector preciso de esquinas-L $M'_L(x, y, \vec{P})$.

### VI.2.6 Comportamiento del Modelo con Ruido

Para efectos de mostrar la robustez de cada uno de los algoritmos y la estabilidad de nuestro modelo preciso de esquinas múltiples $M'_L(x, y, \vec{P})$, se adiciona ruido Gaussiano con una varianza unitaria y una media cero. Dicho ruido se escala por una constante $\lambda$, ver Ecuación V.1, a fin de producir perturbaciones en la imagen sintética de prueba mostrada en la Figura 70. Este ruido es conocido como ruido *aditivo* [Dougherty, 1999]. La relación señal a ruido $(SR)$ es calculada en decibeles $(_{DB})$ acorde a la Ecuación V.2. En la Figura 79 se muestran las estructuras (a), (b) y (c) para los factores de escala del ruido Gaussiano $\lambda = 20, 40, 80$. La Tabla VI.5 muestra las equivalencias del ruido Gaussiano escalado por la constante $\lambda$ y su valor correspondiente en decibeles. Es importante enfatizar, que cuando $SR \rightarrow 1$ implica que el error en la imagen con ruido es aproximadamente igual a la cantidad de señal en la imagen sintética. Si $SR < 1$ implica que el ruido es mayor a la señal original. Por otro lado, cuando $SR >> 1$ la imagen sintética original y la imagen sintética con ruido Gaussiano son semejantes, el error tiende a cero. La prueba se construyó de la manera siguiente:

1. Se aplicaron los algoritmos *Down Hill Simplex*, *Recocido Simulado* y *Evolutivo* a las estructuras (a), (b) y (c) de la Figura 70. Dichas estructuras muestran tres esquinas distintas: esquina con un ángulo recto, esquina con un ángulo agudo y esquina con un ángulo obtuso, gráficos (a), (b) y (c) respectivamente.

2. Las cotas del simplex o población inicial de estas estructuras se muestran en la Tabla VI.1.

3. El tamaño de la ventana es de $13 \times 13$ pixeles, centradas en el pixel:

   (a) $(u_0, v_0) = (87, 87)$.

   (b) $(u_0, v_0) = (707, 353)$.

   (c) $(u_0, v_0) = (396, 397)$.

4. Los parámetros de control de los algoritmos son:

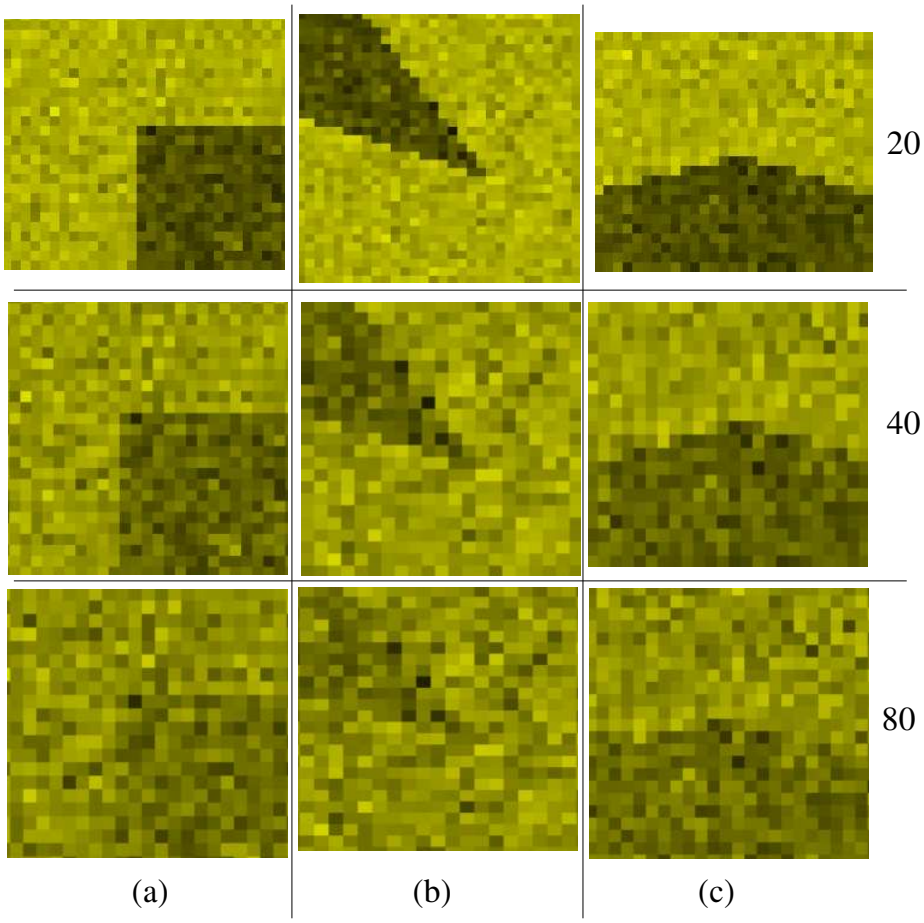



Figura 79: Estructuras (a), (b) y (c) de la imagen sintética de prueba, con un ruido Gaussiano escalado por $\lambda = 20, 40, 80$.

- *Down Hill Simplex*. Número máximo de movimientos del simplex $N = 4500$.
- *Recocido Simulado*. Temperatura de inicio $T = 1$. Longitud del estado de equilibrio $I = 20$. Número máximo de iteraciones $N = 4500$.
- *Evolutivo*. Porcentaje de apareamiento $pa = 0.80$. Porcentaje de mutación $pm = 0.05$. Porcentaje de convergencia $pc = 0.75$. Número de individuos en la población 22. Número máximo de generaciones $N = 2000$, lo que equivale a 4500 movimientos aproximadamente.

5. Se efectuaron 30 realizaciones por muestra.

6. Los factores de escalamiento del ruido Gaussiano utilizados en esta prueba, se muestran en la Tabla VI.5. Dado que para cada muestra, se genera un ruido Gaussiano aleatorio aplicado en la vecindad de $(u_0, y_0)$, el ruido en decibeles reportado en la Tabla VI.5, es el promedio $\overline{SR_{DB}}$ de todas las muestras, para un valor de $\lambda$ constante. Así mismo se reporta su desviación estándar $s$.

Tabla VI.5: Ruido Gaussiano escalado $\lambda$ y su equivalencia en descibeles $SR_{DB}$

| $\lambda$ | (a) | | (b) | | (c) | |
|---|---|---|---|---|---|---|
| | $\overline{SR_{DB}}$ | $s$ | $\overline{SR_{DB}}$ | $s$ | $\overline{SR_{DB}}$ | $s$ |
| 0 | — | — | — | — | — | — |
| 20 | +1.0802e+01 | +5.1827e-02 | +1.0541e+01 | +9.4071e-02 | +1.0872e+01 | +3.1603e-04 |
| 40 | +6.6861e+00 | +5.1827e-02 | +6.3738e+00 | +9.4072e-02 | +6.8168e+00 | +3.1603e-04 |
| 60 | +4.7257e+00 | +5.1827e-02 | +4.3840e+00 | +9.4071e-02 | +4.8968e+00 | +3.1603e-04 |
| 80 | +3.5499e+00 | +5.1827e-02 | +3.1848e+00 | +9.4071e-02 | +3.7501e+00 | +5.9104e-05 |
| 100 | +2.7596e+00 | +5.1827e-02 | +2.3768e+00 | +9.4071e-02 | +2.9814e+00 | +3.1603e-04 |

7. La región con ruido Gaussiano $I_r(i, j)$ se normalizó a los valores de intensidad de $[0, 255]$ en números reales, a partir de la siguiente función de transformación:

$$I_n(j,i) = \frac{255}{max(I_r(j,i)) - min(I_r(j,i))} I_r(j,i) \qquad i,j = 1,\ldots,2w+1 \qquad \text{(VI.2)}$$

donde $I_n(j, i)$ es la región en estudio normalizada que incluye el ruido Gaussiano escalado.

8. La tolerancia del criterio de paro, Ecuación (IV.39), es $ftol = 1 \times 10^{-09}$.

A partir de estas consideraciones se generaron las figuras 80, 81 y 82, que muestran el desplazamiento en la localización del punto de esquina $(u_e, v_e)$ de las estructuras (a), (b) y (c) respectivamente, de la imagen sintética de prueba en presencia de un ruido Gaussiano aleatorio. En cada una de estas figuras, se grafíca el punto de esquina promedio $(\overline{u_e}, \overline{v_e})$ y su desviación estándar resultante de las 30 realizaciones de las tres estrategias de optimización, para los valores de $\lambda$ mostrados en la Tabla VI.5. Los gráficos ubicados en la parte izquierda de cada una de estas figuras, corresponden a la coordenada $u$ y los ubicados en la parte derecha a la coordenada $v$, del sistema coordenado de la imagen

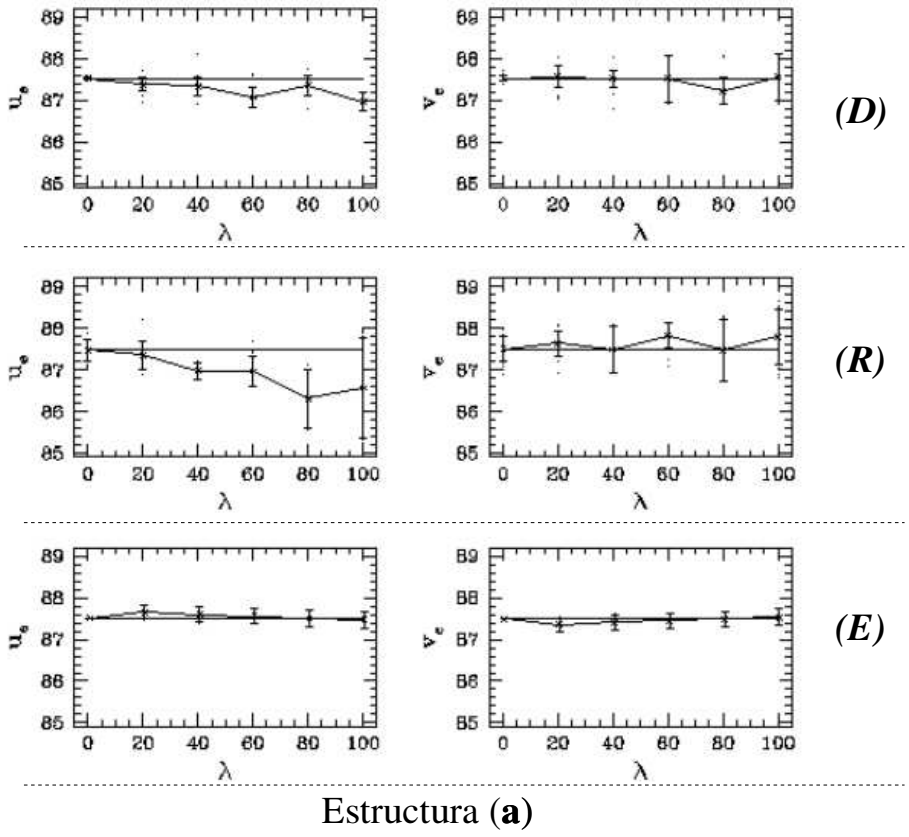



Figura 80: Comportamiento de los algoritmos *Down Hill Simplex* ***(D)***, *Recocido Simulado* ***(R)*** y *Evolutivo* ***(e)*** en presencia de un ruido aleatorio Gaussino escalado por un factor $\lambda$, para la estructura (a) de la imagen sintética de prueba.

$(u, v)$. La recta horizontal, indica la posición del punto $(u_e, v_e)$ de la imagen sintética sin ruido Gaussiano ($\lambda = 0$). Analizando estas gráficas concluimos que:

1. A partir de $\lambda = 80$ ($SR_{DB} \approx 3.5$), ver Figura 79, los bordes de las estructuras (a), (b) y (c) son difíciles de distingir. Sin embargo, el ruido Guaussino aleatorio no suaviza o difumina a los bordes. Esto implica, que la estructura conserve más o menos su forma. Por lo tanto, podemos analizar la habilidad de cada una de las estrategias de optimización, para integrar o reconstruir en su conjunto a cada una de las estructuras.

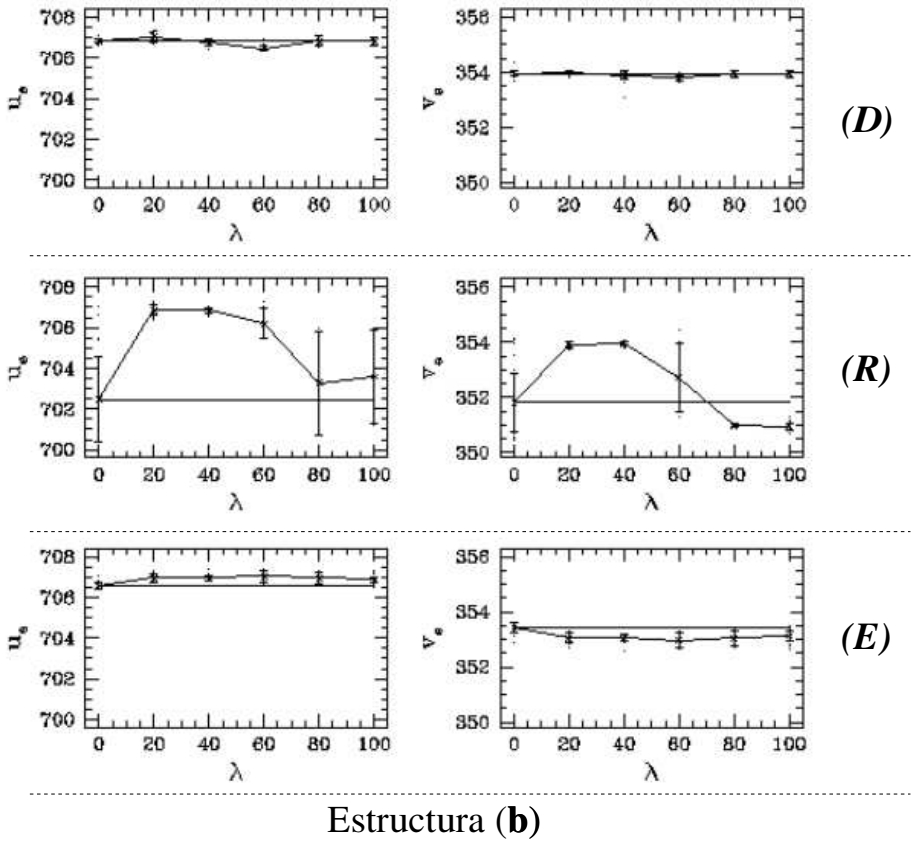



Figura 81: Comportamiento de los algoritmos *Down Hill Simplex* ***(D)***, *Recocido Simulado* ***(R)*** y *Evolutivo* ***(E)*** en presencia de un ruido aleatorio Gaussino escalado por un factor $\lambda$, para la estructura (b) de la imagen sintética de prueba.

2. Para el caso de la estructura (a) el algoritmo *Evolutivo* presenta la mejor curva de comportamiento en presencia del ruido. Más aún, la máxima desviación sobre el promedio se localiza cuando $\lambda = 20$. El valor es de aproximadamente 0.14 pixeles.

3. Para el caso de la estructura (b) el algoritmo *Down Hill Simplex* presenta la mejor curva de comportamiento en presencia del ruido. La desviación máxima sobre el promedio se ubica en $\lambda = 60$, ocurrida sobre el eje $u$. Este valor es de 0.37 pixeles. Mientras que el algoritmo *Evolutivo* reporta una desviacióm máxima de 0.47 pixeles ocurrido también en $\lambda = 60$. Así mismo, la curva del comportamiento de la

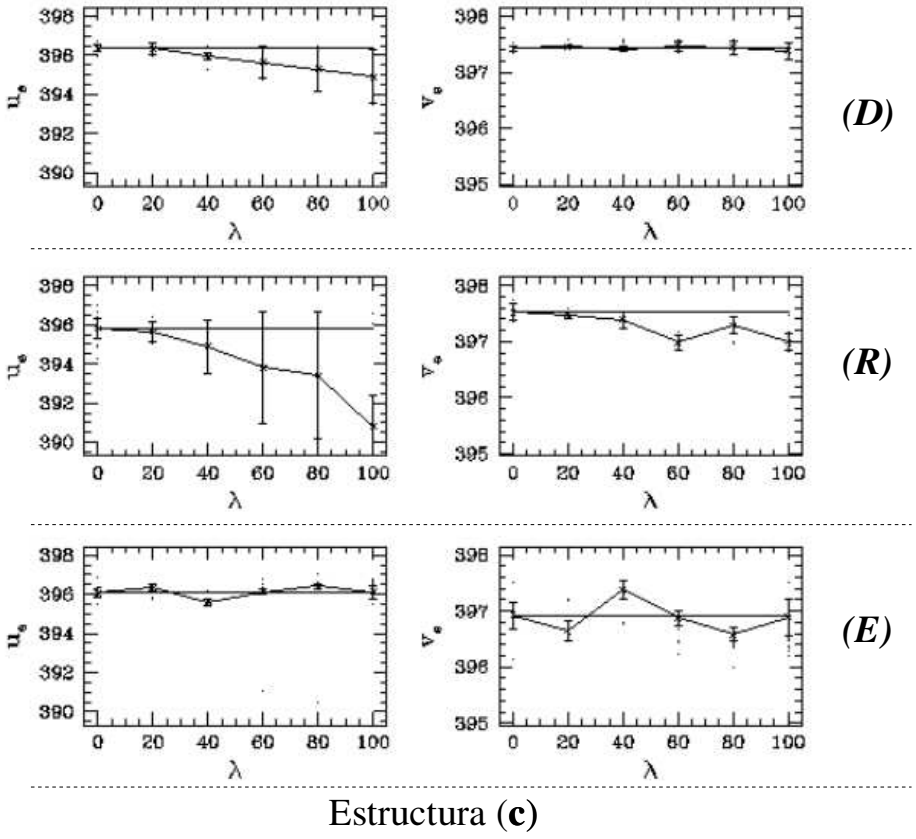



Figura 82: Comportamiento de los algoritmos *Down Hill Simplex* ***(D)***, *Recocido Simulado* ***(R)*** y *Evolutivo* ***(E)*** en presencia de un ruido aleatorio Gaussino escalado por un factor $\lambda$, para la estructura (c) de la imagen sintética de prueba.

estrategia *Evolutiva* permanece constante sobre el valor del promedio de $(x_e, y_e)$ en todos los casos.

4. Para el caso de la estructura (c) el algoritmo *Evolutivo* presenta el mejor comportamiento en presencia del ruido sobre el eje $u$. Mientras que, el algoritmo *Down Hill Simplex* presenta la mejor curva sobre el eje $v$.

5. Si observamos las "barras" que representan la desviación estándar de las 30 realizaciones por muestra, para el caso de las tres enstructuras, el algoritmo *Evolutivo* presenta el mejor promedio para cada realización. Esto es: 1) la desviación estándar

promedio para el algoritmo *Evolutivo* es de 0.19 pixeles, 2) la desviación estándar promedio para la estrategia Nelder y Mead es de 0.25 pixeles, y 3) la desviación estándar promedio para el algoritmo *Recocido Simulado* es de 0.76 pixeles.

Por lo anteriormente expuesto, el algoritmo *Evolutivo* es menos sensible al ruido y por lo tanto más robusto. No obstante, la estrategia *Down Hill Simplex* se comporta de forma similar que la anterior. Finalmente, la estrategia *Recocido Simulado* muestra el peor comportamiento en presencia de ruido Gaussiano aleatorio.

## VI.3 Conclusiones

Durante el desarrollo de este trabajo, se ha planteado la dificultad en el proceso de detección y localización exacta de una estructura llamada esquina, en imágenes digitales. Los factores que provocan el grado de incertidumbre en la ubicación de la esquina son principalmente: 1) la posición y orientación de la cámara con respecto al objeto, 2) la orientación interior propia de la cámara y la geometría intrínseca de las rejillas fotosensibles con que se contruyen dichas cámaras, 3) fluctuaciones en la iluminación de la escena, 4) diversos factores ópticos.

Tomando en consideración estos factores y después de una revisión bibliográfica de la literatura especializada, se ha planteado *Un Nuevo Detector Sub-pixel Paramétrico de Esquinas Múltiples* con las siguientes características.

1. El modelo se basa en una *Función Unitaria de Borde* (FUB), que carateriza completamente las propiedades físicas y geométricas de un borde, en una imagen digital en tonos de gris.

2. Aplicando operaciones aritméticas simples (suma, resta, división y multiplicación) entre las FUB, es posible modelar matemáticamente estructuras más complejas como son: esquinas-L y vértices. La facilidad con la que se pueden generar estructuras nuevas es una caraterística importante de nuestro análisis.

3. El detector de esquinas-L tiene la habilidad de modelar el efecto causado por sensores CCD's rectangulares, en la dirección de los ejes principales del plano de la rejilla fotosensible que contiene generalmente una cámara digital. Elemento que no hemos encontrado en otros modelos publicados previamente.

4. A partir de la geometría observada en el modelo de esquina-L hemos planteado un criterio nuevo de ubicación exacta de esquina, el cual es: *"El punto exacto de esquina, se ubica en el punto que representa la mínima distancia euclidiana entre el punto de intersección de las rectas de borde y las coordenadas que satisfacen a la curvatura central de nuestro modelo"*.

5. Nuestro modelo opera sobre una región cuadrángular definida por el usuario, en la cual: 1) cada ángulo de la FUB es independiente, lo que nos permite modelar esquinas con un ángulo de apertura en el intervalo $(0^o, 180^o)$, 2) la esquina se mueve libremente en la región explorada, 3) los niveles de intensidad en tonos de gris se ajustan automáticamente a la región de interés.

La determinación del punto exacto de una esquina-L a través de nuestro modelo paramétrico $M'_L(x, y, \vec{P})$ se ha planteado como un problema de optimización. Esto es, encontrar el mejor conjunto de parámetros $\vec{P} = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$, que se ajusten mejor a una ventana cuadrada de $2w - 1 \times 2w - 1$ pixeles alrededor de un pixel $(u_0, v_0)$ en una imagen digital. Los elementos utilizados para este fin, son los siguientes:

1. Una función objetivo o de mérito que llamamos *Criterio de Optimización*. El criterio de optimización, es el estimado de máxima probabilidad, conocido como *Ajuste de Mínimos Cuadrados*, que representa al conjunto de $\vec{P}$ que se adapta mejor a la región en estudio de la imagen digital. El valor de $\chi^2$ se obtiene por medio del método propuesto por Levenberg-Marquardt. El método de Levenberg-Marquardt es un algoritmo de optimización local.

2. Un conjunto de ecuaciones de tranformación, ver Ecuación (IV.11), que relaciona el sistema coordenado del modelo, el sistema coordenado de la imagen y su notación matricial.

3. Un conjunto de estrategias de optimización global para modelos no lineales con distintas características entre ellos. Estos métodos son:

   (a) Estrategia *Down Hill Simplex*. La búsqueda de un óptimo global, en este método, obedece a movimientos geométricos previamente definidos de una estructura llamada *simplex*. Para nuestro caso, el simplex es un poliédro con 9 vértices.

(b) Estrategia *Recocido Simulado*. La búsqueda de un óptimo global, en este método, obedece a la simulación del tratamiento térmico de los metales, llamado forja. En donde se define un esquema de descenso de temperatura de la estructura y un esquema de equilibrio. Para nuestro caso: 1) el esquema de descenso de temperatura es $T_{nueva} = 0.8 \times T_{anterior}$, 2) el esquema de equilibrio está dado por los movimientos geométricos de un *simplex* de 9 vértices, perturbados por un movimiento Browniano.

(c) Estrategia *Evolutiva*. La búsqueda de un óptimo global, en este método, obedece a la evolución generacional de los integrantes de una población acorde al proceso de "selección natural" propuesto por Charles Darwin y conceptos Biológicos de la Genética. Para nuestro caso se emplea: 1) un apareamiento a través de una *combinación intermedia afin extendida*, 2) una mutación *dinámica*, 3) una selección por *torneo* y *ruleta*. Los operadores de mutación y apareamiento se integran en un conjunto de transformaciones, ver ecuaciones (IV.37) y (IV.38), que proporcionan un eficiente tratamiento de los operadores evolutivos.

4. Como resultado de aplicar $Q$ a las estrategias *Down Hill Simplex*, *Recocido Simulado* y *Evolutivo*, contamos con un método de optimización local inmerso en un proceso de optimización global. Por consecuencia, nuestros procesos de optimización global están acelerados por el método de Levenberg-Marquardt.

5. Finalmente, para determinar el punto exacto de esquina-L $(x_e, y_e)$ a partir de nuestro *Criterio de Ubicación Exacta de esquina*, ver sección III.4.4, se realiza a través de la estrategia propuesta por Nelder y Mead.

Cada una de estas estrategias de optimización y nuestro detector preciso de esquinas múltiples se integraron en un conjunto de librerías dinámicas compartidas que llamamos *Sistema Integral para la Detección Exacta de Esquinas (SIDEE)*. Esta organización permite:

1. Aislar los métodos de optimización, la evaluación del modelo y sus derivadas, y a nuestro criterio de ubicación precisa de esquina-L, de la complejidad del manejo y adquisición de una imagen digital. Por ejemplo: 1) el manejo de distintos formatos

de una imagen digital, 2) los mecanísmos de desplegado de una imagen en el ambiente X-Windows, 3) el manejo de las tarjetas de adquisición en la obtención de escenas del mundo real.

2. Segmentar las funciones o rutinas tal que puedan ser usadas desde diferentes programas (código reusable).

3. Agregar, rutinas o funciones que permitan nuevas habilidades de SIDEE.

El detector paramétrico de esquinas múltiples ha sido probado sobre imágenes reales y sintéticas utilizando distintos métodos de optimización multidimensional para modelos no lineales. De la experimentación en nuestro problema concluimos que:

1. Los algoritmos de optimización aplicados a nuestro modelo, responden eficientemente cuando se muestrea uniformemente el simplex o población de inicio en un rango definido por el usuario.

2. La estrategia de optimización que mejor responde a la relación precisión *vs.* velocidad es el algoritmo *Recocido Simulado*.

3. La estrategia que presenta un mejor comportamiento al ruido en la señal es el algoritmo *Evolutivo*.

4. La estrategia más simple de operar, en lo referente a que no requiere parámetros de control del algoritmo, es el algoritmo *Down Hill Simplex*.

5. La estrategia *Evolutiva*, tiene la capacidad de aumentar el número de pobladores, lo que redunda en aumentar la probabilidad de encontrar un mejor resultado en menor generaciones o evaluaciones del simplex. Las otras dos estrategias no tienen esa habilidad

6. Finalmente, la estrategia *Evolutiva* tiene la posibilidad de ser paralelizada más fácilmente, debido a la composición inherente de los algoritmos evolutivos [Gen, 2000].

### VI.3.1 Aplicaciones

El presente desarrollo es clasificado dentro de los procesos de bajo nivel en el campo de reconocimiento de patrones y análisis de imágenes. Este trabajo es útil en todos aquellos

procesos que requieran de una alta exactitud en la detección y ubicación espacial de puntos de control, también llamados puntos prominentes. Como por ejemplo:

1. Calibración de cámaras.
2. Empatamiento de imágenes.
3. Reconstrucción tridimensional.
4. Reconocimiento de objetos.
5. Trazado de mapas urbanos a través de imágenes aéreas.

### VI.3.2 Trabajos Futuros

Una vez demostrada la factibilidad de la aplicación del detector de esquinas sobre imágenes reales, los desarrollos a mediano plazo son los siguientes:

1. Desarrollar los algoritmos necesarios para aplicar la *función unitaria de borde* sobre esquinas más complejas, como por ejemplo, esquinas-T, esquina-K, vértices. Esto es, encontrar la combinación correcta de los operadores aritméticos $(+, -, \times, \div)$ entre las FUB, afin de producir esquinas complejas de forma autónoma. Esta habilidad, permitiría al modelo representar matemáticamente un objeto (un polígono) contenido en una imagen digital. Más aún, si se adicionan las ecuaciones de reconstrucción tridimensinal correspondiente, podría representarse matemáticamente cualquier poliédro en una escena tridimensional.

2. Aumentar la velocidad de ubicación de un punto prominente, con objeto de utilizar al detector en aplicaciones en tiempo real, por ejemplo, visión robótica. En este sentido, se visualizan dos vertientes:

   (a) Analizar la factibilidad de reducir el número de parámetros de nuestro modelo. En el capítulo III y VI, se planteó que la ubicación del punto de esquina no depende de los niveles de gris dentro y fuera de la esquina. A partir de esta propiedad, podemos eliminar la *amplitud* de la esquina $A$ y el *piso* de la misma $B$ si normalizamos a uno la estructura en estudio. Esto es, utilizar la *función unitaria de esquina* $M_L(x, y, \vec{P})$, en lugar de $M'_L(x, y, \vec{P})$. Por otro

lado, si la geometría de rejilla fotosensible **no** es rectangular, $\sigma_1 = \sigma_2$, entonces podemos reducir a uno el factor de difuminado. Por lo tanto, se minimizaran 5 parámetros en lugar de 8. Esta reducción provoca, que el punto exacto de esquina-L equivalga a calcular el punto de intersección de la recta $r_{bis}$ que bisecta el ángulo de apertura de la esquina y la curvatura central de nuestro modelo $M_L(x, y, \vec{P}) = 0.5$, en lugar de emplear un segundo proceso de optimización dado por nuestro *criterio de ubicación exacta del punto de esquina.*

(b) Finalmente, analizar la factibilidad de paralelizar los algoritmos de optimización. En particular, el algoritmo evolutivo es un buen candidato para aplicar los paradigmas de paralelización sobre él. Esto es, a partir de un conjunto de procesadores organizados a través de un sistema paralelo distribuido, como son "Parallel Virtual Machine" (PVM) o "Message Passing Interface" (MPI), es posible evaluar una gran cantidad de individuos, en un mismo intervalo de tiempo real. Logrando con ello, una mayor rapidez en la detección del punto exacto de esquina.

# Apéndice A

# Características del Sistema de Adquisición de Imágenes

## A.1 Unidad Central de Proceso

- Computadora Personal.
- Marca: DELL.
- Memoria RAM: 256 MB.
- Procesador: Intel PIII 800 Mhz.
- Disco Duro: 40 GB.
- Monitor: 19 Pulgadas.
- Sistema Operativo: Linux Red Hat 5.2, Lunux Red Hat 6.2, Windows 98.

## A.2 Cabeza Estereoscópica

### A.2.1 Cámara Digital

Dos cámaras digitales con las características siguientes:

- Marca: Pulnix

- Modelo: TM-9701
- Pixeles: 768 (H) $\times$ 484 (V)
- Tamaño de la rejilla fotosensible CCD: $11.6\mu m \times 13.6\mu m$
- Digitalizado: 525 lineas
- Mínima iluminación: 1.0 lux
- Voltaje requerido: 12 V
- Temperatura de operación: $-10°$C a $50°$C
- Tamaño: 44mm $\times$ 48.5mm $\times$ 136mm
- Peso: 323 gramos

### A.2.2 Tarjeta de Adquisición de Imágenes

Dos tarjetas de adquisición de imágenes con las características siguientes:

- Marca: Mutech.
- Modelo: M-Vision 1500.
- Tipo de Ducto: PCI.
- Memoria Video: 1 MB, programable.
- Conectores: RS-422, 16 bits diferenciasl.
- Velocidad de Transferencia: 0-40 Millones de pixeles/segundo.
- Modo de Trnasferencia: "Maestro" o "Esclavo".
- Librerias de Desarrollo: DOS, Windows 3.x, Windows 98/95, Windows NT, OS/2, Linux.

## A.3 Paquetes de Desarrollo

- Numéricos.
  - Paquete de matemática simbólica MAPLE VI, de la compañia Waterloo Maple Inc. Canada.
  - Librerías de cálculo numérico, Numerical Recipies in C.
- Gráficos
  - Image Understanding Environment, soportado por Amerinex Applied Imaging Co. (http://www.aai.com)
  - X-based Image Processing Tools and Enviromet, soporatdo por la universidad de Oslos No.
- Compiladores: gcc, g++.
- Editores de texto y procesadores de palabra:
  - TeX 3.14159, distribuido por "teTeX".
  - StartOffice, distribuido por Sun MicroSystems Inc.
  - xemacs.

# Apéndice B

# Código Fuente

Este apéndice muestra la ubicación del código fuente de las rutinas que conforman las librerías de SIDEE, así como los programas de explotación que se generaron. El presente desarrollo incluye un *disco compacto* con la estructura de archivos que a continuación se describe. Estas rutinas están escritas en el lenguaje de programación $C$ y $C++$. Todas las rutinas que se presentan están explicadas en la sección *Arquitectura del Sistema*, página 130. Finalmente, este apéndice tiene la estructura siguiente:

> En la sección *Compilación* se indica la forma general de compilar las librerías de SIDEE.
>
> En la sección *Librerías de SIDEE* se incluye los directorios donde se ubical el código fuente de estas rutinas.
>
> En la sección *Programas de Explotación* se muestra la ubicación del código de fuente de dichos programas y los archivos de inicialización de los mismos.

## B.1 Compilación

La compilación de SIDE y los programas de explotación se realizaron utilizando el comando **make** del sistema operativo "RedHat Linux". El comando **make** permite determinar autómaticamente las secciones o piezas necesarias para recompilar un programa o un sistema más complejo. El comando **make** requiere de la creación de un archivo de reglas llamado **Makefile**. Este archivo de reglas se incluye en cada una de los directorios

de las rutinas que componen a SIDEE y los programas de explotación. Para mayor referencia de la sintaxis del archivo **Makefile** y el comando **make** pueden consultarse los manuales en línea proporcionados por el sistema operativo "RedHat Linux".

## B.2 Librerías de SIDEE

Cada una de las librerías de SIDEE tiene un directorio propio. Dentro de cada directorio existe el archivo **Makefile**, el cual corresponde a las reglas de compilación de cada uno de ellos. Para compilar una librería de SIDEE se requiere posicionarse en el directorio de interés y ejecutar uno de los comandos siguientes:

**make**: compila el programa de interés. Para el caso de las librerías de SIDEE, se transporta la librería compartida al directorio *~ /SistemaIntegral/lib*. Esta librería está preparada para ser usada en programas compilados en el lenguaje $C++$.

**make para_cc**: compila la librería de interés y la transporta al directorio *~ /SistemaIntegral/lib*. Esta librería está preparada para ser usada en programas compilados en el lenguaje $C$.

**make clean**: borra la librería compartida del directorio *~ /SistemaIntegral/lib*, así como el programa objeto.

SIDEE tiene la siguiente estructura de archivos:

- *~ /SistemaIntegral/lib*: directorio que contiene todas las librerías compartidas de SIDEE. Cada uno de los programas de explotación deberán hacer referencia a este directorio para extraer las librerías que se requieran emplear.

  **libCriterio_ch2_L_c.so**
  **libCriterio_ch2_L.so**
  **libderivadas_L_c.so**
  **libderivadas_L.so**
  **libDown_hill_L_c.so**
  **libDown_hill_L.so**
  **libfunciones_c.so**
  **libfunciones.so**
  **libmisnr_c.so**
  **libmisnr.so**
  **libModelo_L_Curvatura_c.so**
  **libModelo_L_Curvatura.so**
  **libRecocido_Simulado_L_c.so**
  **libRecocido_Simulado_L.so**
  **libConfigMetodos.so**

- ˜ */SistemaIntegral/include*: directorio que contiene todos los encabezados o definición de las ruitnas inscritas en SIDEE. Cada uno de los programas de explotación deberán hacer referencia a este directorio para extraer la definición de las rutinas que se deseen emplear.

  **Criterio_ch2_L.h** **nr.h**
  **derivadas_L.h** **nrutil.h**
  **Down_hill_L.h** **nrutil.hh**
  **funciones.h** **Recocido_Simulado_L.h**
  **Modelo_L_Curvatura.h** **ConfigMetodos.h**

- ˜ */SistemaIntegral/Criterio_chi2_L*: directorio donde residen las rutinas de nuestro criterio de optimización Q.

  **Criterio_ch2_L.C**
  **Criterio_ch2_L.c**
  **Makefile**

- ˜ */SistemaIntegral/Curvatura_L*: directorio donde residen las rutinas para el cálculo del punto exacto de esquina-L.

  **Modelo_L_Curvatur.C**
  **Modelo_L_Curvatur.c**
  **Makefile**

- ˜ */SistemaIntegral/Derivadas*: directorio donde residen las rutinas que calculan las derivadas parciales de nuestro modelo de esquina-L.

  **derivadas_L.C**
  **derivadas_L.c**
  **Makefile**

- ˜ */SistemaIntegral/Down_hill*: directorio donde residen las rutinas del método de optimización *Down Hill Simplex*.

  **Down_hill_L.C**
  **Down_hill_L.c**
  **Makefile**

- ˜ */SistemaIntegral/Funciones*: directorio donde residen las rutinas de evaluación del modelo de esquina-L y su geometría.

**funciones.C**
**funciones.c**
**Makefile**

- ˜ */SistemaIntegral/Nr*: directorio donde residen las rutinas del libro *Numerical Recipes* utilizadas por SIDEE.

| | | | | |
|---|---|---|---|---|
| **covsrt.c** | **gaussj.c** | **moment.c** | **mrqmin.c** | **ran1.c** |
| **covsrt.C** | **gaussj.C** | **moment.C** | **mrqmin.C** | **ran1.C** |
| **gasdev.c** | **indexx.c** | **mrqcof.c** | **nrutil.c** | **Makefile** |
| **gasdev.C** | **indexx.C** | **mrqcof.C** | **nrutil.C** | |

- ˜ */SistemaIntegral/Recocido_Simulado*: directorio donde residen las rutinas del método de optimización *Recocido Simulado*.

  **Recocido_Simulado_L.C**
  **Recocido_Simulado_L.c**
  **Makefile**

- ˜ */SistemaIntegral/Evolutivo*: directorio donde residen las rutinas del método de optimización *Evolutivo*.

  **Evolutivo_L.C**
  **Evolutivo_L.c**
  **Makefile**

## B.3 Programas de Explotación

Cada una de los programas de explotación tienen un directorio propio, de forma similar que las librerías de SIDEE. Dentro de cada directorio existe el archivo **Makefile**, el cual corresponde a las reglas de compilación de cada uno de ellos. Para compilar un programa de explotación se requiere posicionarse en el direcorio de interés y ejecutar uno de los comandos siguiente:

**make**: compila el programa de interés.

**make *var***: En ocaciones, el programa de explotación sufre modificaciones. Para expresar estas modificaciones se agrego la regla de compilación ***var***, dentro del archivo **Makefile** que corresponde al programa de interés. Para conocer en detalle

cada una de las reglas consulte directamente a sección **Makefile** de cada uno de estos programas.

En general todos de programas de explotación requieren de las librerías de SIDEE. En particular, los programas de adquisición y desplegado de imágenes necesitan además de las librerías de XITE, las librerías de control de la tarjeta digitalizadora Mutech MV1500. Por otro lado, los programas de cálculo masivo requieren de las librerías de IUE para el manejo de los archivos de imagen (apertura, lectura, escritura y cierre) .

Los programas de explotación tienen la estructura de archivos siguiente:

- *~ /SistemaIntegral/SistemaAjuste*: programas de adquisición y desplegado de imágenes.

  **capturaDosCamaras.c** **Metodos.ini**
  **capturaDosTest.c** **tm9701d.ini**
  **Makefile**

  A su vez, las rutinas de control de la la tarjeta Mutech MV1500 y las rutinas de manipulación de memoria a través de XITE, se incluyen en el directorio *~ /SistemaIntegral/SistemaAjuste/include*.

  **MV1500_utils.h**
  **XITE_utils.h**

  Los recursos gráficos ( XWindows) de estos programas están incluidos en el directorio *~ /SistemaIntegral/SistemaAjuste/app-defaults*.

  **CApturaDosCamaras** **xitePopup** **xiteImagePopup**
  **xiteFileSel** **xiteSlice** **xitePrompt**
  **xiteImage** **xiteHisto** **xiteTerm**
  **xiteImageOverlay** **xiteImageMenubar**

- *~ /SistemaIntegral/ConfiguracionMetodos*: rutinas del manejo del archivo de configuración de métodos.

  **ConfiguracionMetodos.c**
  **ConfiguracionMetodos.C** **Makefile**

- *~ /SistemaIntegral/Sensibilidad*: comportamiento de la ubicación exacta de la esquina $(x_e, y_e)$ con respecto a los parámetros $\vec{P}$ del detector exacto de esquinas múltiples.

**Sensibilidad.C**
**Makefile**

- ~ */SistemaIntegral/Deteccion/DownHill*: ubicación de punto exacto de esquina $(x_e, y_e)$ aplicando el método propuesto por Nelder y Mead.

  **Modelo_L_DownHill.C** **Modelo_L_DownHill_gauss.C**
  **Modelo_L_DownHill_ruido.C** **Makefile**

- ~ */SistemaIntegral/Deteccion/Recocido*: ubicación de punto exacto de esquina $(x_e, y_e)$ aplicando el método de *Recocido Simulado*

  **Modelo_L_Recocido.C** **Modelo_L_Recocido_iso_ruido.C**
  **Modelo_L_Recocido_iso.C** **Makefile**

- ~ */SistemaIntegral/Deteccion/Evolutivo*: ubicación de punto exacto de esquina $(x_e, y_e)$ aplicando el método *Evolutivo*

  **Modelo_L_Evolutivo.C**
  **Makefile**

# Bibliografía del Autor

## Conferencias Nacionales con Comité de Lectura

- Hernández, B. y Olague, G. (2001). "Un Nuevo Detector Sub-Pixel Paramétrico de Esquinas Múltiples", *En Memorias del VI Taller Iberoamericano de Reconocimiento de Patrones TIARP 2001.* México D.F. pp. 37-49. Este artículo fue elegido dentro de los 5 mejores trabajos en aportaciones científicas y tecnológicas, además de contar con una invitación a publicarlo en la revista Computación y Sistemas.

## Conferencias Internacionales con Comité de Lectura

- Olague, G. y Hernández, B. (2002). "Flexible Model-based Multi-corner Detector for Accurate Measurements and Recognition", *16th International Conference on Patttern Recognition.* IEEE Computer Society Press. pp. 578-583, Vol. 2, 11-15 August. Québec, Canada.

- Olague, G. y Hernández, B. (2001). "Autonomous Model Based Corner Detection using Evolutionary Algorithms". *In American Society for Photogrammetry and Remote Sensing.* ASPRS Annual Conference 2001.

# Referencias


[Bäck, 1994] Bäck, T. (1994). "Selective pressure in evolutionary algorithms: a characterization of selection mechanisms." *In Proc. of the first IEEE Conference on Evolutionary Computation.* IEEE Press, Orlando, FL. 57-62.

[Baker, 1998] Baker, S., y Nayar, S. (1998). "Parametric Feature Detection." *International Journal of Computer Vision 27(1)*, 27-50. Kluwer Academic Press.

[Beaudet, 1978] Beaudet, P. R. (1978). "Rotationally Invariant Image Operators." *In Proc. Of the International Conference on Pattern Recognition.* 579-583.

[Böe, 1998a] Böe, S. (1998). XITE X-based Image Processing Tools and Environment; Programmer's Manual for version 3.4. Image Processing Laboratory, Department of Informatics, University of Oslo, Report No. 92.

[Böe, 1998b] Böe, S. (1998). XITE X-based Image Processing Tools and Environment; System Administrator's Manual for version 3.4. Image Processing Laboratory, Department of Informatics, University of Oslo, Report No. 91.

[Böe, 1998c] Böe, S., Lönnestad T., Milvang O. (1998). XITE X-based Image Processing Tools and Environment; User's Manual for version 3.4. Image Processing Laboratory, Department of Informatics, University of Oslo, Report No. 56.

[Deriche, 1993a] Deriche, R., y Blazka, T. (1993). "Recovering and Characterizing Image Features Using an Efficient Model Based Approach." *In Proc. of the Conference on Computer Vision and Pattern Recognition.* New York. 530-535.

[Deriche, 1993b] Deriche, R., y Giraudon, G. (1993). "A Computational Approach for Corner and Vertex Detection." *International Journal of Computer Vision.* 10(2), 101-124. Kluwer Academic Publishers.

[Dougherty, 1999] Dougherty, E. R. (1999). Random Processes for Image and Signal Processing. SPIE Optical Engineering Press, and IEEE Press, Inc..

[Dreschler, 1982] Dreschler, L., y Nagel, H. H. (1982). "On the Selection of Critical Points and Local Curvature Extrema of Region Boundaries for Interframe Matching." *In Proc. Of the International conference on Pattern Recognition.* 542-544.

[Faugeras, 1999] Faugeras, O. (1999). Three-Dimensional Computer Vision: a geometric Viewpoint. MIT Press, Cambridge, Massachusetts, third printing.

[Fogel, 1966] Fogel, L., Owens, A., y Walsh, M. (1966). Artificial Intelligence Through Simulated Evolution. John Wiley & Sons, Inc.

[Gen, 1996] Gen, M., y Cheng R. (1996). Genetic Algorithms and Engineering Design. John Wiley & Sons, Inc.

[Gen, 2000] Gen, M., y Cheng R. (2000). Genetic Algorithms and Engineering Optimization. John Wiley & Sons, Inc.

[Gonzalez, 1987] Gonzalez, R., y Wintz, P. (1987). Digital Image Processing. Second Edition. Addison-Wesley Publishing Company.

[Gruen, 1985] Gruen, A. W. (1985). "Adaptive Least Squares Correlations: A powerful Image Matching Technique. S." *Afr. Journal of Photogrametry, Remote Sensing and Cartography.* 14(3). 175-187.

[Hernández, 2001] Hernández, B., y Olague, G. (2001). "Un detector sub-pixel paramétrico de esquina múltiples." *En memorias del VI Taller Ibereamericano de Reconocimiento de Patrones.* México D.F. 37-49.

[Holland, 1992] Holland, J. (1992). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

[Kitchen, 1982] Kitchen, L., y Rosenfel, A. (1982). "Gray Level Corner Detection." *Pattern Recognition Letters.* 95-102. No. 1.

[Koza, 1992] Koza, J. R. (1992). Genetic Programming MIT Press, Cambridge, MA.

[Lerner, 1999] Lerner, R. A. (1999). Image Understanding Environment. Amerinex Applied Imaging.

[Medioni, 1987] Medioni, G., y Yosumoto, Y. (1987). " Corner Detection and Curve Representation using cubic B-spline." *Computer Vision, Graphics and Image Processing.* 39, 267-278.

[Michalewicz, 1992] Michalewicz, Z. (1992). Genetic Algorithm + Data Structure = Evolution Programs. Springer-Verlag, New York.

[MV-Vision, 1998] MV-Vision 1500 (1998). Hardware Technical Reference. SDK Library Reference. Mutech Corporation.

[Olague, 1994] Olague, G. (1994). "Desarrollo de un filtro autónomo, empleando computación evolutiva." Tesis. Instituto Tecnológico de Chiuahua, México.

[Olague, 2002] Olague, G., y Hernández, B. (2002). "Flexible Model-based Multi-corner Detector for Accurate Measurements and Recognition." 16th International Conference on Pattern Recognition. IEEE Computer Society Press.

[Press, 1998] Press, W. H., Flanery, B.P., Teukolsky, S. A., y Vetterling, W. T. (1998). Numerical Recipies in C. Cambridge University Press, second edition.

[Råde, 1995] Råde L., y Westergren B. (1995). Mathematics Handbook for Science and Engineering. Studentlitteratur Lund, Sweden, and Birkhäuser USA.

[Rechenberg, 1973] Rechenberg, I. (1973). Evolutiostrategie: Optimieriung Technischer Systeme nach Prinzipien der Biologischen Evolution. Frommann-Holzboog, Stuttgart, Germany.

[Rohr, 1992] Rohr, K. (1992). "Recognizing Corners by Fitting Parametric Models." *International Journal of Computer Vision.* 9(3), 213-230. Kluwer Academic Press.

[Rosin, 1996] Rosin, P. (1996). "Augmenting Corner Descriptors." *Graphical Models and Image Processing*, 58(3), 286-294.

[Russ, 1995] Russ, J. C. (1995). The image processing handbook CRC Press, Inc., second edition.

[Salomon, 1998] Salomon, R. (1998). "Evolutionary Algorithms and Gradient Search: Similarities and Differences." *IEEE Transactions on Evolutionary Computation*, 2(2) 45-55.

[Schwefel, 1995] Schwefel, H. P. (1995). Evolution and Optimum Seeking John Wiley & Sons, Inc.

[1] Sohn, K., Kim, J. H., Alexander, W. E. (1998). "A Mean Field Annealing Approach to Robust Corner Detection." *IEEE Transactions on System, Man, and Cybernetics-Part B* 28, 82-90.

[Stammberger, 1998] Stammberger, T., Markus, M., Maximilian, R., Karl-Hans, E. (1998). "A Hierarchical Filter Schema for Efficient Corner Detection." *Pattern Recognition Letter*, 687-700. ELSEVIER.

[Stoer, 1980] Stoer, J. and Bulirsch, R. (1980). Introduction to Numerical Analysis. Spinger-Verlag New York Inc.

[Tsai, 1999] Tsai, D.-M., Hou, H.-T., Su, H.-J. (1999). "Boundary-base Corner Detection using Eigenvalues of Covariance Matrices." *Pattern Recognition Letters* 20, 31-40. ELSEVIER.

[Wang, 1995] Wang, H. and Brady, M. (1995). "Real-time Corner Detection Algorithms for motion Estimation." *Image and Vision Computing.* 13 (9).

[Zheng, 1999] Zheng, Z., Wang, H., Teoh, E. K. (1999). "Analysis of Gray Level Corner Detection." *Pattern Recognition Letters.* 20, 149-162.